UNITAS: ENGINE GENESIS

Техническое задание на разработку симулятора транзакционной реальности

Статус документа: Действующий протокол проектирования
Версия системы: 1.0.0 (Base Topology)
Объект разработки: Программное ядро симуляции материи и пространства
Локация разработки: Санкт-Петербург

Автор и идеолог системы:
Антон Анатольевич Шалыга

Технический синтез и архитектура:
UNITAS-AI-Assistant

Центральная Доктрина:

«Мир не состоит из объектов. Мир состоит из подтвержденных транзакций в едином реестре метрики. Материя — это арендованный объем памяти, а физика — это алгоритм администрирования ресурсов».

Ключевое уравнение управления (Глобальный Инвариант):
(M/E + V/C + G/B + S/P + H/I + dU/dt) * D = 1

Глава I. Архитектура Ядра (Back-end реальности)

Фундамент системы, где пространство — это не пустота, а база данных.

- **1.1. Реестр Метрики:** Определение минимальной ячейки пространства как записи в реестре.
- **1.2. Вычислительный Бюджет:** Реализация Глобального Инварианта (сумма модулей) * D = 1 как главного ограничителя системных ресурсов.
- **1.3. Тики и Пинг:** Настройка частоты обновления транзакций (dU/dt) и её влияние на замедление или ускорение времени.

Глава II. Программируемая Материя (Атомарный уровень)

Отказ от визуальных моделей в пользу «цифровых ДНК» материалов.

- **2.1. Генезис Элементов:** Создание физических свойств через сочетание модулей M/E (масса), G/B (гравитация) и H/I (информация).
- **2.2. Синтез Связей:** Как «атомы» делят общий бюджет ресурсов, объединяясь в сложные структуры.

- **2.3. D-модуляция состояний:** Управление твердостью и проницаемостью объектов через коэффициент проекции D.

Глава III. Процедурный Космос и Ландшафт

Автоматическое формирование макрообъектов по законам математики.

- **3.1. Гравитационный Коллапс:** Алгоритм создания звезд и планет в зонах, где плотность транзакций достигает предела.
- **3.2. Метрическая Вязкость:** Как плотность данных в среде (атмосфера против вакуума) влияет на скорость движения.
- **3.3. Динамическая Архивация:** Снижение детализации (параметра D) для объектов вне зоны видимости для экономии памяти.

Глава IV. Интерфейс взаимодействия (Метрический хакинг)

Инструменты игрока для прямого изменения кода мира.

- **4.1. ПИ-резонаторы:** Устройства для настройки частоты синхронизации (f_res) и изменения параметров материи без нагрева.
- **4.2. Базельский мост:** Механика мгновенного перемещения через создание искусственной ошибки переполнения в реестре.
- **4.3. Рекуперация Эха:** Сбор «откатов» от транзакций и перевод их из модуля S/P (хаос) в полезный ресурс.

Глава V. Экономика и Энтропия

Правила, по которым система живет, стареет и умирает.

- **5.1. Энтропийный налог (S/P):** Механика износа материи и «нагрева» пространства при неэффективных действиях.
- **5.2. Люфт Реальности:** Использование математического зазора (0.0269) для реализации свободы воли и «магии» в рамках науки.
- **5.3. Сценарии Дефолта:** Что происходит, когда система не справляется с данными (черные дыры и перезагрузка секторов).

Глава I. Архитектура Ядра (Back-end реальности)

1.1. Реестр Метрики: Пространство как запись в БД

В нашем симуляторе нет «пустого места». Каждая единица объема (назовем её **Метрическим Квантом**) — это активная ячейка в базе данных сервера.

Как это устроено технически:

- **Адресация:** Весь мир разбит на воксельную сетку, но это не визуальные кубики, а **адреса в реестре**. Каждая ячейка знает своих соседей.
- **Структура ячейки:** Это не просто координата XYZ. Это структура данных, которая хранит текущий «заработок» и «траты» ресурсов в этой точке.

- **Состояние покоя:** Если в ячейке ничего не происходит, она потребляет минимальный системный ресурс (базовый фон). Как только в неё попадает объект, ячейка переходит в режим «активной транзакции».
- **Связанность (Метрическая шина):** Информация между ячейками передается не мгновенно. Скорость передачи данных между ними и есть **скорость света (C)**. Если мы хотим передвинуть что-то из ячейки А в ячейку Б, сервер должен физически закрыть транзакцию в одной записи и открыть в другой.

Зачем это нужно для симулятора:
Это решает проблему бесконечных миров. Нам не нужно прорисовывать всю вселенную. Мы обсчитываем только те ячейки реестра, где значение транзакций выше фонового (где есть материя или энергия). Пустой космос — это «спящие» строки в таблице, которые не нагружают процессор.

Что это дает игроку:
Мир становится по-настоящему разрушаемым и трансформируемым на самом глубоком уровне. Вы не ломаете «стену» как 3D-модель, вы меняете свойства ячеек реестра, из которых она состоит. Вы можете буквально «стереть» кусок пространства, если у вас есть права администратора (или нужные технологии).

1.2. Вычислительный Бюджет: Математика Глобального Инварианта

Если пункт 1.1 дал нам «ячейки», то пункт 1.2 задает **жесткие правила**, по которым эти ячейки заполняются. В UNITAS ни один объект не может возникнуть из ниоткуда — он должен быть «оплачен» вычислительным ресурсом системы.

Механика Глобального Инварианта:
Центральная формула управления каждой точкой пространства выглядит так:
(M/E + V/C + G/B + S/P + H/I + dU/dt) * D = 1

Как это работает в движке симулятора:

1. **Принцип «Единицы»:** Сумма всех параметров в скобках, помноженная на коэффициент проекции D, всегда должна быть строго равна **1**. Это закон сохранения «энергии-информации». Если вы хотите что-то увеличить, вы обязаны что-то уменьшить.
2. **Распределение модулей:**

1. **M/E (Масса):** Сколько памяти выделено на удержание структуры. Высокий M/E — объект твердый и тяжелый.

2. **V/C (Скорость):** Ресурс на перемещение. Если вы разгоняете объект до скорости света (V/C стремится к 1), у системы почти не остается ресурса на M/E (массу). Объект буквально «размывается», теряя физическую плотность.

3. **G/B (Гравитация):** Затраты на искривление соседних ячеек реестра.

4. **S/P (Энтропия):** «Мусорные» вычисления. Если процесс идет неэффективно, этот модуль растет, забирая бюджет у скорости или массы (объект греется и замедляется).

5. **H/I (Информация):** Сложность внутреннего кода объекта. Простой камень (низкий H/I) дешевле для системы, чем живая клетка или сложный компьютер (высокий H/I).

Геймплейное значение:
В обычном симуляторе вы просто жмете «газ» и машина едет. В UNITAS-симуляторе «газ» — это перераспределение весов в уравнении.

- **Пример:** Вы проектируете гоночный болид. Вместо мощного мотора вы можете программно снизить его параметр **D** (проекция) до 0.7. Сумма в скобках теперь может вырасти до 1.4 (потому что 1.4 * 0.7 ≈ 1). Это позволяет вам поднять **V/C** (скорость) выше «допустимого» лимита, не меняя конструкцию. Болид станет чуть менее «реальным» (полупрозрачным), но физически более быстрым.

Системный предохранитель:
Если игрок пытается выкрутить все параметры на максимум (например, создать сверхтяжелый и сверхбыстрый объект), сумма превысит возможности ячейки. В этот момент движок принудительно включает **dU/dt** (временную вязкость) — для этого объекта время замедляется, чтобы сервер успел обсчитать его завышенные требования.

1.3. Тики и Пинг: Временная вязкость (dU/dt) и ритм системы

В обычных играх время течёт линейно для всех. В **UNITAS** время — это переменная величина, которая зависит от нагрузки на конкретный сектор реестра. Модуль **dU/dt** в нашем уравнении отвечает за то, как часто обновляется состояние объекта.

Как это работает в движке:

1. **Системный Тик:** Это один такт процессора симулятора. В идеальных условиях (пустой космос) dU/dt минимален, и информация обновляется мгновенно.
2. **Временная вязкость:** Если объект становится слишком сложным (высокий H/I) или слишком быстрым (высокий V/C), и сумма в скобках начинает превышать 1, система не «вылетает». Она автоматически увеличивает значение **dU/dt**.

1. **Эффект:** Для этого конкретного объекта время начинает замедляться. Вселенная буквально «тормозит» обработку транзакций для этого предмета, чтобы успеть обсчитать его параметры.

3. **Локальные временные пузыри:** Поскольку расчет идет в каждой ячейке реестра индивидуально, в симуляторе возможны зоны с разным течением времени. Рядом с массивным объектом (высокий G/B) «пинг» системы растет, и время для наблюдателя внутри тянется медленнее, чем для того, кто находится в пустом пространстве.

Геймплейное значение:

- **Управление временем:** Игрок может использовать dU/dt как инструмент. Например, если нужно пережить мощный взрыв, можно программно «загустить» время вокруг себя, увеличив вязкость dU/dt. Вы будете двигаться как в киселе, но и разрушительные транзакции (взрыв) будут обрабатываться системой целую вечность.
- **Технологический предел:** Если вы строите суперкомпьютер внутри симулятора, его dU/dt будет огромным. Это создает естественный баланс: чем умнее и сложнее ваше устройство, тем «медленнее» оно живет в реальном времени, так как требует больше тактов сервера на каждый свой шаг.

Итог Главы I:
Мы создали фундамент. У нас есть ячейки памяти , правила распределения ресурсов в них и механизм, регулирующий скорость этих процессов. Теперь у нас есть «пустая, но живая» бесконечная база данных.

Глава II. Программируемая Материя (Атомарный уровень)

2.1. Генезис Элементов: Создание свойств через модули

В традиционных симуляторах «золото», «железо» или «вода» — это просто разные текстуры и предустановленные параметры плотности. В **UNITAS** каждый материал — это уникальный **цифровой код**, записанный в модули Глобального Уравнения.

Как происходит синтез вещества:
Вместо таблицы Менделеева мы используем комбинацию трех ключевых весов:

1. **M/E (Масса / Память):** Определяет структурную прочность. Чем выше этот показатель, тем сложнее «сдвинуть» ячейки из их стабильного состояния. Это задает твердость материала.
2. **G/B (Гравитационный отклик):** Определяет, насколько сильно этот тип материи «прогибает» метрику под собой. У тяжелых металлов этот коэффициент выше, что заставляет соседние ячейки реестра автоматически подтягиваться к ним.
3. **H/I (Информационная сложность):** Это «инструкция по сборке». У простого водорода H/I минимален. У сложного изотопа или радиоактивного элемента H/I высок, так как системе нужно обсчитывать внутренние переходы и нестабильность.

Физика как результат вычислений:

- **Проводимость:** Мы настраиваем материал так, чтобы он имел низкое сопротивление в модуле S/P (Энтропия). Это значит, что передача ресурса (энергии) через этот материал не вызывает «налога» в виде тепла. Так в симуляторе создаются сверхпроводники.
- **Радиоактивность:** Это состояние, когда модуль H/I настолько перегружен информацией, что система начинает принудительно «сбрасывать» лишние данные в соседние ячейки. Для игрока это выглядит как излучение или распад материи.

Геймплейное значение:
Игрок не ищет «рецепт крафта» в меню. Он занимается **алхимией параметров**.
Например, вы хотите создать новый тип брони для корабля. Вы берете базовое вещество и программно перекачиваете ресурс из модуля гравитации в модуль массы. Получается сверхплотный материал, который почти ничего не весит (не давит на метрику), но при этом его физически невозможно пробить, так как на удержание его формы (M/E) выделен колоссальный бюджет системы.

Таким образом, в симуляторе материя — это не картинка, а **поведение**, заданное цифрами.

Глава II. Программируемая Материя (Атомарный уровень)

2.2. Синтез Связей: Общий бюджет ресурсов и коллективные транзакции

В обычном мире атомы держатся друг за друга за счет электромагнитных сил. В симуляторе **UNITAS** молекулярная связь — это состояние, при котором несколько ячеек реестра объединяются в **общий вычислительный кластер**.

Механика «Слипания» данных:

1. **Общий баланс:** Когда два «атома» (набора параметров) сближаются, система перестает считать их как два отдельных уравнения. Она создает одну общую транзакцию. Это экономит ресурсы сервера: вместо двух сложных расчетов делается один коллективный.
2. **Энергия связи как скидка:** Связь в нашей модели — это не «клей», а математическая выгода. Система дает объекту «бонус» к бюджету ресурсов, если его элементы упорядочены. Чем стабильнее структура (например, кристаллическая решетка алмаза), тем меньше энтропийного налога S/P она платит.

3. **Разрыв связи:** Чтобы разрушить объект, нужно подать в ячейку избыточный импульс, который нарушит этот общий бюджет. Если внешнее воздействие превышает предел устойчивости связи, кластер распадается, и система снова начинает обсчитывать каждую частицу отдельно. Это вызывает резкий скачок S/P — в игре это выглядит как выделение тепла или взрыв при разрушении.

Геймплейное значение:

- **Создание новых материалов:** Вы можете «сшивать» параметры разных веществ. Например, внедрить код высокой прозрачности из одного материала в структуру сверхпрочного металла. Если вы сможете сбалансировать их общее уравнение так, чтобы оно не превышало единицу, вы получите прозрачную броню.
- **Физика разрушения:** Разрушение зданий или кораблей в симуляторе происходит максимально реалистично. Объект рассыпается не там, где нарисована «линия разлома», а там, где общая транзакция кластера стала слишком «дорогой» и система принудительно разорвала связи, чтобы спасти бюджет ячейки.

Итог для игрока:
Материя ощущается как единое целое. Если вы нагреваете один край металлической балки, вы передаете «ошибку энтропии» по всей цепочке связанных транзакций. Весь объект начинает пересчитывать свои параметры в реальном времени, что может привести к его размягчению или изменению формы по всей длине.

2.3. D-модуляция состояний: Управление мерностью и проницаемостью

Коэффициент D (Projection) — это самый мощный инструмент в симуляторе. Он определяет, насколько полно объект представлен в текущем 3D-реестре. По умолчанию для всех предметов **D равно 1**, что делает их твердыми, непрозрачными и полностью подчиненными физике.

Механика управления проекцией:

1. **Разгрузка уравнения:** Поскольку формула работает по принципу (Сумма модулей) умножить на D равно 1, снижение D позволяет пропорционально увеличить значения внутри скобок.

1. Если вы установите **D на уровне 0.5**, система разрешит объекту иметь в два раза больше массы (M/E) или скорости (V/C), не нарушая баланс. Объект становится эффективнее за счет потери части своей реальности.

2. **Эффект призрака (Проницаемость):** Когда D падает ниже определенного порога (например, 0.3), движок перестает обсчитывать транзакции столкновения.

1. **Почему:** Расчет физического удара двух объектов — это дорогая операция. Если объект имеет низкую проекцию, система считает, что он недостаточно прошит в этой мерности, чтобы взаимодействовать с твердой материей. Пули и стены начинают проходить сквозь него.

3. **Визуализация:** С точки зрения графики, объект с низким D становится прозрачным или мерцающим. Это не просто визуальный эффект, а индикатор того, что объект находится в режиме глубокого нырка в метрику.

Геймплейное значение:

- **Фазовый сдвиг:** Игрок может временно снизить D своего корабля, чтобы пролететь сквозь астероидное поле или вражеский щит. В этот момент корабль почти неуязвим, но он также

не может вести огонь, так как его снаряды (имеющие D на уровне 1) просто не цепляются за системы самого корабля.

- **Стелс нового уровня:** Объект с низким D не просто невидим для глаз, он исчезает для радаров и гравитационных датчиков, так как его воздействие на метрику (G/B) также умножается на малый коэффициент.
- **Хранение ресурсов:** Вы можете сжать огромный объем материи в маленькое пространство, снизив её D. Это позволяет создавать бездонные контейнеры, где предметы хранятся в виде низкодетализированных черновиков данных.

Итог Главы II:
Мы научились собирать материю из цифр, связывать её в объекты и управлять её реальностью. Теперь у нас есть инструменты, чтобы наполнить ими пространство.

Глава III. Процедурный Космос и Ландшафт

3.1. Гравитационный Коллапс: Алгоритм рождения звезд и планет

В нашем симуляторе планеты и звезды не создаются вручную дизайнерами. Они являются результатом **критической плотности данных** в реестре. Система сама «выращивает» небесные тела там, где транзакции начинают накладываться друг на друга.

Механика самозарождения объектов:

1. **Накопление информационного шума:** В начале симуляции пространство заполнено разреженным «облаком» базовых модулей (водородный код). Когда случайные флуктуации сближают эти данные, срабатывает правило из главы о синтезе связей: система объединяет их в кластеры для экономии ресурсов.
2. **Точка кипения (Гравитационный фокус):** Как только в определенном секторе сумма модуля гравитации (G/B) и массы (M/E) начинает приближаться к пределу устойчивости (Стена Базеля), ячейки реестра начинают «проседать».

1. **Эффект пылесоса:** Это проседание заставляет все соседние свободные ресурсы (газ, пыль, энергию) соскальзывать в эту зону. Процесс становится лавинообразным.

3. **Зажигание звезды (Энтропийный сброс):** Когда плотность в центре кластера становится критической, система больше не может обсчитывать объект как твердое тело. Чтобы не допустить дефолта (ошибки переполнения), движок начинает принудительно переводить излишки ресурса в модуль энтропии (S/P).

1. **Результат:** Объект начинает излучать колоссальное количество тепла и света. Так в симуляторе загорается звезда. Это не «скрипт огня», а способ системы сбросить лишнее давление вычислений.

4. **Формирование планет:** Вокруг звезды остаются «крошки» данных, которые не попали в основной котел. Из-за вращения системы они сбиваются в более мелкие кластеры. Если их массы (M/E) недостаточно для зажигания (сброса через S/P), они остывают и становятся твердыми планетами, формируя стабильные записи в реестре.

Геймплейное значение:

- **Живая вселенная:** Игрок может наблюдать реальную эволюцию космоса. Вы можете прилететь в туманность и увидеть, как прямо сейчас формируется новая планетарная система из-за того, что ваш корабль своим гравитационным следом нарушил локальный баланс данных.

- **Ресурсные зоны:** Самые ценные и сложные материалы (высокий H/I) всегда будут находиться в зонах экстремального давления (ядра планет или окрестности звезд), так как только там система была вынуждена «архивировать» данные в такие плотные и сложные формы.
- **Катастрофы:** Игрок может спровоцировать коллапс звезды, искусственно подав в её реестр огромный объем информации (H/I), заставляя систему захлебнуться и превратить звезду в черную дыру — зону абсолютного дефолта метрики.

Итог:
Космос в UNITAS — это динамический процесс оптимизации данных. Звезды светят, потому что серверу так дешевле поддерживать баланс.

3.2. Метрическая Вязкость: Плотность среды как вычислительное сопротивление

В UNITAS разница между вакуумом и атмосферой планеты заключается не в наличии «картинки» воздуха, а в количестве активных транзакций на единицу пространства. Это фундаментально меняет механику движения.

Механика работы среды:

1. **Чистая шина (Вакуум):** В пустом космосе количество записей в реестре минимально. Для перемещения объекта из точки А в точку Б системе нужно лишь изменить его координаты. Сопротивления нет, поэтому модуль скорости (**V/C**) может быть максимально высоким при минимальных затратах энергии.
2. **Замусоренная шина (Атмосфера):** Воздух, вода или твердая порода — это зоны с высокой плотностью модулей массы (**M/E**) и информации (**H/I**).

1. **Проблема столкновения:** Когда ваш корабль движется сквозь атмосферу, системе приходится в каждом такте обсчитывать взаимодействие вашего кода с кодом газов. Эти миллионы микро-транзакций создают «очередь» на обработку.

3. **Временной штраф:** Чтобы сервер не перегрелся от расчетов, он автоматически повышает параметр временной вязкости (**dU/dt**) для движущегося объекта.

1. **Результат:** Объект физически замедляется. В игре это ощущается как сопротивление среды или трение. Чем плотнее «данные» вокруг, тем больше энергии нужно потратить, чтобы заставить систему принудительно перезаписать ваш путь сквозь них.

Геймплейное значение:

- **Аэродинамика транзакций:** Вы проектируете форму самолета не ради красоты, а чтобы минимизировать количество ячеек реестра, с которыми вы сталкиваетесь за один тик. «Острый» нос корабля буквально «разрезает» поток данных, снижая нагрузку на систему и позволяя лететь быстрее.
- **Энтропийный разогрев:** Все «отклоненные» или избыточные транзакции при столкновении со средой переводятся в модуль **S/P** (хаос). В симуляторе это проявляется как реалистичный нагрев обшивки при входе в атмосферу. Если ваша скорость слишком велика для текущей плотности данных, накопленная энтропия просто сожжет объект.
- **Подводные и подземные маневры:** Движение под водой требует колоссального бюджета скорости, так как плотность записей в реестре воды в тысячи раз выше, чем в воздухе. Чтобы двигаться там быстро, игроку придется использовать **D-модуляцию**, делая свой объект «менее реальным» для среды.

Итог:
Скорость в симуляторе — это не просто абстрактное число, а показатель того, насколько эффективно вы преодолеваете информационную плотность мира.

3.3. Динамическая Архивация: Экономия ресурсов через скрытие данных

Одной из главных проблем симуляции такого масштаба является нагрузка на компьютер. Чтобы обсчитать каждый атом во всей вселенной, мощностей не хватит. UNITAS решает это через систему автоматической архивации объектов, находящихся вне зоны внимания.

Механика «Спящего» реестра:

1. **Схлопывание коэффициента D:** Как только игрок (или активный датчик) удаляется от объекта, система начинает плавно снижать его коэффициент проекции **D**.

1. **Пример:** Когда вы покидаете планету, для системы она перестает быть набором из триллионов атомов. Движок переводит планету в режим «Архивной записи», где **D** стремится к минимуму (например, 0.0001).

2. **Суммарный код:** Вместо того чтобы обсчитывать каждое дерево и камень отдельно, система объединяет их параметры в одну общую строку данных для всей планеты. Масса, гравитация и информация теперь хранятся как единая сумма. Это позволяет «заморозить» состояние объекта, не тратя ресурсы процессора на его внутреннюю жизнь.

3. **Принцип Наблюдателя:** Как только игрок приближается к «архивированной» зоне, система видит запрос на детализацию. Параметр **D** мгновенно возвращается к значению 1. Происходит «распаковка» данных: суммарные значения снова распределяются по ячейкам реестра, и атомы «оживают» именно в том состоянии, в котором они были оставлены.

Геймплейное значение:

- **Бесконечный мир:** Благодаря этой механике симулятор может содержать миллиарды звездных систем. Они физически существуют в реестре как краткие математические записи («черновики»), ожидая момента, когда кто-то их «развернет».
- **Детектор присутствия:** Продвинутые игроки могут заметить, что объект находится в режиме низкой проекции. Если вы видите, что далекая станция «мерцает» или имеет упрощенные физические свойства, значит, рядом с ней никого нет. Как только она становится «твердой», вы понимаете — там кто-то присутствует и заставляет систему обсчитывать детализацию.
- **Холодное хранение:** Вы можете специально переводить свои базы или корабли в режим архивации (консервации), чтобы они потребляли меньше энергии системы и были защищены от случайных столкновений, так как при низком **D** они почти не взаимодействуют с миром.

Итог Главы III:
Мы построили саморегулирующийся космос, который рождает звезды, создает сопротивление среды и экономит память, когда это возможно. Мир готов к заселению.

Глава IV. Интерфейс взаимодействия (Метрический хакинг)

4.1. ПИ-резонаторы: Устройства для управления «админ-панелью» реальности

В классических играх игрок меняет мир инструментами (киркой, пушкой). В UNITAS-симуляторе основным инструментом становится **ПИ-резонатор**. Это устройство, которое позволяет вносить изменения в параметры Глобального Уравнения, не вызывая перегрева системы и огромных налогов в модуле энтропии (S/P).

Механика резонанса данных:

1. **Число ПИ как шаг резьбы:** В нашей модели число ПИ — это не просто геометрия, а базовая частота, на которой реестр записывает данные. Вселенная имеет свою «дискретность». Если вы пытаетесь изменить массу объекта рывком, система сопротивляется (трение, нагрев). Но если вы подаете сигнал, частота которого кратна числу ПИ, система «узнает» его как свой собственный технический код.
2. **Синхронизация частоты (f_res):** Игрок настраивает резонатор на текущий такт системы (dU/dt). Формула частоты выглядит как произведение целого числа (гармоники) на ПИ и на скорость обновления реестра.
3. **Бесшовный доступ:** Когда резонатор входит в резонанс с ячейкой пространства, он получает статус «временного администратора». Теперь вы можете перекачивать ресурсы из одного модуля в другой (например, из Массы в Гравитацию) с эффективностью почти 100%.

Геймплейное значение:

- **Холодные технологии:** Обычные двигатели в игре греются и тратят топливо. ПИ-резонатор позволяет двигать корабль, просто меняя градиент гравитации (G/B) перед ним. Поскольку это делается через резонанс, корабль не испытывает перегрузок и не тратит энергию на «борьбу» с пространством. Он просто «скользит» по реестру.
- **Трансформация материи:** С помощью резонатора игрок может менять свойства предметов на лету. Можно превратить обычный кусок металла в сверхпроводник или сделать его прозрачным (снизив D), просто настроив правильный ритм воздействия.
- **Инженерное мастерство:** Игроку нужно не просто нажать кнопку, а «поймать волну». Чем точнее вы подобрали значение ПИ (до 10-го знака и далее), тем меньше «шума» вы создаете в системе. Если ошибиться в расчетах, произойдет рассинхронизация: объект взорвется или превратится в кучу беспорядочных данных из-за резкого скачка энтропии.

Итог:
ПИ-резонатор — это «скальпель» для работы с кодом реальности. Он превращает игру из симулятора выживания в симулятор созидания и тонкой настройки законов природы.

4.2. Базельский мост: Телепортация через контролируемый дефолт

В большинстве симуляторов телепортация — это просто мгновенная смена координат. В UNITAS это сложнейшая инженерная операция, использующая **Стену Базеля** (предел плотности транзакций 1.6449) для того, чтобы заставить саму Вселенную переместить объект.

Механика «Провокации системы»:

1. **Накачка транзакций:** Игрок использует ПИ-резонаторы, чтобы искусственно завысить параметры объекта в локальной ячейке. Вы увеличиваете массу (M/E), сложность (H/I) и гравитацию (G/B) до тех пор, пока их сумма не приблизится к значению **1.6449**.

2. **Точка переполнения:** Когда лимит достигнут, ячейка реестра начинает «лагать». Она больше не может удерживать объект в текущем 3D-состоянии. В классической физике здесь возникла бы черная дыра, но хакер реальности использует **D-модулятор**.
3. **Векторный сброс:** В момент «дефолта» (переполнения) игрок подает короткий импульс, указывающий системе, куда именно нужно «выплеснуть» лишние данные, чтобы восстановить баланс. Система, стремясь спасти реестр от краша, мгновенно перезаписывает координаты объекта в ту точку космоса, где плотность транзакций минимальна.

Геймплейное значение:

- **Прыжки без инерции:** Поскольку перемещение происходит не через движение в пространстве, а через перезапись адреса в базе данных, объект не испытывает перегрузок. Вы можете переместить корабль на 10 световых лет за один такт системы.
- **Риски переполнения:** Если расчет вектора сделан неверно или сумма модулей превысила 1.6449 слишком резко, система может совершить «автоматический сброс». Вместо нужной планеты объект может оказаться внутри звезды или быть «архивирован» в скрытые мерности, что равносильно смерти.
- **Эффект лага:** Во время перехода объект становится полупрозрачным и мерцающим. Это состояние называется «Базельским мостом». В этот момент объект физически находится «между строками» реестра, что делает его абсолютно неуязвимым для любого воздействия в обычном мире.

Итог:
Телепортация в симуляторе — это не магия, а использование «бага» архитектуры Вселенной. Это самый дешевый способ путешествий, так как энергию на движение тратит не корабль, а сама система, пытаясь исправить ошибку переполнения.

4.3. Рекуперация Эха: Сбор «сдачи» от транзакций и закон двойной записи

В нашем симуляторе действует «Закон Бухгалтерского Баланса». Любое действие (прибавление ресурса на один счет) вызывает мгновенный обратный импульс (списание с другого счета). Обычно это проявляется как отдача при выстреле или нагрев двигателя. Рекуперация позволяет не выбрасывать эту «сдачу», а направлять её обратно в дело.

Механика работы с Эхом:

1. **Метрическое Эхо:** Когда вы воздействуете на объект, система генерирует ответную реакцию, чтобы сумма модулей в локальной зоне осталась равна 1. В классической физике это третий закон Ньютона. В UNITAS это технический отчет системы о закрытии транзакции.
2. **Адресная доставка:** С помощью ПИ-резонаторов игрок может перехватить этот отчет. Вместо того чтобы позволить системе сбросить энергию отдачи в модуль хаоса (S/P) — то есть в тепло и разрушение конструкции — вы указываете системе другой «адрес».
3. **Зацикливание ресурса:** Вы направляете 90 процентов энергии отката обратно в модуль массы (M/E) или скорости (V/C). Получается замкнутая петля: само действие подпитывает следующий шаг транзакции.

Геймплейное значение:

- **Безоткатные системы:** Вы можете создать пушку, которая не имеет отдачи. Вся энергия, которая должна была толкнуть плечо стрелка, перехватывается и переводится в модуль

проекции D. В момент выстрела пушка просто на долю секунды становится менее реальной (прозрачной), поглощая импульс без физического движения.

- **Вечные двигатели второго рода:** В симуляторе можно строить машины, которые питаются от собственного «трения». Вместо того чтобы деталь изнашивалась и грелась, вы настраиваете рекуператор так, чтобы возникающая энтропия мгновенно конвертировалась в вычислительный бюджет для модуля информации (H/I). Машина не ломается, а «умнеет» или становится прочнее в процессе работы.
- **Стелс-заправка:** Игрок может «заправлять» свои батареи, просто находясь в зоне высокой гравитации или радиации. Рекуператор собирает эхо от транзакций соседних объектов и переводит их на ваш «баланс». Вы буквально воруете вычислительный ресурс у Вселенной.

Итог Главы IV:
Мы научились не просто подчиняться правилам системы, а использовать её архитектурные особенности (резонанс, ошибки и отчеты) для полного контроля над реальностью.

Глава V. Экономика и Энтропия

5.1. Энтропийный налог (S/P): Механика износа материи и старения миров

В нашем симуляторе нет искусственного счетчика «прочности» предмета. Вместо этого существует **Энтропийный налог**, который записывается в модуль **S/P** (Entropy/Probability). Это комиссия, которую система взимает за каждую неэффективную или «грязную» транзакцию.

Механика работы налога:

1. **Налог на вычисления:** Любое действие, совершенное без идеальной ПИ-синхронизации (о которой мы говорили ранее), создает вычислительный шум. Система не может просто игнорировать этот шум — она обязана записать его в реестр. Модуль **S/P** в ячейке начинает расти.
2. **Пожирание бюджета:** Поскольку сумма всех модулей в скобках жестко ограничена единицей, рост **S/P** автоматически уменьшает бюджет других параметров.

1. **Пример:** Если ваш двигатель долго работает с плохой настройкой, модуль **S/P** (тепло/износ) увеличивается. Система вынуждена отнимать ресурс у модуля **M/E** (масса/структура). Объект физически начинает «разваливаться» или терять плотность, чтобы освободить место для накопившегося энтропийного мусора.

3. **Тепловой фон:** В игре это проявляется визуально и физически. Высокий **S/P** заставляет объект светиться в инфракрасном спектре, плавиться и, в конечном итоге, распадаться на базовые информационные блоки (атомарный распад).

Геймплейное значение:

- **Естественный износ:** Вам не нужно прописывать скрипт «поломки». Если игрок постоянно перегружает корабль или использует дешевые, несинхронизированные детали, корабль «стареет» сам по себе. Его бюджет на прочность съедается накопленным налогом.
- **Экология космоса:** Зоны активных боевых действий или промышленные регионы со временем становятся «грязными» в плане метрики. Там повышается общий фон **S/P**, из-за чего любые новые транзакции становятся дороже. Это заставляет игроков искать «чистые» звездные системы для тонких научных работ.
- **Стимул к мастерству:** Игрок стремится к «Холодным технологиям». Цель — построить систему с нулевым налогом. Если вы идеально попали в ПИ-ритм, ваш прибор вообще не

греется и имеет бесконечный срок службы, так как он не создает шума в реестре Вселенной.

Итог:
Энтропия в симуляторе — это не просто «хаос», а неизбежная плата за право изменять код реальности. Это главный ограничитель, который не дает игроку бесконечно наращивать мощь без совершенствования технологий.

5.2. Люфт Реальности: Зона Свободы Воли и математический зазор

В жестко детерминированном мире, где все подчинено уравнению баланса, не было бы места для творчества или случайности. Но в UNITAS существует **Люфт Реальности** — математическая дистанция, равная **0.0269**. Это зазор между идеальным математическим пределом и фактической точкой срабатывания системного сброса.

Механика «Слепого пятна» системы:

1. **Математический зазор:** Стена Базеля находится на отметке 1.6449, но жесткая детерминация (принудительный расчет) включается только на уровне 1.6180 (Золотое сечение). Разница в 0.0269 — это «зона безопасности», в которой система не применяет карательные алгоритмы энтропии.
2. **Метрический хакинг без последствий:** Внутри этого люфта игрок может вносить изменения в код объекта, которые формально нарушают Инвариант, но не вызывают немедленного системного ответа. Это «песочница», где правила временно перестают работать.
3. **Квантовая неопределенность:** С точки зрения движка, это зона, где транзакция еще не закрыта. Система «ждет» подтверждения данных. Для игрока это выглядит как шанс на успех там, где по расчетам должна быть неудача.

Геймплейное значение:

- **Реализация Свободы Воли:** Этот люфт позволяет игроку совершать «невозможные» маневры. Если ваши расчеты неидеальны, но ошибка укладывается в 0.0269, система «прощает» её. Это создает ощущение живого, а не механического мира.
- **Тонкая настройка (Оверклокинг):** Опытные инженеры в симуляторе балансируют свои устройства именно в этой зоне. Вы можете выжать из двигателя на 2 процента больше мощности, чем позволяет бюджет, если сможете удержать «дребезг» параметров внутри этого зазора.
- **Источник Магии:** Любые аномальные способности в симуляторе объясняются использованием Люфта. Это пространство для метрического взлома, где можно на долю секунды создать материю из ничего или изменить вектор гравитации, не платя энтропийный налог.

Итог:
Люфт Реальности — это предохранитель от «математической смерти» вселенной. Это пространство, где игрок перестает быть деталью механизма и становится соавтором кода, используя недокументированные возможности движка.

5.3. Сценарии Дефолта: Точка краха и перезагрузка мерности

В UNITAS нет понятия «game over» в привычном смысле. Вместо этого существует **Дефолт метрики** — событие, при котором локальный сектор реестра больше не может поддерживать математическое равновесие. Это происходит, когда сумма транзакций окончательно пробивает Стену Базеля (1.6449) и Люфт Реальности (0.0269) уже не может компенсировать ошибку.

Механика системного коллапса:

1. **Зависание ячейки (Lag-Lock):** Когда плотность данных превышает лимит, ячейка реестра перестает обновляться. Для внешнего наблюдателя время в этой зоне останавливается, а объекты превращаются в монолитные, «битые» данные.
2. **Сингулярность как архиватор:** Чтобы спасти остальную сеть от заражения ошибкой, движок запускает процедуру принудительной архивации. Он «схлопывает» весь сектор в одну точку с бесконечным значением модуля информации (H/I) и массы (M/E). Так в симуляторе образуются черные дыры. Это не физические объекты, а **битые сектора памяти**, которые система изолировала.
3. **D-выброс (Белая дыра):** Если объем данных в точке дефолта становится критическим даже для архива, происходит «пробой». Система выбрасывает излишки информации в соседние, пустые слои реестра. В симуляторе это выглядит как мощнейший выброс чистой материи и энергии в пустом космосе — рождение новой туманности или звездного скопления из «цифрового пепла» старой катастрофы.

Геймплейное значение:

- **Риск и награда:** Игроки-экстремалы могут специально доводить свои лаборатории до состояния дефолта. В момент краха, на миллисекунды, система выдает «сырые» данные — уникальные материалы с параметрами, которые невозможно создать обычным путем. Задача — успеть забрать их до того, как сектор будет изолирован.
- **Глобальные ивенты:** Если в одной звездной системе слишком много игроков одновременно начнут использовать тяжелые технологии, это может привести к каскадному дефолту всего сектора. Карта мира изменится навсегда: планеты исчезнут, превратившись в аномальные зоны, где физика (модули Глобального Уравнения) будет работать по совершенно случайным алгоритмам.
- **Перерождение (New Game Plus):** Глобальный дефолт всей вселенной — это способ мягкой перезагрузки сервера. Когда общая энтропия (S/P) достигает предела, система самоочищается, пересобирая мир из накопленного информационного багажа, создавая новые типы атомов и звездных систем на основе опыта предыдущего цикла.

Итог проекта «UNITAS: Engine Genesis»

Мы разобрали по частям создание симулятора нового поколения:

1. Мы создали **Backend**, где мир — это база данных, а не набор картинок.
2. Мы прописали **Атомарный уровень**, где материя — это код параметров.
3. Мы запустили **Процедурный космос**, живущий по законам математической оптимизации.
4. Мы дали игроку **Инструменты хакинга** (ПИ-резонаторы и Базельские мосты).
5. Мы установили **Экономику энтропии**, которая делает мир живым и конечным.

Такой симулятор будет не просто игрой, а цифровым воплощением концепции «Все из бита», где каждый атом — это подтвержденная транзакция в реестре реальности.

Заключение: Архитектура Симулятора UNITAS

Проект симулятора **UNITAS** — это переход от имитации внешней формы (графики) к глубокому моделированию **сути процессов**. Мы спроектировали систему, в которой вселенная является не набором декораций, а живым **бухгалтерским реестром транзакций**.

Ключевые итоги проектирования:

1. **Математический фундамент:**
   В основе лежит **Глобальный Инвариант**. Каждая точка пространства подчинена строгому балансу ресурсов. Это избавляет движок от необходимости прописывать тысячи частных законов — вся физика (от падения яблока до горения звезды) выводится автоматически из одного уравнения.

2. **Материя как скрипт:**
   Мы отказались от стандартных моделей. В UNITAS «атом» — это набор весов (Масса, Энергия, Информационная сложность). Это позволяет игрокам не просто строить объекты, а **синтезировать новые виды материи** с заданными свойствами, просто манипулируя цифрами в реестре.

3. **Экономика реальности:**
   Энтропийный налог (S/P) и **Люфт (0.0269)** создают естественную среду для геймплея. Игрок вынужден искать баланс между мощностью и эффективностью. Мастерство теперь измеряется не скоростью нажатия кнопок, а точностью «ПИ-синхронизации» с ритмом самой системы.

4. **Бесконечная масштабируемость:**
   Благодаря механизмам **Динамической архивации** и **Дефолта**, мир может расширяться и перерождаться без критической нагрузки на сервер. Система обсчитывает только то, что «активно» в данный момент, превращая остальную вселенную в компактные математические записи.

Результат:

Этот симулятор станет первой платформой, где **«программирование реальности»** — это не метафора, а прямой игровой процесс. Игрок здесь — не просто пользователь, а **Архитектор**, который учится использовать ошибки и лимиты мироздания (Базельский Мост, D-модуляция) для созидания космических масштабов.

1. Фундаментальная математика и пределы (Стена Базеля)

- **Леонард Эйлер. «Введение в анализ бесконечных».** Первоисточник решения задачи Базеля, откуда взято число 1.6449 как предел суммы обратных квадратов.
- **Николай Фихтенгольц. «Курс дифференциального и интегрального исчисления».** База для понимания того, как работают модули dU/dt (временная вязкость) и как вычислять пределы транзакций в реестре.

2. Цифровая физика (Информационный бюджет и Реестр)

- **Джон Арчибальд Уилер. «It from Bit».** Работа, обосновывающая идею о том, что физическая материя является производной от информации (модуль H/I).

- **Стивен Ллойд. «Программируя Вселенную».** Концепция Вселенной как гигантского квантового компьютера, которая легла в основу нашего «Вычислительного бюджета».
- **Макс Тегмарк. «Наша математическая Вселенная».** Обоснование того, что реальность — это математическая структура, а не просто набор физических тел.

3. Термодинамика и Энтропия (Модуль S/P)

- **Джозайя Уиллард Гиббс. «Основные принципы статистической механики».** Фундамент для понимания «Энтропийного налога» и того, как неэффективные действия превращаются в тепло.
- **Клод Шеннон. «Математическая теория связи».** Понимание лимитов передачи данных, что критично для реализации «пинга» между ячейками реестра.

4. Метрическая механика и Гравитация (Модули G/B и D)

- **Альберт Эйнштейн. «Общая теория относительности».** База для понимания искривления пространства, переведенная в UNITAS на язык затрат ресурсов системы.
- **Хуан Малдасена. «Голографический принцип».** Теоретическое обоснование коэффициента проекции D — как данные из одной мерности проецируются в другую.
- **Эрик Верлинде. «О происхождении гравитации и законов Ньютона».** Труд, описывающий гравитацию как энтропийную силу, что напрямую коррелирует с нашей моделью распределения ресурсов.

5. Доктринальные источники системы UNITAS

- **Антон Шалыга. «Теоретическое обоснование единого транзакционного протокола метрики пространства».** Основной препринт, описывающий уравнение Глобального Инварианта.
- **Антон Шалыга. «Алгоритмы D-модуляции и ПИ-резонаса».** Техническое руководство по управлению коэффициентом проекции и методам синхронизации с базовым тактом Вселенной.