# План разработки RTS-проекта (Unity 2D) – Полный Production Pipeline

Цель: В течение квартала пошагово создать и отладить полный сквозной production-пайплайн для RTS-игры (Unity 2D) с упором на архитектуру и оптимизации. План предназначен для джуниор-разработчика и охватывает все ключевые подсистемы – от архитектурного каркаса проекта до геймплейных механик, AI, UI, производительности и CI/CD. Документ структурирован по подсистемам и этапам разработки, служит внутренней документацией и трекером прогресса.

## Модульная структура и архитектура проекта

Проект следует модульной архитектуре с четким разделением на слои (Domain, Data, Application, AI, Presentation, Infrastructure). Ниже приведена структура solution с пространствами имен:

/Game  
 /Domain // Чистые модели (POCO) и интерфейсы  
 Economy/ // Экономика: бюджеты, производство, издержки  
 Infection/ // Состояния клеток, параметры заражения  
 Tech/ // Дерево технологий и эффекты  
 Map/ // Координаты, графы смежности, зоны  
 AI/ // Контракты AI (IStrategicPlanner, IUtilityScorer и др.)  
 Common/ // Общие компоненты: EventBus, команды, результаты, время  
 /Data // ScriptableObject-конфиги (статические данные)  
 Units/, Buildings/, Techs/, Factions/, InfectionParams/  
 /Application // Сервисы приложения: симуляция, тик, сохранения  
 Simulation/, Saving/, Pathfinding/, Services/  
 /AI // Реализация AI: Utility AI, GOAP, Behavior Trees  
 Strategic/, Tactical/, Perception/  
 /Presentation // Презентация: MonoBehaviour для сцен, UI, визуальных эффектов  
 UI/, VFX/, Audio/, Tilemaps/  
 /Infrastructure // Инфраструктура: доступ к данным, JSON, DI/Service Locator

Расширения модулей: По мере разработки добавляются детали в каждую подсистему. Например, в Domain/Economy следует разделить две ветви экономики – государственный бюджет и бюджет игрока – со своими подсистемами (налоги, приоритеты госзаказа, субсидии, займы, лимиты капитальных/операционных затрат)[[1]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B8%20%D0%BA%D0%B0%D1%80%D1%82%20%D0%B4%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D0%BE%20%D1%84%D1%80%D0%BE%D0%BD%D1%82%D1%83). Вводятся сущности контрактов (строительство, исследование, карантин) с жизненным циклом и штрафами за невыполнение[[1]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B8%20%D0%BA%D0%B0%D1%80%D1%82%20%D0%B4%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D0%BE%20%D1%84%D1%80%D0%BE%D0%BD%D1%82%D1%83). В Domain/Infection закладываются несколько слоев данных: уровень заражения, плотность населения, мобильность, санитарные кордоны, транспортные коридоры[[2]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,json%2Fbinary%29%20%2B%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F). В Application/Pathfinding выносится система навигации (графы на сетке, регионы, узкие места) для тактического и стратегического ИИ[[3]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%BB%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%8F). В AI/Perception – кэши влияния (influence maps) и «карты давления» по фронту, чтобы AI читал готовые поля при вычислении приоритетов[[3]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%BB%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%8F). В Infrastructure – абстракция для сохранений (JSON или бинарные) с версионированием состояния[[4]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,json%2Fbinary%29%20%2B%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F).

Важно: В рантайме используются иммутабельные объекты C# (“тёплые” модели) на основе данных из ScriptableObject-конфигов. Сами ScriptableObject служат только статическими каталогами (юнитов, зданий, технологий и т.д.), не модифицируются на лету; при запуске они копируются в обычные объекты-модели[[5]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%3A%20,Json%20%28%D0%B5%D1%81%D0%BB%D0%B8%20%D0%BD%D1%83%D0%B6%D0%B5%D0%BD%20%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9)[[6]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%2F%D1%83%D0%BD%D0%B8%D1%87%D1%82%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9%29%20%D0%B2%D1%81%D0%B5%D0%B3%D0%B4%D0%B0%20%D1%87%D0%B5%D1%80%D0%B5%D0%B7). Это предотвращает непредсказуемые изменения ассетов в Editor и обеспечивает сериализуемость моделей для сохранений.

## Цикл симуляции, пауза и фиксированный тик

Задача: сделать игровой цикл детерминированным и фиксированным по времени, независимым от FPS, с поддержкой паузы. Симуляция должна обновляться фиксированными шагами (например, 5–10 тиков в секунду), вне зависимости от просадок/скачков кадровой частоты[[7]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D0%BE%D1%81%D1%82%D0%B0%D0%BD%D0%B0%D0%B2%D0%BB%D0%B8%D0%B2%D0%B0%D0%B5%D1%82%20%D1%81%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8E%2C%20%D0%BD%D0%BE%20%D0%BD%D0%B5%20UI). При паузе останавливается только симуляция, тогда как UI и анимации интерфейса продолжают обновляться.

Реализация: создается собственный компонент SimulationLoop (MonoBehaviour) с интерфейсом времени IClock (например, содержит Delta и IsPaused). В Update() этот компонент накапливает время с помощью Time.unscaledDeltaTime и вызывает тик системы каждый раз, когда накоплен фиксированный интервал tick[[8]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82%D1%8B%3A%20,%D1%82%D0%B8%D0%BA%20%C2%AB%D0%B8%D0%B7%20%D0%BA%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B8%C2%BB)[[9]](file://file_00000000bdd462439d54689f98b935a3#:~:text=void%20Update%28%29%20,Tick%28tick). Например, при tick = 0.1 сек (10 тиков/сек) цикл выглядит так:

* Каждый кадр: если игра не на паузе, увеличить аккумулятор acc на unscaledDeltaTime[[8]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82%D1%8B%3A%20,%D1%82%D0%B8%D0%BA%20%C2%AB%D0%B8%D0%B7%20%D0%BA%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B8%C2%BB).
* Пока acc >= tick: вычитать из него tick и последовательно вызывать методы Tick() у всех подсистем: InfectionSystem, EconomySystem, StrategicAI, TacticalAI и т.д., затем в самом конце тик-цикла вызывать EventBus.Flush() для обработки накопленных событий[[10]](file://file_00000000bdd462439d54689f98b935a3#:~:text=while%20%28acc%20,Flush%28%29%3B). Таким образом соблюдается строгий порядок обновления и декомпозиция по подсистемам.
* Реализовать паузу: флаг IsPaused блокирует накопление времени – при паузе Delta = 0 и цикл тиков пропускается[[8]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82%D1%8B%3A%20,%D1%82%D0%B8%D0%BA%20%C2%AB%D0%B8%D0%B7%20%D0%BA%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B8%C2%BB). Метод SimulationLoop.SetPaused(bool) управляет этим флагом внешне (через UI или горячую клавишу)[[11]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20sealed%20class%20SimulationLoop%20%3A,IsPaused%20%3F%200f%20%3A%20tick)[[12]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,IsPaused%20%3D%20v%3B).
* Добавить поддержку изменения скорости: 0.5× / 1× / 2× / 4×. Проще всего вводить множитель времени, влияющий на длительность tick или на скорость накопления. Например, для ускорения 2× можно временно уменьшить tick вдвое, чтобы цикл шел чаще (UI останется на unscaled-времени и продолжит работать нормально)[[13]](file://file_0000000077706243ab98f6cd120afa5f#:~:text=%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D0%B8%20%C3%970,10). Альтернативно – увеличивать acc быстрее.
* Добавить режим «шаг за шагом»: отдельная кнопка или клавиша, которая при паузе выполняет ровно один тик симуляции. Это можно сделать либо вызвав цикл while(acc>=tick) один раз принудительно, либо напрямую вызвав Tick() всех систем по очереди и сразу остановившись. Желательно защититься через Assert, чтобы за один шаг действительно произошел только один тик[[14]](file://file_0000000077706243ab98f6cd120afa5f#:~:text=%D0%9E%D0%B4%D0%B8%D0%BD%D0%BE%D1%87%D0%BD%D1%8B%D0%B9%20%D1%82%D0%B8%D0%BA%20,11).

Варианты паузы: Рекомендуется контролировать паузу явным кодом (как описано выше). Использование Time.timeScale = 0 также возможно для глобальной паузы, но тогда необходимо убедиться, что все, что должно продолжать работать (UI, меню, анимации интерфейса), переведено на работу с реальным временем. Например, компонент Animator в UI следует переключить в режим Unscaled Time, а таймеры – использовать Time.unscaledDeltaTime или WaitForSecondsRealtime[[15]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%20%D0%B2%20Unity%20%EE%88%80cite%EE%88%82turn5search2%EE%88%82turn5search1%EE%88%81). Такой подход (UI на unscaled-времени при timeScale=0) считается каноническим паттерном в Unity[[16]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%EE%88%80cite%EE%88%82turn0search5%EE%88%81.%20,%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%20%D0%B2%20Unity%20%EE%88%80cite%EE%88%82turn5search2%EE%88%82turn5search1%EE%88%81).

💡 Совет по DOTS: Если проект перейдет на ECS, можно использовать встроенную группу FixedStepSimulationSystemGroup для фиксированного тика – она обеспечивает фиксированный шаг «из коробки» в ECS[[17]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%82%D0%B8%D0%BA%20%C2%AB%D0%B8%D0%B7%20%D0%BA%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B8%C2%BB).

Отладка цикла: Добавьте в UI простой оверлей с отображением текущего номера тика и коэффициента скорости. Это поможет проверить, что тики идут стабильно и не пропускаются (например, индикатор «Tick: X» обновляется каждую итерацию)[[18]](file://file_00000000a1386243b45d516491d7291b#:~:text=,tick). Логируйте начало и конец каждого тика в консоль (порядок вызова систем), чтобы убедиться, что последовательность всегда одна и та же[[19]](file://file_00000000a1386243b45d516491d7291b#:~:text=1,%D0%B1%D0%BE%D0%BB%D0%B5%D0%B5%20%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%88%D0%B0%D0%B3%D0%B0%20%D0%BF%D1%80%D0%B8%20Step%C2%BB). При резком изменении FPS (включить/выключить VSync) тиковая частота должна оставаться постоянной – это ключевой тест корректности фиксированного цикла[[20]](file://file_00000000a1386243b45d516491d7291b#:~:text=5,%EE%88%80cite%EE%88%82turn0search12%EE%88%81). В профайлере (Window → Analysis → Profiler) следите за ровным временем на тик в разделе Scripts, отсутствие всплесков Garbage Collection[[20]](file://file_00000000a1386243b45d516491d7291b#:~:text=5,%EE%88%80cite%EE%88%82turn0search12%EE%88%81).

## Данные, конфигурации и система сохранений

ScriptableObject → Runtime: Все игровые параметры и каталоги (юниты, здания, технологии, параметры инфекции и т.д.) хранятся в ScriptableObject-ассетах в папке Data. При старте игры они преобразуются в чистые C#-объекты (POCO-модели) внутри слоя Domain[[5]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%3A%20,Json%20%28%D0%B5%D1%81%D0%BB%D0%B8%20%D0%BD%D1%83%D0%B6%D0%B5%D0%BD%20%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9). Это означает, что во время игры ассеты не мутируют вообще. Любые изменения состояния происходят только в runtime-объектах, что обеспечивает безопасность и легкую сериализацию состояния игры. Например, описание технологии TechDef (SO) при запуске конвертируется в иммутабельный объект Tech с теми же параметрами[[21]](file://file_00000000bdd462439d54689f98b935a3#:~:text=ScriptableObject%20%E2%86%92%20Runtime,%D1%81%D1%82%D0%B0%D1%80%D1%82%D0%B5%20%D0%BF%D1%80%D0%B5%D0%B2%D1%80%D0%B0%D1%89%D0%B0%D0%B5%D0%BC%20%D0%B2%20%D1%87%D0%B8%D1%81%D1%82%D1%8B%D0%B5%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B)[[22]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%2F%2F%20Domain%2FTech%2FTech,string%3E%20Prereq%29%3B). Такой подход сочетает удобство настройки баланса через инспектор Unity и надежность чистых моделей (отсутствие side-эффектов).

JSON-сохранения: Для сохранения состояния используется сериализация в JSON. На начальном этапе достаточно встроенного JsonUtility, однако важно учитывать его ограничения: он не поддерживает типы вроде Dictionary и полиморфные поля (наследники) без дополнительных атрибутов[[23]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%28%D0%B7%D0%B5%D1%80%D0%BA%D0%B0%D0%BB%D0%B8%D1%82%D1%8C%20%D0%B2%D1%8B%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8). Поэтому схемы данных для сохранения планируются с учетом этих ограничений – например, вместо словарей применять списки пар ключ-значение или использовать простой собственный формат. Если проект потребует более гибкую сериализацию, можно перейти на System.Text.Json или сторонние библиотеки, но это увеличит сложность. Ранний этап включает простую реализацию сохранения/загрузки всего GameState в JSON и тест этого функционала (см. этапы разработки)[[24]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%2011%E2%80%9314%20,5%2F%C3%971%2F%C3%972%2C%20%D0%BE%D0%B2%D0%B5%D1%80%D0%BB%D0%B5%D0%B8%20%D0%B8%D0%BD%D1%84%D0%B5%D0%BA%D1%86%D0%B8%D0%B8%2F%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F).

Версионирование сейвов: В вводится система версий сохранений – например, поле SaveVersion в корне JSON-файла[[25]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B0%20runtime,%D0%BC%D0%B8%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%BF%D1%80%D0%B8%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B5). При загрузке игра читает версию и выполняет миграцию данных, если файл был сохранен на старой версии. Это важно для поддержки обратной совместимости по ходу развития проекта. Рекомендуется реализовать простую структуру: переключатель (switch) по версии, и в случае устаревшего формата – преобразовать данные (или сообщить об ошибке, если downgrade невозможен).

Addressables для ресурсов: Для управления игровыми ассетами (спрайты, звуки и прочее) планируется использовать Addressables. Это позволит подгружать ресурсы по запросу и выгружать их, уменьшая потребление памяти. Ключевой момент – дисциплина: на каждый Addressables.LoadAssetAsync должен приходиться соответствующий Addressables.Release, иначе будут утечки памяти[[26]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%EE%88%80cite%EE%88%82turn2search10%EE%88%81). В профилировщике памяти нужно отслеживать количество загруженных адресуемых объектов до и после сцен, а также вести логирование загрузок/выгрузок (например, через счетчик или debug-лог)[[27]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn1search10%EE%88%81).

Система сохранений и команда на уровне инфраструктуры: Подсистема Saving (в Application/Saving) предоставляет сервис сохранения игры. Сохранение выполняется только в безопасных точках – например, сразу после завершения тик-цикла, когда все события обработаны. Это предотвращает ситуацию, когда часть изменений применена, а часть – нет. Планируется внедрить буфер команд и событий: игра может логировать команды (например, строительство здания) и выполнять сохранение после полного применения команд в тик (барьер тика)[[28]](file://file_00000000bea06243b50b33d9516a800b#:~:text=7%29%20%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2C%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%2C%20%D1%81%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20,50%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BA%D0%B0%D1%85%C2%BB%20%E2%86%92%20%D0%BE%D0%B4%D0%B8%D0%BD%20%D0%B1%D0%B0%D1%82%D1%87)[[29]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D1%82%D1%81%D1%8F%20%D0%B2%20DOTS,%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8%20%D0%B8%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn2search16%EE%88%82turn2search10%EE%88%81).

Unit-тесты данных: Рекомендуется написать тесты, проверяющие модели данных и сериализацию. Например,Assert-тесты, что при сериализации/десериализации модели через JsonUtility все поля восстанавливаются правильно, включая подтипы (если используются SerializeReference)[[30]](file://file_00000000a1386243b45d516491d7291b#:~:text=1%29%20,%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%20%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%B0%D0%B2%D0%BB%D0%B8%D0%B2%D0%B0%D0%B5%D1%82%20%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D1%8B). Отдельно проверить, что ни один ScriptableObject не поменялся во время игры: например, в Editor-режиме можно Assert-ить, что AssetDatabase.Contains(obj) возвращает false для любого runtime-объекта (то есть объект не является ассетом)[[30]](file://file_00000000a1386243b45d516491d7291b#:~:text=1%29%20,%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%20%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%B0%D0%B2%D0%BB%D0%B8%D0%B2%D0%B0%D0%B5%D1%82%20%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D1%8B).

## Экономика: двойная модель бюджета и ресурсы

Экономическая система включает две параллельных экономики – государственную (ИИ) и личную (игрока). Это реализовано двумя объектами, реализующими интерфейс IEconomy (например, StateEconomy и PlayerEconomy), каждый со своим балансом, доходами и расходами[[31]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%B8%20%C2%AB%D1%82%D0%BE%D1%87%D0%B5%D1%87%D0%BD%D1%83%D1%8E%20%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D1%83%C2%BB%20%D0%98%D0%98).

* Госбюджет (StateEconomy): формируется из налогов и базовых ресурсов. Тратится автоматически ИИ-государством по стратегическим приоритетам – оборона, карантинные мероприятия, базовые исследования и логистика[[31]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%B8%20%C2%AB%D1%82%D0%BE%D1%87%D0%B5%D1%87%D0%BD%D1%83%D1%8E%20%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D1%83%C2%BB%20%D0%98%D0%98)[[32]](file://file_00000000bea06243b50b33d9516a800b#:~:text=4%29%20%D0%AD%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0%20%28%D0%B4%D0%B2%D0%BE%D0%B9%D0%BD%D0%BE%D0%B5%20%D0%92%D0%92%D0%9F%29%20,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8). ИИ перераспределяет бюджет между этими направлениями на основе Utility AI (см. раздел AI) – то есть автоматически оценивает, куда вложения сейчас важнее всего[[33]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8). Например, при вспышке инфекции приоритет сместится на оборону и карантин. Гос-ИИ может также субсидировать игрока (например, в виде “контрактов” или целевых грантов).
* Бюджет игрока (PlayerEconomy): это своего рода личный ВВП игрока[[31]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%B8%20%C2%AB%D1%82%D0%BE%D1%87%D0%B5%D1%87%D0%BD%D1%83%D1%8E%20%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D1%83%C2%BB%20%D0%98%D0%98). Игрок напрямую инвестирует эти средства в особые проекты – прорывные исследования, постройку фортификаций, поддержку операций ИИ. Игрок может брать краткосрочные займы под проценты, если не хватает средств[[33]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8). В геймдизайне это отражает волевые усилия игрока, которые выходят за рамки бюрократической машины государства.

Обе экономики взаимодействуют. В проект вводится механика контрактов с KPI: например, если гос-ИИ берет на себя защиту определенной линии фронта, но терпит неудачу (прорыв инфекции), то часть госбюджета выплачивается как компенсация (штраф) – фактически переходит в распоряжение игрока или теряется[[34]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8). Это стимулирует ИИ выполнять обязательства. Контракты (BuildOrder, ResearchOrder и т.д.) могут оформляться как отдельные объекты в системе, публикуемые через EventBus (пример команды строительства ниже) и отслеживающие свой статус.

Множественные ресурсы: Экономика усложняется введением четырех базовых ресурсов вместо абстрактных денег[[35]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%C2%A79). Предусмотрены: трудовые часы населения (LaborHours) – универсальный производственный ресурс, энергия (Power) – генерация/потребление электричества в день, материалы (Materials) – стройматериалы, детали, боекомплект (совмещенный ресурс), еда (Food) – снабжение населения и войск[[35]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%C2%A79). Бюджет игрока и государства оперируют этими ресурсами (в различных пропорциях).

Экономический цикл работает с суточным тиком – раз в игровой день происходит расчет экономики по фиксированным этапам[[36]](file://file_0000000004006243943cbb9888dd4b65#:~:text=9%29%20%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%82%D0%B8%D0%BA%D0%B0%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F). Порядок суточного тика: 1. Рост населения: рассчитывается прирост мирного населения за день (CivGrowth) на основе базового процента и текущих условий[[36]](file://file_0000000004006243943cbb9888dd4b65#:~:text=9%29%20%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%82%D0%B8%D0%BA%D0%B0%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F). Часть этого прироста может быть автоматически передана под управление ИИ (см. ниже). 2. Передача процентов ИИ: от общего суточного прироста и добытых ресурсов вычисляется доля, которая сразу уходит ИИ. Для каждого ресурса (включая прирост населения) на UI предусмотрен слайдер 0–100% – сколько % игрок автоматически отдаёт гос-ИИ[[37]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,8)[[36]](file://file_0000000004006243943cbb9888dd4b65#:~:text=9%29%20%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%82%D0%B8%D0%BA%D0%B0%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F). Например, если игрок считает, что ИИ лучше управится с энергией, он может перенаправить 50% произведенной энергии в госбюджет каждый день. Передача вычисляется до поступления ресурсов на склады игрока, с округлением вниз до целых единиц[[38]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1,%28%D0%BF%D1%80%D0%B8%20%D0%BD%D0%B0%D0%BB%D0%B8%D1%87%D0%B8%D0%B8%20%D0%BD%D0%B5%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%B8%D1%85%20%D0%98%D0%98). 3. Базовое потребление: автоматически списывается еда на содержание населения, еда и материалы на содержание армии, энергия на работу построек, потери при хранении и прочие постоянные расходы[[39]](file://file_0000000004006243943cbb9888dd4b65#:~:text=2.%20%2A%2A%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0%20,%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%2C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B0%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B5%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8%2C%20%D0%B7%D0%B0%D0%B1%D0%B8%D1%82%D1%8B%D0%B5%20%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B). 4. Производство зданий: генерируются ресурсы экономическими постройками (фермы производят еду, электростанции – энергию, шахты – материалы и т.д.), конвертируются трудовые часы населения в реальные товары[[39]](file://file_0000000004006243943cbb9888dd4b65#:~:text=2.%20%2A%2A%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0%20,%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%2C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B0%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B5%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8%2C%20%D0%B7%D0%B0%D0%B1%D0%B8%D1%82%D1%8B%D0%B5%20%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B). 5. Очереди производства и обучения: обрабатываются очереди строительства зданий и подготовки военных юнитов. Например, казармы в день могут призвать ограниченное число новобранцев, превращая часть свободных гражданских (LaborHours) в военных за несколько дней обучения[[40]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%D0%B2%20%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%2F%D0%B3%D0%B0%D1%80%D0%BD%D0%B8%D0%B7%D0%BE%D0%BD%20%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%20%D0%B7%D0%B0%D0%B2%D0%B5%D1%80%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)[[41]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,). Для зданий – тратятся материалы/труд на прогресс строительства. В этом шаге уменьшаются запасы ресурсов согласно заявкам в очередях и увеличивается процент завершения проектов. 6. Логистика: обновляются статусы всех активных транспортных приказов (см. раздел логистики). Если какие-то подразделения или ресурсы должны были прибыть в этот день, они перемещаются в конечную точку; вычисляется время до прибытия оставшихся конвоев, проверяется пропускная способность коридоров[[42]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%29.%204.%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F). 7. Фортификации: аналогично, продвигается прогресс возведения или ремонта фортификационных сооружений, учитывается износ временных укреплений со временем[[43]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%29.%204.%20,%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%2C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B0%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B5%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8%2C%20%D0%B7%D0%B0%D0%B1%D0%B8%D1%82%D1%8B%D0%B5%20%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B). 8. Отчет и события: по итогам дня формируется сводный отчет (для экрана экономики) и генерируются события/оповещения: обнаружение дефицита ресурса, простаивание мощностей, перегруженность логистических узлов и т.п.[[43]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%29.%204.%20,%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%2C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B0%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B5%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8%2C%20%D0%B7%D0%B0%D0%B1%D0%B8%D1%82%D1%8B%D0%B5%20%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B).

Правило баланса: система не допускает отрицательных значений ресурсов – если расход превышает доступный запас, включаются механики понижения потребления и распределения дефицита[[44]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%2A%2A%D1%88%D1%82%D0%B0%D1%82%D0%BD%D1%83%D1%8E%20%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B2%20%D1%82%D1%80%D1%83%D0%B4%D0%B0,%28%D1%81%D0%BC.%20%C2%A710). Например, при нехватке энергии отключаются объекты по приоритетам (сначала наименее важные)[[45]](file://file_0000000004006243943cbb9888dd4b65#:~:text=10%29%20%D0%94%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%20%D0%B8%20%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D1%8B%20,%D0%B2%D1%81%D1%91%20%D0%B2%D0%B8%D0%B4%D0%BD%D0%BE%20%D0%B2%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%BC%20%D0%BE%D1%82%D1%87%D1%91%D1%82%D0%B5), при нехватке еды – прирост населения падает до нуля, растет риск негативных событий (бунты, дезертирство), а при длительном голоде население начинает убывать[[46]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D0%B5%20%C2%AB%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D0%BE%20%D0%B2%D0%B0%D0%B6%D0%BD%D0%BE%D0%B3%D0%BE%C2%BB%2C%203,%D0%B2%D1%81%D1%91%20%D0%B2%D0%B8%D0%B4%D0%BD%D0%BE%20%D0%B2%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%BC%20%D0%BE%D1%82%D1%87%D1%91%D1%82%D0%B5). Все автоматические сокращения происходят прозрачно: игрок видит, какие статьи урезаны, в суточном отчете не должно быть «скрытых» списаний[[47]](file://file_0000000004006243943cbb9888dd4b65#:~:text=2.%20%D0%9F%D1%80%D0%B8%20%D0%BD%D0%B5%D1%85%D0%B2%D0%B0%D1%82%D0%BA%D0%B5%20,%D0%B2%D1%81%D1%91%20%D0%B2%D0%B8%D0%B4%D0%BD%D0%BE%20%D0%B2%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%BC%20%D0%BE%D1%82%D1%87%D1%91%D1%82%D0%B5).

Типы зданий и юнитов: Экономические постройки делятся на категории: экономические здания (фермы, электростанции, шахты, склады) – увеличивают производство или хранение ресурсов; исследовательские (лаборатории, университеты) – позволяют открывать улучшения и повышают эффективность производства; военные (производственные) – казармы, заводы техники, аэродромы, которые производят боевые единицы и требуют ресурсы на содержание[[48]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D1%87%D0%B0%D1%81%D1%8B%20%D0%B8%20%D0%B5%D0%B4%D1%83%20%D0%BD%D0%B0%20%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B8). У каждой постройки задана стоимость строительства, время постройки (в днях), потребление/производство ресурсов в работе и на обслуживание[[49]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D1%8C%20%D0%BD%D0%B0%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC). Военные здания также имеют параметр конверсии: сколько гражданских можно обучить в день, длительность подготовки одного солдата и стоимость на одного рекрута в еде, материалах, энергии и труде[[41]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,).

UI для экономики: Требуется реализовать экран суточного отчета экономики – на нем отображаются балансы ресурсов, прирост/убыль за день, выявленные узкие места и т.д.[[50]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87). На отдельной панели – слайдеры передачи % ИИ для каждого ресурса (5 штук: труд, энергия, материалы, еда, прирост населения) с кнопкой сброса на 0%[[50]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87). Это дает игроку наглядный контроль над тем, сколько ресурсов он отдает под управление ИИ ежедневно.

💡 Оптимизация экономики: Расчеты экономики можно проводить разреженно – нет нужды обновлять бюджеты каждый кадр. Достаточно раз в N тиков (например, раз в игровые сутки) применять формулы, а в остальные тики держать показатели неизменными. Также все агрегаты (общий доход, расход) стоит кешировать и обновлять инкрементно при изменениях, а не вычислять с нуля каждый раз[[51]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%BE%D0%B4%D1%85%D0%BE%D0%B4). Это позволит сэкономить на расчетах, особенно при большом числе объектов.

Unit-тесты экономики: Написать простые тесты-инварианты для бюджета: например, после серии операций TryReserve и Commit сумма денег не исчезает и не создается из воздуха (балансы сходятся)[[52]](file://file_00000000a1386243b45d516491d7291b#:~:text=%2A%2A%D0%9F%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%BA%D0%B8%2F%D1%82%D0%B5%D1%81%D1%82%D1%8B%3A%2A%2A%201%29%20Unit,%C2%AB%D0%B4%D0%B5%D0%BD%D1%8C%D0%B3%D0%B8%20%D0%BD%D0%B5%20%D0%B8%D1%81%D1%87%D0%B5%D0%B7%D0%B0%D1%8E%D1%82%2F%D0%BD%D0%B5%20%D0%BF%D0%BE%D1%8F%D0%B2%D0%BB%D1%8F%D1%8E%D1%82%D1%81%D1%8F%C2%BB). Смоделировать 100 тиков с хаотичными расходами и проверять, что итоговые балансы соответствуют ожидаемым (ни один не стал отрицательным и т.п.)[[53]](file://file_00000000a1386243b45d516491d7291b#:~:text=1%29%20Unit,%C2%AB%D0%B4%D0%B5%D0%BD%D1%8C%D0%B3%D0%B8%20%D0%BD%D0%B5%20%D0%B8%D1%81%D1%87%D0%B5%D0%B7%D0%B0%D1%8E%D1%82%2F%D0%BD%D0%B5%20%D0%BF%D0%BE%D1%8F%D0%B2%D0%BB%D1%8F%D1%8E%D1%82%D1%81%D1%8F%C2%BB). Логировать отказы в расходах (например, ReserveFail с указанием суммы, баланса, источника) – это поможет выявить проблемы настроек[[54]](file://file_00000000a1386243b45d516491d7291b#:~:text=2,%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D1%8C%2C%20%D1%87%D1%82%D0%BE%20%D0%BD%D0%B5%D1%82%20%D0%B4%D1%80%D0%BE%D0%B6%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%B2).

## Модель заражения и инфекции (SIR/SEIR)

Основы модели: В игре реализуется гибрид эпидемиологической модели типа SIR/SEIR и клеточного автомата. Каждая клетка на карте имеет состояние заражения: Susceptible (восприимчивая), Infected (зараженная) или Recovered (выздоровевшая и иммунная); либо ведется непрерывный счет уровня инфекции 0..1[[55]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9E%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F%20%D1%82%D0%BE%D1%87%D0%BA%D0%B0%3A%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%20%D0%BD%D0%B0,%D0%B4%D0%BE%D0%BB%D1%8F%20%D0%BD%D0%B0%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%2F%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Распространение болезни рассчитывается по соседям на сетке (соседство может быть выбранным Moore 8 направлений или Von Neumann 4 направления – важно зафиксировать и использовать единообразно)[[56]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%A7%D1%82%D0%BE%20%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%B8%D1%82%D1%8C%20%28%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%BA%D0%B8%29%3A%2A%2A%20,%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2F%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%BD%D0%BE%20%D1%80%D0%B5%D0%B6%D1%83%D1%82%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE). Вероятность заражения клетки зависит от доли зараженных соседей, плотности населения в клетке, мобильности населения и защитных факторов.

Расчет тика инфекции: Система InfectionSystem каждую итерацию симуляции делает несколько шагов[[57]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%2F%2F%20Application%2FInfectionSystem%20public%20sealed%20class,%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%D1%85%20%7D%20%7D): (1) Диффузия вируса по соседям: для каждой клетки вычисляется влияние зараженных соседей и изменяется уровень инфекции/статус (S→I с некоторой вероятностью)[[57]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%2F%2F%20Application%2FInfectionSystem%20public%20sealed%20class,%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%D1%85%20%7D%20%7D). (2) Учет укреплений и карантинов: если в клетке или на границах есть защитные постройки (стена, блокпост) или зона карантина, они понижают эффективность распространения (например, снижают вероятность передачи или скорость роста)[[56]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%A7%D1%82%D0%BE%20%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%B8%D1%82%D1%8C%20%28%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%BA%D0%B8%29%3A%2A%2A%20,%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2F%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%BD%D0%BE%20%D1%80%D0%B5%D0%B6%D1%83%D1%82%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE). (3) Лечение/иммунитет: зараженные могут переходить в состояние R (выздоровевшие) с некоторой скоростью, которая может повышаться, если на клетку направлены ресурсы (например, медпомощь). Таким образом реализуется S→I→R динамика. Параметры (базовая инфекционность, длительность болезни, процент иммунитета и т.д.) настраиваются в конфиге InfectionParams.

Дополнительные механики инфекции: - Очаги и фронт инфекции: Если уровень заражения в группе соседних клеток превышает порог, считается, что образовался очаг. Набор граничных клеток между зараженными и чистыми зонами образует фронт заражения, который гос-ИИ будет стремиться удерживать[[58]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%29%20%EE%88%80cite%EE%88%82turn4search0%EE%88%81.%20,%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2F%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%BD%D0%BE%20%D1%80%D0%B5%D0%B6%D1%83%D1%82%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE). В системе вводится понятие frontier (граница распространения). - Карантинные зоны: Игрок или ИИ могут вводить карантин в определенных регионах, уменьшая там мобильность населения и вероятность передачи. Цена – падение производства в этих клетках (меньше рабочих рук, логистические перебои)[[58]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%29%20%EE%88%80cite%EE%88%82turn4search0%EE%88%81.%20,%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2F%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%BD%D0%BE%20%D1%80%D0%B5%D0%B6%D1%83%D1%82%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE). - Транспорт и распространение: Болезнь может распространяться не только локально, но и через транспортные коридоры (дороги, авиа, железную дорогу). Поэтому инфекция может «перепрыгивать» на большие расстояния, особенно если логистика не контролируется. Это отражается дополнительными событиями (например, занесение инфекции в удаленный город через аэропорт).

Алгоритмические улучшения: Чтобы обновление сетки не деградировало при большом размере, внедряется обновление только активных клеток. То есть держится список «грязных» клеток, где произошли изменения или рядом есть инфекция, и на каждом тик-цикле пересчитываются только они[[59]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BE%D0%B1%D1%89%D0%B5%D0%B5%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5). Также реализуется двойной буфер: сначала рассчитываются новые значения в буфере (gridB), читая из старого (gridA), затем буферы меняются местами – это гарантирует, что внутри одного тика изменения не влияют на соседей до следующего шага[[60]](file://file_00000000a1386243b45d516491d7291b#:~:text=%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9%20%D0%B2%D1%8B%D0%B1%D1%80%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BA%D0%B8%20%28gizmos%29,%D0%BA%D0%B0%D1%80%D1%82%D1%8B%3A%20%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%B0%D0%B5%D1%82%20%C2%AB%D0%B3%D0%BB%D0%B0%D0%B7%D0%BE%D0%BC%C2%BB%20%D0%BB%D0%BE%D0%B2%D0%B8%D1%82%D1%8C%20%D1%81%D0%BA%D0%B0%D1%87%D0%BA%D0%B8%2F%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D0%B8). Этот прием предотвращает искусственное ускорение распространения из-за последовательного обновления.

Оптимизация производительности инфекции: В будущем, когда число клеток велико, планируется вынести расчет диффузии инфекции в параллельные Job’ы с Burst-компиляцией. Например, можно реализовать IJobParallelFor, разбивающий сетку на чанки, и хранить поля заражения в NativeArray<float>[[59]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BE%D0%B1%D1%89%D0%B5%D0%B5%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5). Burst даст значительный выигрыш, особенно если убрать из цикла лишние выделения. При миграции на ECS инфекция может храниться как компонент на сущности-клетке; расчет выполняться системой в FixedStepSimulationSystemGroup (фиксированный шаг) – тогда даже Time.deltaTime не нужно вручную накапливать[[61]](file://file_00000000bea06243b50b33d9516a800b#:~:text=Jobs%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20,%D0%BE%D0%B1%D1%89%D0%B5%D0%B5%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5). Такой подход использует преимущества DOTS: данные в плотных массивах, минимизация расходов на управление.

Unit-тесты инфекции: Проверить корректность базовой модели SIR: с заданными параметрами через N тиков доля S уменьшается, I растет и затем падает, R растет до стабилизации – то есть модель ведет себя правдоподобно[[62]](file://file_00000000a1386243b45d516491d7291b#:~:text=%D0%98%D0%BD%D1%84%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%20). Убедиться, что выбранное соседство (Moore или Von Neumann) соблюдается – можно сделать gizmos-отображение соседей выбранной клетки для визуальной проверки на сцене[[63]](file://file_00000000a1386243b45d516491d7291b#:~:text=%EE%88%80cite%EE%88%82turn4search0%EE%88%81%202,%D0%BA%D0%B0%D1%80%D1%82%D1%8B%3A%20%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%B0%D0%B5%D1%82%20%C2%AB%D0%B3%D0%BB%D0%B0%D0%B7%D0%BE%D0%BC%C2%BB%20%D0%BB%D0%BE%D0%B2%D0%B8%D1%82%D1%8C%20%D1%81%D0%BA%D0%B0%D1%87%D0%BA%D0%B8%2F%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D0%B8). Протестировать механизм двойного буфера: в симуляции на 1 тик убедиться, что обновление клетки не влияет повторно на соседей в тот же тик. Также важно логировать размер frontier (количество активных клеток) и время шага, чтобы видеть эффективность оптимизации при разных масштабах[[64]](file://file_00000000a1386243b45d516491d7291b#:~:text=%E2%80%94%20%D1%82%D0%B5%D1%81%D1%82%20%C2%AB%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BD%D0%B5%20%D0%B7%D0%B0%D1%82%D0%B8%D1%80%D0%B0%D0%B5%D1%82%D1%81%D1%8F,%D0%BA%D0%B0%D1%80%D1%82%D1%8B%3A%20%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%B0%D0%B5%D1%82%20%C2%AB%D0%B3%D0%BB%D0%B0%D0%B7%D0%BE%D0%BC%C2%BB%20%D0%BB%D0%BE%D0%B2%D0%B8%D1%82%D1%8C%20%D1%81%D0%BA%D0%B0%D1%87%D0%BA%D0%B8%2F%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D0%B8).

## Искусственный интеллект: стратегический и тактический (Utility AI, GOAP, BT)

Подход: ИИ реализуется гибридно, сочетая три уровня: Utility AI для оценки приоритетов, GOAP для планирования действий и Behavior Trees для тактического исполнения[[65]](file://file_00000000bdd462439d54689f98b935a3#:~:text=1,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B). Такая комбинация позволяет ИИ реагировать на ситуацию осмысленно на разных масштабах.

* Utility AI (утилитарный ИИ): быстрый слой оценки того, что сейчас наиболее важно. Реализуется набором функций-рассуждений (Considerations), каждая из которых возвращает рейтинг 0..1 для определенной цели[[65]](file://file_00000000bdd462439d54689f98b935a3#:~:text=1,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B). Например, одна функция оценивает срочность сдерживания очага инфекции – чем больше заражение на границе и меньше доступных ресурсов, тем выше приоритет удержать линию[[66]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20sealed%20class%20HoldLineConsideration%20%3A,). Другая функция – важность закрыть узкое место (chokepoint) фортификацией, третья – вложиться в конкретное исследование вакцины, четвертая – поддержать действия игрока в горячей точке. Все оценки вычисляются на основе текущего мира (WorldState), который предоставляет необходимые метрики: уровень инфекции в точке, силы своих войск в регионе, наличие укреплений, узловые позиции и т.д.[[67]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%A1%D0%BB%D0%BE%D0%B9%20%D0%B2%D0%BE%D1%81%D0%BF%D1%80%D0%B8%D1%8F%D1%82%D0%B8%D1%8F%3A%20,GridPos%20p). Пример: Consideration HoldLine возвращает высокий score, если на клетке-форпосте близок прорыв инфекции[[68]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%60%60%60csharp%20public%20interface%20IConsideration%20,). Utility AI регулярно (каждый N тиков) пересчитывает эти оценки и выбирает наиболее приоритетную цель для ИИ[[69]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B8%20%D0%B1%D0%BE%D0%B3%D0%B0%D1%82%D0%B0%20%D0%BD%D0%B0%20%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn1search1%EE%88%81). Этот подход очень легковесен, объясним (можно вывести, почему ИИ выбрал именно эту цель) и хорошо подходит для верхнеуровневого управления.
* GOAP (Goal-Oriented Action Planning): когда цель выбрана, стратегический ИИ строит план из последовательности действий для ее достижения[[65]](file://file_00000000bdd462439d54689f98b935a3#:~:text=1,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B). Планировщик (Planner) берет набор возможных действий (Action) с пред- и постусловиями и ищет цепочку, ведущую к цели, похожим образом на планировщик STRIPS. Применение GOAP – на стратегическом уровне (“государственном” ИИ): например, цель “локализовать вспышку в регионе X” может выразиться планом: собрать ресурсы → исследовать технологию «Щит-I» (улучшение вакцины) → построить 3 турели на дуге вокруг региона X[[65]](file://file_00000000bdd462439d54689f98b935a3#:~:text=1,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B). Каждое действие (IAction) имеет проверку Preconditions(IWorldState) и эффект Apply()[[70]](file://file_00000000bdd462439d54689f98b935a3#:~:text=GOAP%20%28%D1%8D%D1%81%D0%BA%D0%B8%D0%B7%29%3A%20,IAction%3E%20Plan%28Goal%20g%2C%20IWorldState%20w). Классический референс этой системы – AI игры F.E.A.R. (Jeff Orkin)[[71]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%C2%AB%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20X%C2%BB%2C%20%C2%AB%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D1%82%D1%8C%20%D0%B8%D0%B3%D1%80%D0%BE%D0%BA%D0%B0%C2%BB%20%E2%80%94,%D0%B8%20%D0%B1%D0%BE%D0%B3%D0%B0%D1%82%D0%B0%20%D0%BD%D0%B0%20%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn1search1%EE%88%81), где GOAP впервые успешно применили для динамического планирования поведения врагов. Для нашего ИИ планировщик будет ограничен несколькими типами действий: строительство определенного здания, начало исследования, переброска войск, введение карантина и т.д. – этого достаточно, чтобы формировать осмысленные последовательности. GOAP-план обновляется реже, чем Utility (например, при смене цели или раз в несколько секунд). Ограничиваем глубину и время поиска плана, чтобы не было просадки производительности, и кешируем уже найденные краткие планы (если ситуация мало изменилась)[[72]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D1%8F%D1%82%D1%8C%20%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%BF%D1%80%D0%B8%20%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D1%8B).
* Behavior Trees (поведенческие деревья): на тактическом уровне (микро-ИИ отдельных отрядов) используется древовидная логика поведения[[73]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%E2%86%92%20%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%C2%AB%D0%A9%D0%B8%D1%82,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B). BT хорошо подходят для реакций в реальном времени: патрулирования, атаки, отступления, ремонта объектов, локальной обороны и т.п. Например, дерево для солдат может быть: Selector( если есть угроза рядом → занять оборону, иначе если есть повреждения в районе → начать ремонт, иначе → патрулировать )[[74]](file://file_00000000bdd462439d54689f98b935a3#:~:text=BT%20%28%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BA%D0%BE%D0%B4%20%D1%83%D0%B7%D0%BB%D0%BE%D0%B2%29%3A%20,Patrol%20%29). Behavior Tree выполняется каждый тик (для активных юнитов) и реагирует на события (враги в зоне, получен урон, приказ). Мы стараемся, чтобы BT не занимались выбором глобальных целей – пусть этим ведает Utility/GOAP – а реализовывали именно последовательность действий для текущей задачи[[75]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81). BT легко отлаживать и визуально представлять, поэтому они будут основой тактического ИИ в бою.

Influence Maps и восприятие: В компоненте AI/Perception будут формироваться специальные карты влияния. Influence Map – это двумерная матрица (grid), на которой числом выражено влияние той или иной силы на каждую клетку[[76]](https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai#:~:text=Something%20that%20I%20just%20thought,called%20the%20influence). Классический пример: влияние войск – каждая своя единица добавляет +X влияния на клетки рядом, а враг – отрицательное влияние. Получается тепловая карта: области с высоким положительным значением контролируются игроком, с отрицательным – врагом. Такие карты позволяют AI быстро оценивать обстановку: например, видя распределение влияния, стратегический AI может определять направления, где оборона слаба или где враг сосредоточен[[77]](https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai#:~:text=So%20for%20example%2C%20if%20you,the%20points%20of%20highest%20influence). В нашем проекте будут как минимум: карта заражения (где высок уровень инфекции – как “вражеское” влияние), карта дружеских сил (концентрация своих войск), карта укреплений/контроля. Utility AI как раз читает эти карты при расчете – вместо того, чтобы каждый раз сканировать все объекты, он получает заранее посчитанные метрики[[3]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%BB%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%8F). Influence Maps легко объяснимы геймдизайну и богаты на эвристики[[78]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%EE%88%80cite%EE%88%82turn1search0%EE%88%81.%20,%D0%B8%20%D0%B1%D0%BE%D0%B3%D0%B0%D1%82%D0%B0%20%D0%BD%D0%B0%20%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn1search1%EE%88%81). Их обновление происходит инкрементально: при каждом значимом изменении (перемещение войск, постройка, падение города) пересчитывается влияния в окрестностях, а не вся карта целиком[[79]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D1%8F%D1%82%D1%8C%20%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%BF%D1%80%D0%B8%20%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D1%8B). В тяжелых случаях расчет тоже выносится в Job’ы.

Пример влияния: ИИ может иметь Consideration: «PressureToRetreat» – оценивает, насколько велика в данной точке сила врага относительно своих. Это можно вычислить, взяв значение influence map врага минус influence map своих на клетке, нормировать к [0..1]. Таким образом Utility AI получит высокий скор для отступления, если враг доминирует[[76]](https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai#:~:text=Something%20that%20I%20just%20thought,called%20the%20influence)[[77]](https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai#:~:text=So%20for%20example%2C%20if%20you,the%20points%20of%20highest%20influence).

Частота обновления ИИ: В целях производительности стратегический AI будет работать на пониженной частоте. Например, Utility-оценки – раз в 0.5–1 секунду, а тактические BT – каждый тик[[80]](file://file_00000000bea06243b50b33d9516a800b#:~:text=2,%D1%86%D0%B8%D0%BA%D0%BB%D1%83)[[81]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D1%8F%D1%82%D1%8C%20%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%BF%D1%80%D0%B8%20%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D1%8B). Также тяжелые вычисления (например, много условий в Utility) можно разбивать по кадрам – считать по одной группе целей за тик, чтоб не все сразу (budgeted AI). Планировщик GOAP ограничивается, чтобы не планировать новый путь каждый кадр – достаточно перепланировать, когда изменились входные условия или раз в несколько секунд.

Отладка AI: Для понимания работы AI добавим логирование его решений. Каждые N тиков (скажем, раз в секунду) в Dev-режиме выводить таблицу Utility-скоров по целям: это покажет, какие приоритеты видит ИИ[[82]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5). При построении GOAP-плана логировать шаги: например, [GOAP] Plan: Gather → Research[Shield-I] → Build[Turret×3][[83]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search2%EE%88%81). Для Behavior Tree можно логировать активные ветки и статус листьев (Success/Fail/Running) каждый тик[[75]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81). Кроме того, визуализируем планы: например, отображать на карте текущую цель стратегического ИИ, или рисовать линиями путь, который планировщик намерен пройти (последовательность действий), а для BT – подсвечивать текущую экшен-ноду. Influence Maps тоже можно визуально отладить – через оверлей-тепловую карту или изолинии на карте, показывающие границы контроля[[82]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5). Это значительно упростит настройку и поиск ошибок в решениях AI.

## События, команды и система строительства

В архитектуре предусмотрен паттерн EventBus для обмена сообщениями между подсистемами. Он обеспечивает асинхронную рассылку событий через очередь, чтобы ослабить связь между, например, симуляцией и UI[[84]](file://file_00000000bdd462439d54689f98b935a3#:~:text=EventBus%20,%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%2C%20%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B1%D1%8E%D0%B4%D0%B6%D0%B5%D1%82%D0%B0). Любая система может опубликовать событие (EventBus.Publish(new SomethingHappened(...))), а заинтересованные слушатели, подписанные через Subscribe, получат его при следующем фрейме, когда EventBus выполнит Flush()[[85]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20static%20class%20EventBus%20,On%28%28dynamic%29e%29%3B). Благодаря этому, игровые события (построено здание, начато исследование, вспышка инфекции) обрабатываются поштучно в конце тика, не вызывая немедленных побочных эффектов.

Команды (Commands): Это паттерн для действий, которые можно логировать, отменять или воспроизводить. В игре команды используются для ключевых изменений состояния: постройка здания, приказ юниту, распределение бюджета[[84]](file://file_00000000bdd462439d54689f98b935a3#:~:text=EventBus%20,%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%2C%20%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B1%D1%8E%D0%B4%D0%B6%D0%B5%D1%82%D0%B0)[[86]](file://file_00000000bdd462439d54689f98b935a3#:~:text=Commands%20%E2%80%94%20%D1%83%D0%B4%D0%BE%D0%B1%D0%BD%D0%BE%20%D0%BB%D0%BE%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%2F%D0%BE%D1%82%D0%BA%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%82%D1%8C%20%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%8F,%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%2C%20%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B1%D1%8E%D0%B4%D0%B6%D0%B5%D1%82%D0%B0). Каждая команда – иммутабельный объект (например, PlaceBuildingCmd с параметрами типа здания, координат, инициатора). Команда помещается в EventBus как событие и потребляется соответствующим сервисом, который ее исполняет. Такой подход упрощает отладку – можно хранить историю команд за последние X тиков и при необходимости откатывать их. В сохранении игры можно фиксировать список команд от прошлого сохранения как лог изменений.

Пример – строительство здания: Игрок кликает на карте для строительства. Создается команда PlaceBuildingCmd(BuildingType type, GridPos at, Guid who)[[87]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0%3A%20,Publish%28order), где who – ID инициатора (игрок или ИИ). Сервис строительства (BuildService) проверяет через CanPlace возможность строительства (достаточно ли средств на переданном бюджете IEconomy, свободна ли клетка)[[87]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0%3A%20,Publish%28order). Если нельзя – возвращает BuildOrder.Failed. Если можно – списывает средства (money.Commit(cost)), создает объект заказа строительства BuildOrder и публикует его как событие через EventBus[[88]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20BuildOrder%20Place,Publish%28order%29%3B%20return%20order%3B). Сам физический процесс строительства будет обработан в системе строительства (например, добавит объект здания с прогрессом постройки). Этот механизм позволяет отделить инициацию действия (команду) от выполнения (событийное реагирование), а также легко логировать все постройки (для аналитики или сетевого реплея, если понадобится).

Coalescing событий: Если за один тик накапливается много однотипных событий, их можно “склеивать”. Например, вместо 50 отдельных событий роста инфекции по каждой клетке можно выпустить одно событие “инфекция выросла в 50 клетках” с соответствующими данными[[89]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,50%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BA%D0%B0%D1%85%C2%BB%20%E2%86%92%20%D0%BE%D0%B4%D0%B8%D0%BD%20%D0%B1%D0%B0%D1%82%D1%87). Это снизит нагрузку на обработчики и освободит лог от избыточных записей. Реализуется на уровне EventBus или в менеджере событий: перед Flush проверять очередь на батчи однотипных сообщений.

Сохранение и события: Важно сохранить правильный порядок – сохранение игры выполняется после завершения обработки команд и событий текущего тика. Для этого можно установить правило: сохранение можно инициировать (через команду SaveGame) только во время паузы или в конце тика, когда игровой цикл не в середине вычислений. В ECS-реализации то же правило обеспечивается использованием EntityCommandBuffer – структурные изменения (создание/удаление Entity) происходят одним пакетом в конце системы, что эквивалентно flush-событиям[[90]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%2F%D1%84%D0%BB%D0%B0%D1%88%20%D0%BF%D0%BE%20%D0%BA%D0%BE%D0%BD%D1%86%D1%83%20%D1%82%D0%B8%D0%BA%D0%B0).

Отладка событий/команд: Проверить, что EventBus.Flush() всегда вызывается после тиков систем – иначе могут быть рассинхроны (например, UI показывает событие раньше, чем обновилось состояние)[[91]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn1search6%EE%88%81). Логировать сам момент флаша: например, выводить [EventBus] Flushed (count=X) – сколько событий обработано за тик. Убедиться, что события действительно очищаются (очередь пуста) к началу следующего цикла. Для команд с откатом можно написать тест: выполнить команду, затем “откатить” (например, не списывать деньги) и проверить инварианты (балансы те же и т.п.). В многопоточной DOTS-окружении – следить, чтобы никакие EntityManager.AddComponent и т.п. не вызывались вне ECB, иначе будут непредвиденные sync point’ы[[90]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%2F%D1%84%D0%BB%D0%B0%D1%88%20%D0%BF%D0%BE%20%D0%BA%D0%BE%D0%BD%D1%86%D1%83%20%D1%82%D0%B8%D0%BA%D0%B0).

## Логистика и транспортные коридоры

В игре реализуется непрямое управление логистикой. Игрок не таскает юниты вручную, а отдает приказы на переброску войск или ресурсов между узлами, а система логистики автоматически доставляет их, учитывая ограничения.

Логистические узлы и коридоры: На карте выделяются специальные узловые точки: города, склады, станции, порты, аэродромы и т.п. – места, где могут накапливаться войска и ресурсы[[92]](file://file_0000000004006243943cbb9888dd4b65#:~:text=5,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2). Эти узлы соединены коридорами (дороги, железные дороги, авиамаршруты). У каждого коридора есть параметр пропускной способности (например, N единиц в день) и базовое время в пути по нему (например, 6 часов)[[93]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%81%D1%87%D0%B8%D1%82%D0%B0%D0%B5%D1%82%20,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2)[[94]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,6).

Приказы переброски: Игрок через интерфейс логистики создает заказ: откуда, куда, что перебросить и с каким приоритетом[[95]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2). Например: “Перебросить 10 солдат из города A на фронт N, высокая скорость”. После этого автоматизация: система логистики находит маршрут (возможно, несколько коридоров через промежуточные узлы), ставит юниты в очередь и начинает их транспорт. Юниты двигаются автоматически – игрок их непосредственно не контролирует во время перевозки[[93]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%81%D1%87%D0%B8%D1%82%D0%B0%D0%B5%D1%82%20,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2).

Детали реализации: В каждый момент времени у коридора есть занятость/очередь. Алгоритм находит кратчайший по времени маршрут с учетом ёмкости (capacity) коридоров[[96]](file://file_0000000004006243943cbb9888dd4b65#:~:text=11%29%20%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%3A%20%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8%20%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20,%D0%B0%D1%80%D0%BC%D0%B8%D0%B9%3B%20%D0%BF%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%B4%D1%91%D1%82%20%D0%BF%D0%BE%20%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E). Если коридор перегружен, заказы ждут. Можно вводить режим Rush – повышенный приоритет: например, +50% скорость за цену +X% к потреблению топлива/материалов на транспорт[[97]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B0%D1%80%D0%BC%D0%B8%D0%B9%3B%20%D0%BF%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%B4%D1%91%D1%82%20%D0%BF%D0%BE%20%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E). Также, для удобства, игрок может задавать домашние узлы для армий – куда по умолчанию направлять подкрепления, чтобы не делать постоянно ручные заказы[[98]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B0%D1%80%D0%BC%D0%B8%D0%B9%3B%20%D0%BF%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%B4%D1%91%D1%82%20%D0%BF%D0%BE%20%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E).

Резервирование транспорта: Если несколько заказов идут по одному маршруту, система бронирует места: чтобы не было ситуации, что низкоприоритетный конвой занял весь коридор. Высокоприоритетные могут перегнать, но с дополнительными затратами. В случае отмены/завершения заказов, освобождаются слоты и выстраивается очередь заново[[93]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%81%D1%87%D0%B8%D1%82%D0%B0%D0%B5%D1%82%20,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2).

UI логистики: Предусмотрен экран логистики – список активных приказов с указанием откуда→куда, содержимого, статуса (в пути / ожидает / прибыл) и ETA прибытия[[99]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87). Игрок может нажать “ускорить” на конкретном заказе, что переведет его в режим Rush (если готов потратить доп. ресурсы).

Отладка логистики: Проверить, что при переброске войска исчезают из исходного узла и появляются только по прибытию в целевой (никакого телепорта). Симулировать нагруженность: отправить больше, чем capacity, и убедиться, что лишние ждут и корректно переносятся на следующий день. Метрики, которые стоит логировать: средняя загрузка коридоров, время простоя грузов/войск на узлах, бутылочные горлышки (какой ресурс или узел чаще всего ограничивает скорость)[[100]](file://file_0000000004006243943cbb9888dd4b65#:~:text=13%29%20%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%20%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20,%D1%83%D0%B7%D0%BA%D0%B8%D0%B5%20%D0%BC%D0%B5%D1%81%D1%82%D0%B0%20%D0%BF%D0%BE%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%83%2F%D1%83%D0%B7%D0%BB%D1%83).

## Фортификации и оборонительные сооружения

Игрок и ИИ могут строить фортификации на карте – это защитные сооружения, замедляющие распространение инфекции и облегчающие оборону. Строительство подразделяется на временные и постоянные укрепления[[49]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D1%8C%20%D0%BD%D0%B0%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC).

* Временные: окопы, наблюдательные пункты, мешки с песком – возводятся быстро (часы), требуют мало ресурсов, дают лёгкое укрытие. Но они имеют высокую деградацию: со временем их эффективность падает, под натиском (обстрелом, заражением) разрушаются относительно быстро[[101]](file://file_0000000004006243943cbb9888dd4b65#:~:text=12%29%20%D0%A4%D0%BE%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8%3A%20%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8%20%28SHOULD%29%201,%D0%BF%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D1%8B%D0%B5%20%E2%80%94%20%D0%BC%D0%B5%D0%B4%D0%BB%D0%B5%D0%BD%D0%BD%D0%B5%D0%B5%20%D0%B8%20%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%BE%D0%BF%D1%80%D0%B8%D0%B3%D0%BE%D0%B4%D0%BD%D1%8B).
* Постоянные: ДОТы, бункеры, капониры, рвы – строятся долго (дни) и дорого, требуют много материалов и труда, иногда потребляют энергию, но дают серьёзную защиту и служат долго[[49]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D1%8C%20%D0%BD%D0%B0%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC). Постоянные укрепления могут поддерживаться ремонтом, их деградация медленная и они ремонтопригодны[[101]](file://file_0000000004006243943cbb9888dd4b65#:~:text=12%29%20%D0%A4%D0%BE%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8%3A%20%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8%20%28SHOULD%29%201,%D0%BF%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D1%8B%D0%B5%20%E2%80%94%20%D0%BC%D0%B5%D0%B4%D0%BB%D0%B5%D0%BD%D0%BD%D0%B5%D0%B5%20%D0%B8%20%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%BE%D0%BF%D1%80%D0%B8%D0%B3%D0%BE%D0%B4%D0%BD%D1%8B).

Параметры фортификаций: Каждому типу задана стоимость (в трудочасах, материалах, возможно энергии), время строительства, уровень защиты/укрытия (например, процент снижения урона или вероятности заражения за укреплением), требования обслуживания (нужно ли тратить что-то на поддержание)[[49]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D1%8C%20%D0%BD%D0%B0%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC).

Строительство: Фортификации возводятся специальными инженерными юнитами. Игрок указывает что и где строить, а бригады инженеров начинают работу. Можно строить линейные объекты (например, окоп вдоль линии) – тогда стоимость масштабируется с длиной, и задача автоматически дробится между доступными бригадами[[102]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0%3A%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC%20%D0%B1%D1%80%D0%B8%D0%B3%D0%B0%D0%B4%2F%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0). Также можно строить узловые точки обороны – отдельные укрепления (башни, ДОТы). Возможны улучшения: временное укрепление можно модернизировать в постоянное без полного сноса, с частичным зачетом уже вложенных ресурсов[[102]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0%3A%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC%20%D0%B1%D1%80%D0%B8%D0%B3%D0%B0%D0%B4%2F%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0).

Влияние на инфекцию: Фортификации выполняют двойную роль – как оборонительные сооружения от врагов (если бы они были) и как барьеры для инфекции. В терминах модели заражения они выступают как модификаторы, снижающие вероятность распространения через клетку или ребро сетки. Например, карантинный редут (санитарный кордон) может уменьшать шанс передачи инфекции в соседнюю область на X%. Это отражается в InfectionSystem.Tick (модификаторы в шаге диффузии)[[57]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%2F%2F%20Application%2FInfectionSystem%20public%20sealed%20class,%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%D1%85%20%7D%20%7D). Также фортификации влияют на AI: защищенные регионы получают больший вес как “сдержанные” – Utility AI может снизить приоритет вкачивать ресурсы в уже хорошо укрепленный участок, переключаясь на более слабые места.

UI строительства фортификаций: В интерфейсе будет режим строительства: игрок выбирает тип (выпадающий список или панель с иконками: временные, постоянные)[[50]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87), затем кликом на карте ставит заказ. Для линий – возможно, рисует линию. Показываются требования (сколько материалов и времени потребуется) и, после заказа, он появляется в очереди задач строительства с указанием прогресса[[103]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D0%BC%D0%B5%D1%81%D1%82%D0%BE%20%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D1%8F.%204.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87). На экране фортификаций можно видеть все текущие стройки и ремонты.

Отладка фортификаций: Убедиться, что параллельность строительства соответствует числу бригад: если доступно 2 инженерных взвода, постройка идет в 2 раза быстрее или две сразу. Проверить, что улучшение (апгрейд) учитывает уже потраченные ресурсы (например, если окоп улучшать до бункера, часть работ уже сделана). Смоделировать деградацию: запустить время вперед (например, через debug) и увидеть, что эффективность временных сооружений падает, а сообщение «укрепление нуждается в ремонте» появляется.

## Pathfinding (пути юнитов) и карты пути

Юниты (например, военные или конвои) перемещаются по сетке с помощью алгоритмов поиска пути. Базовая реализация – алгоритм A. Однако, учитывая размеры карты, сразу закладываются оптимизации: Jump Point Search (JPS) и иерархический поиск HPA\*\*.

* A: классический алгоритм поиска кратчайшего пути на grid-графе. Реализуем сначала его для корректности. В отладочном режиме делаем визуализацию: при запросе пути, рисуем на сцене\* через Gizmos или LineRenderer открытые и закрытые узлы, а также итоговый путь[[104]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn3search5%EE%88%82turn3search20%EE%88%81). Это моментально выявит проблемы с эвристикой или проходимостью (можно увидеть, где поиск буксует). Также сравниваем длину пути с эталоном (например, с прямой дистанцией \* стоимость шага) – для сетки без препятствий разница должна быть минимальна.
* Jump Point Search (JPS): оптимизация над A, сокращающая поиск за счет пропуска симметричных путей. На равномерной сетке JPS ускоряет A\* в разы, зачастую на порядок[[105]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=breaking%20algorithm%20which%20speeds%20up,including%20abstraction%20and%20memory%20heuristics)[[106]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=considered,existing%20speedup%20techniques%20%E2%80%94%20including). Он не требует предварительной подготовки данных и не расходует доп. память – алгоритм во время поиска “перепрыгивает” через последовательности тривиальных узлов. Мы интегрируем JPS в поиск: после отладки A включаем опцию JPS, убеждаемся, что найденные пути идентичны по длине (оптимальные) и что время поиска снизилось (например, логировать время или считать число рассмотренных узлов)[[107]](https://jair.org/index.php/jair/article/view/12255#:~:text=Regarding%20Goal%20Bounding%20and%20Jump,based%20optimal%20pathfinding). Есть также улучшенная версия JPS+ (offline-оптимизации, как кэширование)[[108]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BF%D1%80%D0%BE%D1%85%D0%BE%D0%B4%20%D1%87%D0%B5%D1%80%D0%B5%D0%B7%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D1%91%D0%BD%D0%BD%D1%8B%D0%B5%20%C2%AB%D0%BA%D1%80%D0%B0%D1%81%D0%BD%D1%8B%D0%B5%C2%BB%20%D0%B7%D0%BE%D0%BD%D1%8B), но на первом этапе достаточно базового JPS. Согласно исследованиям, JPS может ускорить поиск более чем в 10 раз на больших сетках[[105]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=breaking%20algorithm%20which%20speeds%20up,including%20abstraction%20and%20memory%20heuristics)[[109]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=2.%20It%20involves%20no%20pre,%5B1), и часто не уступает сложным иерархическим техникам.
* HPA (Hierarchical Pathfinding A): для очень больших карт применяем иерархию. Суть: разбить карту на кластеры (блоки, например 10×10 клеток), внутри каждого заранее посчитать “внутренние” пути между входами, и построить граф кластеров[[110]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=Hierarchical%20Path,clusters%2C%20the%20maximal%20sized%20entrances)[[111]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=3,run%20can%20be%20used). Тогда поиск между далекими точками сначала идет на абстрактном графе кластеров, затем спускается внутрь. HPA дает резкое ускорение на огромных картах, уменьшая число узлов в разы[[110]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=Hierarchical%20Path,clusters%2C%20the%20maximal%20sized%20entrances)[[112]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=the%20nodes%20in%20the%20abstract,graph%20instead%20of%20the). В нашем плане HPA – опциональная оптимизация (можно реализовать, если карта действительно большая). Но мы закладываем ее возможность: код архитектуры разделяет поиск на слои (Application/Pathfinding), что облегчает внедрение HPA\* при необходимости[[3]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%BB%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%8F).

Дополнения к Pathfinding: После нахождения пути добавляется сглаживание (например, алгоритм string-pulling, чтобы убрать лишние зигзаги). Также проверяется, что маршрут не ведет через опасные зоны: если какие-то клетки сильно заражены (выше порога), pathfinding может увеличивать вес этих клеток или вообще их избегать (чтобы конвои не ехали через “красную зону”). Это интегрируется как дополнительная стоимость в графе (например, вес ребра += инфекция\*большой коэффициент).

Асинхронный поиск: Запросы на путь можно выполнять во вторичном потоке, особенно если юнитов много. Unity Jobs позволяют запустить несколько параллельных поисков (batch pathfinding), чтобы на основном потоке лишь получать результаты[[113]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D1%80%20%C2%AB%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%E2%86%92%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%C2%BB%20%D0%B8%20%D0%BA%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D1%85%20%C2%AB%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%C2%BB). Нужно обеспечить потокобезопасность (никаких обращений к UnityAPI внутри поиска, только данных).

Отладка поиска: Помимо визуализации узлов, сделать сравнение JPS vs A\*: для нескольких типовых маршрутов логировать длину и время. Проверить частичные обновления: при строительстве нового здания, например, один кластер становится менее проходимым – убедиться, что HPA обновляет только затронутые кластеры, а не весь граф[[113]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D1%80%20%C2%AB%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%E2%86%92%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%C2%BB%20%D0%B8%20%D0%BA%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D1%85%20%C2%AB%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%C2%BB). Кешировать популярные маршруты: если известно, что часто идет трафик между двумя конкретными регионами, можно сохранять последний успешный путь или хотя бы дистанцию (Goal Bounding).

## Оптимизация производительности и масштабирование

По мере роста сложности симуляции важно предусмотреть шаги оптимизации и инструменты профилирования.

Эскалация технологий: Наша стратегия – начать с MonoBehaviour и простых обновлений, затем по мере возникновения узких мест переносить тяжелые части на C# Jobs + Burst, и в крайнем случае – на DOTS/ECS[[114]](file://file_00000000bdd462439d54689f98b935a3#:~:text=8%29%20%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B8%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20,%D1%82%D1%8F%D0%B6%D1%91%D0%BB%D1%8B%D0%B5%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%80%D0%B0%D0%B7%D0%B1%D0%B8%D0%B2%D0%B0%D0%B9%20%D0%BF%D0%BE%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B0%D0%BC)[[115]](file://file_00000000bea06243b50b33d9516a800b#:~:text=8%29%20%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B8%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20,Sprite%20Atlas%20%D0%B4%D0%BB%D1%8F%20%D1%81%D0%BE%D1%80%D1%82%D0%B8%D0%BD%D0%B3%D0%B0). Например, первоначально InfectionSystem и AI пишутся обычным образом; когда количество агентов/клеток растет и производительности не хватает, выносим расчеты распространения инфекции, влияние карт, массовые сканы в Jobs с использованием NativeArray и пометкой [BurstCompile][[116]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D1%8D%D1%82%D0%BE%20%D0%B1%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%8F%20%D0%B4%D0%BE%D1%80%D0%BE%D0%B6%D0%BA%D0%B0%20%D0%BA%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8E)[[117]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%20%28%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%B8%D0%BA%D0%B0%29%3A%2A%2A%20,%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8%20%D0%B8%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn2search16%EE%88%82turn2search10%EE%88%81). Это даст прирост без кардинального переписывания логики. Если же и этого мало (например, карта огромна или 10000 агентов), «горячие» системы переносим на ECS: создаем соответствующие компоненты (Cell, Agent) и системы, работающие с ними параллельно, с фиксированным шагом[[118]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%82%D1%8F%D0%B6%D1%91%D0%BB%D1%8B%D0%B5%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%80%D0%B0%D0%B7%D0%B1%D0%B8%D0%B2%D0%B0%D0%B9%20%D0%BF%D0%BE%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B0%D0%BC). В Entities важно строго использовать EntityCommandBuffer для создания/удаления сущностей, и выполнять их в конце кадра, чтобы избежать множественных sync-point’ов[[119]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,point%E2%80%99%D0%BE%D0%B2%20%D0%B2%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B5%20%EE%88%80cite%EE%88%82turn0search4%EE%88%82turn0search8%EE%88%81). Мы планируем экспериментально проверить DOTS на подсистеме инфекции (POC, см. этапы разработки) – это даст опыт, насколько оправдан переход.

Профилирование с самого начала: Уже на 1-2 день подключаем Unity Profiler и Frame Debugger к проекту[[120]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81). Проверяем базовые вещи: сколько отрисовок (draw calls) дает Tilemap, нет ли переполнения GC при запуске цикла, как ведут себя UI-элементы. Для Tilemap: используем режим Chunk для статичных слоев (минимум draw calls), а для слоев, где должны вставать спрайты между тайлами – режим Individual + Sprite Atlas, чтобы спрайты сортировались правильно[[121]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B2%D1%8B%D0%B8%D0%B3%D1%80%D1%8B%D1%88%20%D0%BD%D0%B0%20%D0%B2%D0%B0%D1%88%D0%B5%D0%B9%20%D1%81%D1%86%D0%B5%D0%BD%D0%B5%20%EE%88%80cite%EE%88%82turn3search13%EE%88%82turn3search0%EE%88%82turn3search1%EE%88%81). Frame Debugger подтвердит, что батчинг работает (тайлы атласированы). Эти оптимизации по рендерингу сразу закладываем, чтобы потом не переделывать карту.

Pool-инг объектов: Часто создаваемые и уничтожаемые объекты (например, снаряды, эффекты выстрелов, всплывающие маркеры) нужно пуллить – то есть переиспользовать из заранее подготовленного пула, вместо постоянного Instantiate/Destroy. Unity предоставляет шаблон ObjectPool<T>, его используем для таких объектов[[122]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B2%D1%8B%D0%B8%D0%B3%D1%80%D1%8B%D1%88%20%D0%BD%D0%B0%20%D0%B2%D0%B0%D1%88%D0%B5%D0%B9%20%D1%81%D1%86%D0%B5%D0%BD%D0%B5%20%EE%88%80cite%EE%88%82turn3search13%EE%88%82turn3search0%EE%88%82turn3search1%EE%88%81). Особенно актуально пуллить объекты, загружаемые через Addressables, т.к. их instantiation более тяжелое на главном потоке. Это предотвратит фризы при массовом создании эффектов.

Частота обновления систем: Оптимизируем за счет разрежения: стратегический AI обновляем реже (раз в N тиков, см. раздел AI), экономику – раз в игровой день. Тяжелые расчеты, которые все же остаются каждый тик, распараллеливаем или делим на части по разным кадрам (например, рассчитывать influence map не всю сразу, а по секторам, обновляя каждый сектор раз в несколько тиков). Такой подход “размазает” нагрузку и избегает спайков.

Unity Profiling Tools: Настраиваем профилировочные сценарии – например, отдельные режимы игры или debug-кнопки, которые нагружают систему: большая карта (чтобы увидеть поведение pathfinding), массовый пожар (много одновременных событий, проверка EventBus и логики) или волна инфекции (чтобы протестировать Jobs на инфекции)[[123]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%E2%80%94%20%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D0%B7%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%82%D1%8F%D0%B6%D1%91%D0%BB%D1%8B%D1%85%20%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%20%D0%BD%D0%B0,%28%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D0%BE%D0%B9%20%D0%B2%D0%B5%D1%82%D0%BA%D0%B5%29%20%EE%88%80cite%EE%88%82turn0search5%EE%88%81). Прямо в dev-меню можно сделать кнопки, запускающие эти сценарии, и кнопку “Открыть Profiler/Frame Debugger”[[124]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D0%BC%D0%B5%D0%BD%D1%8E.%20%EE%88%80cite%EE%88%82turn0search12%EE%88%82turn0search20%EE%88%81) для удобства. Это ускорит выявление узких мест.

Конкретные оптимизации (сводка): Использовать NativeArray и Burst в джобах для вычисления диффузии инфекции, генерации influence-карт, массовых выборок по сетке – это типичные задачи, где Burst легко дает ×5-10 к скорости[[125]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%20%28%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%B8%D0%BA%D0%B0%29%3A%2A%2A%20,%D0%B8%20%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D0%B8%D1%82%D0%B8%D0%BA%D0%B8%20%D0%B2%20Addressables). В ECS использовать FixedStepSimulationSystemGroup для систем, критичных к фиксированному шагу (чтобы они не зависели от Time.deltaTime плавучего), как это сделано в DOTS-Physics[[126]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%82%D1%8B,%D0%B8%20%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D0%B8%D1%82%D0%B8%D0%BA%D0%B8%20%D0%B2%20Addressables). Соблюдать баланс между системами: например, разделить группы обновления по фазам (Initialization, Simulation, Presentation) и проверять, что все системы в правильных группах, чтобы не было лишних sync точек[[127]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D0%B8%20%D0%B0%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D1%8B%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.%20%EE%88%80cite%EE%88%82turn0search13%EE%88%81). Использовать потоковую запись логов осторожно: Debug.Log внутри tight loop джоба может сильно замедлить, поэтому для отладки Bursted-кода лучше временно отключать Burst или использовать Unity.Burst.BurstCompiler.EnableUnsafeStaticBlock для условного логирования[[128]](file://file_00000000a1386243b45d516491d7291b#:~:text=%E2%80%94%20%D0%92%D1%8B%D0%BD%D0%B5%D1%81%D0%B8%20%D0%B4%D0%B8%D1%84%D1%84%D1%83%D0%B7%D0%B8%D1%8E%2F%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20Jobs,%D1%83%20%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B3%D0%BE%20job.%20%EE%88%80cite%EE%88%82turn1search11%EE%88%81).

Память: Следить за алокациями. В идеале, в горячих циклах (Tick) не должно быть вообще сборок мусора. Этого добиваемся использованием пулов, preallocate коллекций, и избегая LINQ/Alloc внутри Update. В профайлере режим Profile Editor (или Attach to Player) позволит увидеть, если какой-то метод генерирует GC Alloc каждый кадр. Такие места переписываем.

## Интерфейс и инструменты отладки

UI разрабатывается не только для игрока, но и для внутреннего дебага. В проекте будет специальное developer overlay-меню, показывающее техническую информацию и дающее контроль для тестирования.

Оверлей тиков и скорости: В углу экрана отображается текущий режим времени (пауза/×0.5/×1/×2/×4) и счетчик тиков симуляции[[129]](file://file_0000000077706243ab98f6cd120afa5f#:~:text=%D0%94%D0%BE%D0%B1%D0%B0%D0%B2%D1%8C%D1%82%D0%B5%20UI,%D0%BC%D0%B5%D0%BD%D1%8E%20%D0%B4%D0%BB%D1%8F%20%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%B0)[[18]](file://file_00000000a1386243b45d516491d7291b#:~:text=,tick). Там же – кнопки управления: “Пауза/Продолжить”, “Шаг” (один тик), переключатели скорости. Это служебный UI для разработчиков, но и игроку можно часть оставить (управление скоростью часто присутствует в RTS).

Тепловые карты и графики: Для отладки сложных систем вводятся режимы визуализации. К примеру, переключатель “Показать инфекцию” раскрашивает тайлы градиентом от зеленого (0% заражения) к красному (100%)[[130]](file://file_00000000a1386243b45d516491d7291b#:~:text=2%29%20%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5%20%C2%AB%D0%B8%D0%BD%D1%84%D0%BB%D1%8E%D0%B5%D0%BD%D1%81,%EE%88%80cite%EE%88%82turn2search7%EE%88%82turn2search3%EE%88%81). Так сразу видно очаги и помогает отследить, корректно ли работает распространение (где-то скачок цвета – возможно баг). Другой режим – “Карта влияния” по своим войскам и врагу: можно отладить, правильно ли считается influence AI (полезно при настройке Utility AI, чтобы убедиться, что, скажем, врагов ИИ “видит” как красные зоны и реагирует)[[130]](file://file_00000000a1386243b45d516491d7291b#:~:text=2%29%20%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5%20%C2%AB%D0%B8%D0%BD%D1%84%D0%BB%D1%8E%D0%B5%D0%BD%D1%81,%EE%88%80cite%EE%88%82turn2search7%EE%88%82turn2search3%EE%88%81). Также режим “статус клеток” – текстово показывать на тайле основные данные (уровень заражения, население, укрепление). Эти инструменты ускоряют балансировку.

Визуализация AI-планов: По запросу (кнопка “Показать план ИИ”) на экране может появляться схематичная диаграмма или просто текст, отражающий текущий план GOAP стратегического ИИ (см. лог выше) и активную ветку Behavior Tree для выбранного юнита[[83]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search2%EE%88%81)[[75]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81). Например, выбрав инженера, разработчик увидит: “BT: Patrol → Repair (running)” – значит, сейчас ремонтирует. А стратегический план может показать: “Goal: Contain outbreak in Region 3 → Plan: Build Fortifications (75% done)”. Такая визуализация сильно помогает удостовериться, что AI делает то, что мы ожидаем.

Логирование событий: В dev-меню можно вывести окно Event Log, где показываются последние важные игровые события: “Day 30: Power deficit – shutting down Factory 1”, “New research completed: Vaccine I”, “Outbreak intensified in Sector B (40% infected)” и т.д. Это в основном для дизайнеров и тестеров, чтобы просматривать историю происходящего и замечать, если какое-то событие отсутствует или неправильно формируется. EventBus упрощает сбор такой ленты – достаточно подписать UI-лог на нужные типы событий.

Быстрый доступ к профилю: Dev-меню содержит кнопки: “Open Profiler” и “Open Frame Debugger”. Нажатие быстро активирует соответствующие Unity-инструменты (для редактора)[[124]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D0%BC%D0%B5%D0%BD%D1%8E.%20%EE%88%80cite%EE%88%82turn0search12%EE%88%82turn0search20%EE%88%81). Это не часть билда для игрока, но экономит время разработчикам при тестировании оптимизаций.

Unit Test Runner интеграция: Хотя это не UI игры, а инструментарий, упомянем: Unity Test Framework предоставляет окошко со списком тестов. Мы создадим категории EditMode и PlayMode тестов (подробности ниже). Важно интегрировать их в процесс – например, настроить, чтобы при каждом CI-билде они пробегали, и в самом Unity можно было в один клик запустить весь тестпак.

## Тестирование, отладка и CI/CD процесс

Автоматическое тестирование: В проекте будет набор юнит-тестов (Edit Mode) и интеграционных плейтестов (Play Mode) с помощью Unity Test Framework. - Edit Mode тесты запускаются прямо в редакторе (без сцены). Ими покрываем чистую логику: экономические расчеты (см. инварианты выше), модели заражения (например, что SIR модель с заданными параметрами дает ожидаемый исход), функции предусловий/эффектов GOAP (проверить, что Action действительно приводит WorldState к нужному эффекту)[[83]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search2%EE%88%81). Эти тесты быстрые и могут гоняться часто (например, при каждом коммите). - Play Mode тесты запускаются во встроенном игроке или Editor с эмуляцией сцены. Например, можно сделать фикстуру: маленькая карта 10×10, один очаг инфекции, пара юнитов, один город – и проиграть 200 тиков симуляции, затем проверить итоговые инварианты (не вылез ли ресурс в отрицание, сохранилось ли равновесие, правильно ли сохраняется/грузится состояние)[[131]](https://docs.unity3d.com/Packages/com.unity.test-framework@1.0/manual/edit-mode-vs-play-mode-tests.html#:~:text=Manual%20docs,Editor%20and%20test%20game%20code)[[132]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search0%EE%88%82turn2search12%EE%88%81). Также можно автоматически загружать/сохранять игру в тесте, сравнивать определенные поля до/после. Такой автотест выступает как “дымовой” интеграционный тест.

Unity позволяет запускать эти тесты на CI. Мы планируем настроить Continuous Integration/Delivery: каждый пуш в репозиторий будет вызывать автоматическую сборку проекта и прогоны тестов. Например, через GitHub Actions или Unity Cloud Build можно настроить выполнение EditMode и PlayMode тестов на headless-режиме Unity[[133]](https://samuel-asher-rivello.medium.com/best-practices-6-ci-cd-with-unity-buildalon-be3286f05274#:~:text=In%20today%E2%80%99s%20fast,work%20and%20catch%20issues%20early). Это гарантирует, что при внесении новых изменений мы не ломаем существующую функциональность – тесты сразу поймают регресс.

Continuous Integration (CI): внедрение CI существенно повысит надежность. При каждом слиянии кода автоматическая сборка будет проверять, что проект компилируется и все тесты проходят[[134]](https://unity.com/topics/what-is-ci-cd#:~:text=CI%20is%20a%20DevOps%20best,can%20snowball%20into%20bigger%20issues). Unity предоставляет инструменты для этого (например, пакет Test Runner можно запускать через командную строку). Преимущества CI – ошибки ловятся рано, команда всегда имеет рабочую версию проекта[[134]](https://unity.com/topics/what-is-ci-cd#:~:text=CI%20is%20a%20DevOps%20best,can%20snowball%20into%20bigger%20issues). Мы придерживаемся принципа частых малых коммитов, чтобы интеграция шла постоянно, а не большими редкими кусками – это ускоряет выявление проблем.

Continuous Delivery (CD): настроим автоматическую сборку игровых билдов по расписанию (например, раз в неделю ночной билд) или по тегу релиза. Unity Cloud Build или Jenkins могут собирать проект под нужные платформы, прогонять тесты и выгружать сборку тестировщикам. Это позволит в конце квартала иметь не абстрактный код, а конкретный MVP билд игры, готовый к демонстрации. К концу данного плана ожидается MVP-срез (см. ниже критерии готовности).

DoD – критерии готовности: Все фичи, помеченные как MUST, должны быть реализованы и покрыты тестами (автотестами или сценариями)[[135]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%28DoD%29%20,%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B9%2C%20%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%BD%D1%8B%D0%B5%20%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B8%20%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D0%B0%D1%85). UI-интерфейсы (слайдеры %, экран отчета, логистика, строительство фортификаций) должны работать и отображать реальные данные из системы[[135]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%28DoD%29%20,%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B9%2C%20%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%BD%D1%8B%D0%B5%20%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B8%20%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D0%B0%D1%85). Проверка: ни один ресурс не уходит в минус без обработки, никакие расходы не скрыты от игрока, все критические события (дефицит, прорыв) генерируют уведомления. Если эти условия выполнены и тесты зелены – можно считать квартальный план успешно реализованным.

## Этапы разработки и контроль прогресса (3 месяца)

Процесс разбит на модули (спринты), каждый длится ~1–2 недели, с учебными целями и наращиванием сложности. Ниже представлен поэтапный план на ~12 недель:

1. Недели 1–2: Инициализация проекта и базовый каркас
2. Создать репозиторий, пустой Unity-проект и настроить структуру папок по архитектуре (Domain, Application, etc)[[120]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81). Добавить базовую сцену с Tilemap и системами координат (конвертер мир↔сетка)[[136]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%201%E2%80%932%20,%D1%82%D0%B8%D0%BA%D0%BE%D0%BC%20%D0%B8%20%D0%BF%D0%B0%D1%83%D0%B7%D0%BE%D0%B9).
3. Реализовать SimulationLoop с фиксированным тиком и паузой[[136]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%201%E2%80%932%20,%D1%82%D0%B8%D0%BA%D0%BE%D0%BC%20%D0%B8%20%D0%BF%D0%B0%D1%83%D0%B7%D0%BE%D0%B9). Добавить интерфейс IClock и базовые интерфейсы/классы времени, EventBus (пока пустой шаблон).
4. Проверка производительности ранняя: подключить Unity Profiler и Frame Debugger, проверить отрисовку Tilemap (Chunk vs Individual)[[120]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81). Убедиться, что цикл тика работает стабильно при разных FPS (например, вручную ограничить/освободить FPS и посмотреть на счетчик тиков).
5. Недели 3–4: Основы геймплея – экономика и инфекция v1, сохранение
6. Создать ScriptableObject-дефы: Units, Buildings, Techs, InfectionParams (пустые объекты с полями, чтобы заполнить через инспектор)[[137]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%94%D0%BD%D0%B8%203%E2%80%935%2A%2A%20%E2%80%94%20SO,%E2%86%92%20%C3%972%20%E2%86%92%20%D1%81%D0%B5%D0%B9%D0%B2%2F%D0%BB%D0%BE%D0%B0%D0%B4%20%EE%88%80cite%EE%88%82turn2search0%EE%88%81). На их основе реализовать Runtime-модели (классы Domain) и первые системы: EconomySystem v1 и InfectionSystem v1. Экономика v1 – простая: один ресурс (например, материалы как совокупный бюджет) и 2 кошелька (гос и игрок) с возможностью TryReserve и Commit. Infection v1 – простейший клеточный автомат SIR: каждая клетка имеет state, в каждом тик соседние заражают с фиксированной вероятностью[[137]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%94%D0%BD%D0%B8%203%E2%80%935%2A%2A%20%E2%80%94%20SO,%E2%86%92%20%C3%972%20%E2%86%92%20%D1%81%D0%B5%D0%B9%D0%B2%2F%D0%BB%D0%BE%D0%B0%D0%B4%20%EE%88%80cite%EE%88%82turn2search0%EE%88%81).
7. Реализовать сохранение/загрузку состояния игры в JSON (используя JsonUtility). Учесть ограничения: например, вместо Dictionary в моделях использовать списки или сериализуемые классы[[23]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%28%D0%B7%D0%B5%D1%80%D0%BA%D0%B0%D0%BB%D0%B8%D1%82%D1%8C%20%D0%B2%D1%8B%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8). Добавить поле версии в сохраняемые данные. Протестировать end-to-end: запустить игру, промотать несколько тиков, сохранить, загрузить – убедиться, что состояние идентично (балансы, статус клеток и т.д.)[[138]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%E2%80%94%20SO,%E2%86%92%20%C3%972%20%E2%86%92%20%D1%81%D0%B5%D0%B9%D0%B2%2F%D0%BB%D0%BE%D0%B0%D0%B4%20%EE%88%80cite%EE%88%82turn2search0%EE%88%81). Этот тест можно автоматизировать (PlayMode тест: старт → несколько тиков → сейв → лоад → сравнение)[[138]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%E2%80%94%20SO,%E2%86%92%20%C3%972%20%E2%86%92%20%D1%81%D0%B5%D0%B9%D0%B2%2F%D0%BB%D0%BE%D0%B0%D0%B4%20%EE%88%80cite%EE%88%82turn2search0%EE%88%81).
8. Недели 5–6: ИИ – базовые решения и влияние, расширение команд
9. Добавить первые Utility-скоры (3–5 штук) для гос-ИИ: например, приоритет сдерживания инфекции, приоритет укрепления узкого прохода, приоритет инвестировать в конкретное исследование[[139]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,BT%20%D0%B4%D0%BB%D1%8F%20%D0%BF%D0%B0%D1%82%D1%80%D1%83%D0%BB%D1%8F%2F%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%B0). Пока можно захардкодить их возвращающие рандом или по простым формулам, главное – интегрировать структуру Utility AI. Сделать простой стратегический тикер: отдельный MonoBehaviour или часть SimulationLoop, который раз в N тиков выводит в лог выбранную цель ИИ на основе Utility.
10. Сформировать первую Influence Map – карта заражения: массив float по тайлам, где 0 = чисто, 1 = полностью заражено. Каждый тик обновлять ее на основе InfectionSystem (или даже внутри InfectionSystem). Визуализировать эту карту через UI оверлей (см. UI раздел) – цветом или интенсивностью[[140]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%94%D0%BD%D0%B8%206%E2%80%938%2A%2A%20%E2%80%94%20Utility,T%3E%60%29%20%EE%88%80cite%EE%88%82turn6search6%EE%88%81). Добавить концепцию “линии фронта” – список клеток на границе заражения, и тоже отобразить (например, контуром).
11. Внедрить EventBus полностью и добавить обработку базовых команд: например, команду PlaceBuilding (строительство) и StartResearch (начать исследование). Пока реализации могут быть упрощенными (PlaceBuilding сразу строит без времени, Research сразу завершается), но структура событий должна заработать[[141]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%203%E2%80%935%20,%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0%20S%2FI%2FR). Протестировать: при вызове команды через код или UI сообщение попадает в EventBus, исполняется сервисом и приводит к изменению состояния. Убедиться, что EventBus.Flush вызывается после тиков, как запланировано.
12. Недели 7–8: Продвинутый AI и Pathfinding
13. GOAP-планировщик v1: реализовать упрощенный планировщик с 2–3 типами действий и одной простой целью. Например, цель “улучшить медицину” может требовать действий: накопить X ресурсов → выполнить исследование вакцины[[139]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,BT%20%D0%B4%D0%BB%D1%8F%20%D0%BF%D0%B0%D1%82%D1%80%D1%83%D0%BB%D1%8F%2F%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%B0). Можно использовать брутфорс поиск по небольшому пространству действий. Подключить GOAP к стратегическому ИИ: Utility AI выбирает цель, GOAP планирует, потом в лог выводится план и начинается исполнение.
14. Behavior Tree v1: разработать несколько узлов BT для тактических юнитов. Например, патруль по маршруту (циклическая последовательность точек), ремонт ближайшего строения, реагирование на врага (пока врагами может выступать инфекция как абстрактный враг). Построить небольшое дерево и навесить его на условный “инженерный юнит” или “солдата” для теста. Проверить, что BT корректно переключается между узлами (можно логировать активные ноды).
15. Pathfinding: интегрировать A для юнитов. Сделать класс PathfindingService, который по запросу выдает путь (список координат). Протестировать на примере: разместить юнит и цель, вызвать поиск – отобразить линию пути через LineRenderer в сцене[[104]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn3search5%EE%88%82turn3search20%EE%88%81). Затем включить оптимизацию JPS\* для поиска на сетке. Убедиться, что путь не изменился (оптимальность сохранена), а скорость возросла (можно профилировать на длинных маршрутах). Включить поддержку пакетного поиска: например, несколько агентов запрашивают путь одновременно – продумать, как избежать подвисания (например, выполнять один путь за кадр, если нет срочности, или использовать Job на поиск)[[113]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D1%80%20%C2%AB%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%E2%86%92%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%C2%BB%20%D0%B8%20%D0%BA%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D1%85%20%C2%AB%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%C2%BB).
16. Недели 9–10: Логистика, фортификации и расширенная экономика
17. Многоресурсная экономика: расширить EconomySystem до 4 ресурсов (труд, энергия, материалы, еда). Реализовать суточный цикл экономики как описано: прирост населения, распределение % ИИ, потребление, производство, очереди строительства/обучения, отчеты[[36]](file://file_0000000004006243943cbb9888dd4b65#:~:text=9%29%20%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%82%D0%B8%D0%BA%D0%B0%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F). Добавить новые здания из design-документа (ферма, электростанция, шахта, казармы и т.д.) с параметрами производства/потребления[[142]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D1%87%D0%B0%D1%81%D1%8B%20%D0%B8%20%D0%B5%D0%B4%D1%83%20%D0%BD%D0%B0%20%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B8). Сделать UI-экран экономики и слайдеры передачи ИИ[[50]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87). Эта часть объемная, возможно ее реализация займет и 11-ю неделю.
18. Логистика система: ввести класс LogisticsManager. Задать несколько узлов (например, City\_A, Front\_N) и соединить их дорогой. Реализовать упрощенно: игрок нажимает “перебросить 1 юнит из City\_A в Front\_N” – система телепортирует (для начала) юнита с задержкой в несколько тиков, симулируя дорогу. Затем усложнить: добавить расчет времени, ограничение пропускной способности, очередь. Протестировать на нескольких параллельных заказах. UI: простейший список текущих трансферов с прогрессом[[99]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87).
19. Фортификации: добавить возможность строить фортификации. Например, в UI добавить кнопку “Построить окоп” – при выборе клетки создается постройка типа Fortification с временем строительства. Инженерные юниты (можно упростить – пусть строится само по себе за заданное время) через каждый тик уменьшают оставшиеся часы. По завершении – на клетке отмечается укрепление (можно поменять спрайт тайла или поставить объект). Интегрировать влияние на инфекцию: если клетка имеет фортификацию, снизить скорость заражения через нее. Добавить в EventBus событие FortificationBuilt для логов.
20. Недели 11–12: Оптимизация, тестирование и финализация
21. Jobs + Burst: профилировать самые тяжелые места (вероятно InfectionSystem, InfluenceMap расчет). Вынести одну-две такие задачи в джобы с Burst. Например, сделать Job для обновления infection grid frontier параллельно[[59]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BE%D0%B1%D1%89%D0%B5%D0%B5%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5). Сравнить время до/после в профайлере. Аналогично для influence map – IJobParallelFor пробегает по тайлам и суммирует влияния. Включить опцию Burst Debug (Native Debug Mode) и убедиться, что все джобы работают без ошибок[[143]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D1%83%20%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B3%D0%BE%20job.%20%EE%88%80cite%EE%88%82turn1search11%EE%88%81).
22. Адресация ассетов: если к этому моменту подключены реальные ассеты (спрайты большого размера и т.п.), перевести их загрузку на Addressables. Протестировать на слабых устройствах или в профайлере память: загружать сцену, потом выгружать – не осталось ли “висящих” объектов.
23. UI и удобство: довести до ума интерфейс: все разработанные экраны (экономика, логистика, отчеты) связать с данными. Добавить всплывающие подсказки, финальные правки.
24. Юнит-тестпак: дописать недостающие тесты, особенно на новые системы экономики (например, сценарий “энергетический голод” – отключения идут правильно по приоритетам)[[100]](file://file_0000000004006243943cbb9888dd4b65#:~:text=13%29%20%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%20%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20,%D1%83%D0%B7%D0%BA%D0%B8%D0%B5%20%D0%BC%D0%B5%D1%81%D1%82%D0%B0%20%D0%BF%D0%BE%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%83%2F%D1%83%D0%B7%D0%BB%D1%83), на логику логистики (ETA считается корректно). Запустить весь тестпак – добиться зеленого статуса.
25. CI/CD: настроить в репо действие CI: сборка проекта в headless и запуск тестов. Проверить, что при ошибке теста CI падает (чтобы не пропустить регресс). Также настроить nightly build на Unity Cloud Build или локальном CI – получить автономный build игры.
26. MVP проверка: Собрать минимально жизнеспособный срез (MVP) по требованиям MUST[[144]](file://file_0000000004006243943cbb9888dd4b65#:~:text=14%29%20%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9%20%D1%81%D1%80%D0%B5%D0%B7%20,%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8). Должно быть: суточный тик с 4 ресурсами, хотя бы одно экономическое здание (ферма) и одно военное (казармы) работают, передача % ИИ влияет на баланс, работает логистика на 1 маршруте, есть один вид временной (окоп) и постоянной (бункер) фортификации, UI отчета дня и слайдеры реализованы[[145]](file://file_0000000004006243943cbb9888dd4b65#:~:text=14%29%20%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9%20%D1%81%D1%80%D0%B5%D0%B7%20,%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8). Провести внутреннее тестирование: сценарии на дефицит, мобилизацию, логистический перегруз[[100]](file://file_0000000004006243943cbb9888dd4b65#:~:text=13%29%20%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%20%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20,%D1%83%D0%B7%D0%BA%D0%B8%D0%B5%20%D0%BC%D0%B5%D1%81%D1%82%D0%B0%20%D0%BF%D0%BE%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%83%2F%D1%83%D0%B7%D0%BB%D1%83). Все выявленные проблемы – в багтрекер и исправить.

При соблюдении этого плана к концу квартала разработчик получит прочное понимание создания архитектуры RTS и оптимизации, а проект будет обладать всеми основными подсистемами и готовым пайплайном для дальнейшего развития. Далее возможна полировка, балансировка и добавление второстепенных фич, но фундамент (архитектура + production pipeline) уже будет заложен.

[[1]](file://file_00000000bea06243b50b33d9516a800b" \l ":~:text=,%D0%B8%20%D0%BA%D0%B0%D1%80%D1%82%20%D0%B4%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D0%BE%20%D1%84%D1%80%D0%BE%D0%BD%D1%82%D1%83) [[2]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,json%2Fbinary%29%20%2B%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F) [[3]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%BB%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%8F) [[4]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,json%2Fbinary%29%20%2B%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F) [[6]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%2F%D1%83%D0%BD%D0%B8%D1%87%D1%82%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9%29%20%D0%B2%D1%81%D0%B5%D0%B3%D0%B4%D0%B0%20%D1%87%D0%B5%D1%80%D0%B5%D0%B7) [[15]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%20%D0%B2%20Unity%20%EE%88%80cite%EE%88%82turn5search2%EE%88%82turn5search1%EE%88%81) [[16]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%EE%88%80cite%EE%88%82turn0search5%EE%88%81.%20,%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%20%D0%B2%20Unity%20%EE%88%80cite%EE%88%82turn5search2%EE%88%82turn5search1%EE%88%81) [[23]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%28%D0%B7%D0%B5%D1%80%D0%BA%D0%B0%D0%BB%D0%B8%D1%82%D1%8C%20%D0%B2%D1%8B%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8) [[25]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B0%20runtime,%D0%BC%D0%B8%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%BF%D1%80%D0%B8%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B5) [[26]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%EE%88%80cite%EE%88%82turn2search10%EE%88%81) [[28]](file://file_00000000bea06243b50b33d9516a800b#:~:text=7%29%20%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2C%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%2C%20%D1%81%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20,50%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BA%D0%B0%D1%85%C2%BB%20%E2%86%92%20%D0%BE%D0%B4%D0%B8%D0%BD%20%D0%B1%D0%B0%D1%82%D1%87) [[29]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D1%82%D1%81%D1%8F%20%D0%B2%20DOTS,%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8%20%D0%B8%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn2search16%EE%88%82turn2search10%EE%88%81) [[32]](file://file_00000000bea06243b50b33d9516a800b#:~:text=4%29%20%D0%AD%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0%20%28%D0%B4%D0%B2%D0%BE%D0%B9%D0%BD%D0%BE%D0%B5%20%D0%92%D0%92%D0%9F%29%20,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8) [[33]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8) [[34]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%92%D0%92%D0%9F%20%D1%83%D1%85%D0%BE%D0%B4%D0%B8%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B5%D0%BD%D1%81%D0%B0%D1%86%D0%B8%D0%B8) [[51]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%BE%D0%B4%D1%85%D0%BE%D0%B4) [[56]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%A7%D1%82%D0%BE%20%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%B8%D1%82%D1%8C%20%28%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%BA%D0%B8%29%3A%2A%2A%20,%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2F%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%BD%D0%BE%20%D1%80%D0%B5%D0%B6%D1%83%D1%82%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE) [[58]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%BA%D0%B0%D0%BD%D0%BE%D0%BD%29%20%EE%88%80cite%EE%88%82turn4search0%EE%88%81.%20,%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2F%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%BD%D0%BE%20%D1%80%D0%B5%D0%B6%D1%83%D1%82%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE) [[59]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BE%D0%B1%D1%89%D0%B5%D0%B5%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5) [[61]](file://file_00000000bea06243b50b33d9516a800b#:~:text=Jobs%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20,%D0%BE%D0%B1%D1%89%D0%B5%D0%B5%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5) [[69]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B8%20%D0%B1%D0%BE%D0%B3%D0%B0%D1%82%D0%B0%20%D0%BD%D0%B0%20%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn1search1%EE%88%81) [[71]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%C2%AB%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20X%C2%BB%2C%20%C2%AB%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D1%82%D1%8C%20%D0%B8%D0%B3%D1%80%D0%BE%D0%BA%D0%B0%C2%BB%20%E2%80%94,%D0%B8%20%D0%B1%D0%BE%D0%B3%D0%B0%D1%82%D0%B0%20%D0%BD%D0%B0%20%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn1search1%EE%88%81) [[72]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D1%8F%D1%82%D1%8C%20%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%BF%D1%80%D0%B8%20%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D1%8B) [[78]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%EE%88%80cite%EE%88%82turn1search0%EE%88%81.%20,%D0%B8%20%D0%B1%D0%BE%D0%B3%D0%B0%D1%82%D0%B0%20%D0%BD%D0%B0%20%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn1search1%EE%88%81) [[79]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D1%8F%D1%82%D1%8C%20%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%BF%D1%80%D0%B8%20%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D1%8B) [[80]](file://file_00000000bea06243b50b33d9516a800b#:~:text=2,%D1%86%D0%B8%D0%BA%D0%BB%D1%83) [[81]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D1%8F%D1%82%D1%8C%20%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%BF%D1%80%D0%B8%20%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D1%8B) [[89]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,50%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BA%D0%B0%D1%85%C2%BB%20%E2%86%92%20%D0%BE%D0%B4%D0%B8%D0%BD%20%D0%B1%D0%B0%D1%82%D1%87) [[90]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%2F%D1%84%D0%BB%D0%B0%D1%88%20%D0%BF%D0%BE%20%D0%BA%D0%BE%D0%BD%D1%86%D1%83%20%D1%82%D0%B8%D0%BA%D0%B0) [[108]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%BF%D1%80%D0%BE%D1%85%D0%BE%D0%B4%20%D1%87%D0%B5%D1%80%D0%B5%D0%B7%20%D0%B7%D0%B0%D1%80%D0%B0%D0%B6%D1%91%D0%BD%D0%BD%D1%8B%D0%B5%20%C2%AB%D0%BA%D1%80%D0%B0%D1%81%D0%BD%D1%8B%D0%B5%C2%BB%20%D0%B7%D0%BE%D0%BD%D1%8B) [[113]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%3A%2A%2A%20,%D0%BF%D0%B0%D1%80%20%C2%AB%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%E2%86%92%D1%80%D0%B5%D0%B3%D0%B8%D0%BE%D0%BD%C2%BB%20%D0%B8%20%D0%BA%D1%80%D0%B0%D1%82%D1%87%D0%B0%D0%B9%D1%88%D0%B8%D1%85%20%C2%AB%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%C2%BB) [[115]](file://file_00000000bea06243b50b33d9516a800b#:~:text=8%29%20%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B8%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20,Sprite%20Atlas%20%D0%B4%D0%BB%D1%8F%20%D1%81%D0%BE%D1%80%D1%82%D0%B8%D0%BD%D0%B3%D0%B0) [[116]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D1%8D%D1%82%D0%BE%20%D0%B1%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%8F%20%D0%B4%D0%BE%D1%80%D0%BE%D0%B6%D0%BA%D0%B0%20%D0%BA%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8E) [[117]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%20%28%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%B8%D0%BA%D0%B0%29%3A%2A%2A%20,%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8%20%D0%B8%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%20%EE%88%80cite%EE%88%82turn2search16%EE%88%82turn2search10%EE%88%81) [[119]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,point%E2%80%99%D0%BE%D0%B2%20%D0%B2%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B5%20%EE%88%80cite%EE%88%82turn0search4%EE%88%82turn0search8%EE%88%81) [[120]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81) [[121]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B2%D1%8B%D0%B8%D0%B3%D1%80%D1%8B%D1%88%20%D0%BD%D0%B0%20%D0%B2%D0%B0%D1%88%D0%B5%D0%B9%20%D1%81%D1%86%D0%B5%D0%BD%D0%B5%20%EE%88%80cite%EE%88%82turn3search13%EE%88%82turn3search0%EE%88%82turn3search1%EE%88%81) [[122]](file://file_00000000bea06243b50b33d9516a800b#:~:text=,%D0%B2%D1%8B%D0%B8%D0%B3%D1%80%D1%8B%D1%88%20%D0%BD%D0%B0%20%D0%B2%D0%B0%D1%88%D0%B5%D0%B9%20%D1%81%D1%86%D0%B5%D0%BD%D0%B5%20%EE%88%80cite%EE%88%82turn3search13%EE%88%82turn3search0%EE%88%82turn3search1%EE%88%81) [[123]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%E2%80%94%20%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D0%B7%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%82%D1%8F%D0%B6%D1%91%D0%BB%D1%8B%D1%85%20%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%20%D0%BD%D0%B0,%28%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D0%BE%D0%B9%20%D0%B2%D0%B5%D1%82%D0%BA%D0%B5%29%20%EE%88%80cite%EE%88%82turn0search5%EE%88%81) [[125]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%9A%D0%B0%D0%BA%20%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B8%D1%82%D1%8C%20%28%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%B8%D0%BA%D0%B0%29%3A%2A%2A%20,%D0%B8%20%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D0%B8%D1%82%D0%B8%D0%BA%D0%B8%20%D0%B2%20Addressables) [[126]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%82%D1%8B,%D0%B8%20%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%BE%D0%B2%D1%8B%D0%B5%20%D0%BF%D0%BE%D0%BB%D0%B8%D1%82%D0%B8%D0%BA%D0%B8%20%D0%B2%20Addressables) [[137]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%94%D0%BD%D0%B8%203%E2%80%935%2A%2A%20%E2%80%94%20SO,%E2%86%92%20%C3%972%20%E2%86%92%20%D1%81%D0%B5%D0%B9%D0%B2%2F%D0%BB%D0%BE%D0%B0%D0%B4%20%EE%88%80cite%EE%88%82turn2search0%EE%88%81) [[138]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%E2%80%94%20SO,%E2%86%92%20%C3%972%20%E2%86%92%20%D1%81%D0%B5%D0%B9%D0%B2%2F%D0%BB%D0%BE%D0%B0%D0%B4%20%EE%88%80cite%EE%88%82turn2search0%EE%88%81) [[140]](file://file_00000000bea06243b50b33d9516a800b#:~:text=%2A%2A%D0%94%D0%BD%D0%B8%206%E2%80%938%2A%2A%20%E2%80%94%20Utility,T%3E%60%29%20%EE%88%80cite%EE%88%82turn6search6%EE%88%81) document-2.txt

<file://file_00000000bea06243b50b33d9516a800b>

[[5]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%3A%20,Json%20%28%D0%B5%D1%81%D0%BB%D0%B8%20%D0%BD%D1%83%D0%B6%D0%B5%D0%BD%20%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B8%D0%B9) [[7]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D0%BE%D1%81%D1%82%D0%B0%D0%BD%D0%B0%D0%B2%D0%BB%D0%B8%D0%B2%D0%B0%D0%B5%D1%82%20%D1%81%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8E%2C%20%D0%BD%D0%BE%20%D0%BD%D0%B5%20UI) [[8]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82%D1%8B%3A%20,%D1%82%D0%B8%D0%BA%20%C2%AB%D0%B8%D0%B7%20%D0%BA%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B8%C2%BB) [[9]](file://file_00000000bdd462439d54689f98b935a3#:~:text=void%20Update%28%29%20,Tick%28tick) [[10]](file://file_00000000bdd462439d54689f98b935a3#:~:text=while%20%28acc%20,Flush%28%29%3B) [[11]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20sealed%20class%20SimulationLoop%20%3A,IsPaused%20%3F%200f%20%3A%20tick) [[12]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,IsPaused%20%3D%20v%3B) [[17]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%82%D0%B8%D0%BA%20%C2%AB%D0%B8%D0%B7%20%D0%BA%D0%BE%D1%80%D0%BE%D0%B1%D0%BA%D0%B8%C2%BB) [[21]](file://file_00000000bdd462439d54689f98b935a3#:~:text=ScriptableObject%20%E2%86%92%20Runtime,%D1%81%D1%82%D0%B0%D1%80%D1%82%D0%B5%20%D0%BF%D1%80%D0%B5%D0%B2%D1%80%D0%B0%D1%89%D0%B0%D0%B5%D0%BC%20%D0%B2%20%D1%87%D0%B8%D1%81%D1%82%D1%8B%D0%B5%20%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B) [[22]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%2F%2F%20Domain%2FTech%2FTech,string%3E%20Prereq%29%3B) [[24]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%2011%E2%80%9314%20,5%2F%C3%971%2F%C3%972%2C%20%D0%BE%D0%B2%D0%B5%D1%80%D0%BB%D0%B5%D0%B8%20%D0%B8%D0%BD%D1%84%D0%B5%D0%BA%D1%86%D0%B8%D0%B8%2F%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F) [[31]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%B8%20%C2%AB%D1%82%D0%BE%D1%87%D0%B5%D1%87%D0%BD%D1%83%D1%8E%20%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D1%83%C2%BB%20%D0%98%D0%98) [[55]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9E%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F%20%D1%82%D0%BE%D1%87%D0%BA%D0%B0%3A%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%20%D0%BD%D0%B0,%D0%B4%D0%BE%D0%BB%D1%8F%20%D0%BD%D0%B0%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%2F%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [[57]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%2F%2F%20Application%2FInfectionSystem%20public%20sealed%20class,%D0%B2%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%D1%85%20%7D%20%7D) [[65]](file://file_00000000bdd462439d54689f98b935a3#:~:text=1,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B) [[66]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20sealed%20class%20HoldLineConsideration%20%3A,) [[67]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%A1%D0%BB%D0%BE%D0%B9%20%D0%B2%D0%BE%D1%81%D0%BF%D1%80%D0%B8%D1%8F%D1%82%D0%B8%D1%8F%3A%20,GridPos%20p) [[68]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%60%60%60csharp%20public%20interface%20IConsideration%20,) [[70]](file://file_00000000bdd462439d54689f98b935a3#:~:text=GOAP%20%28%D1%8D%D1%81%D0%BA%D0%B8%D0%B7%29%3A%20,IAction%3E%20Plan%28Goal%20g%2C%20IWorldState%20w) [[73]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%E2%86%92%20%D0%B8%D1%81%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%C2%AB%D0%A9%D0%B8%D1%82,%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F%20%D1%85%D0%BE%D1%80%D0%BE%D1%88%D0%BE%20%D0%BE%D1%82%D0%BB%D0%B0%D0%B6%D0%B8%D0%B2%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%B8%20%D1%87%D0%B8%D1%82%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B) [[74]](file://file_00000000bdd462439d54689f98b935a3#:~:text=BT%20%28%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BA%D0%BE%D0%B4%20%D1%83%D0%B7%D0%BB%D0%BE%D0%B2%29%3A%20,Patrol%20%29) [[84]](file://file_00000000bdd462439d54689f98b935a3#:~:text=EventBus%20,%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%2C%20%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B1%D1%8E%D0%B4%D0%B6%D0%B5%D1%82%D0%B0) [[85]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20static%20class%20EventBus%20,On%28%28dynamic%29e%29%3B) [[86]](file://file_00000000bdd462439d54689f98b935a3#:~:text=Commands%20%E2%80%94%20%D1%83%D0%B4%D0%BE%D0%B1%D0%BD%D0%BE%20%D0%BB%D0%BE%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%2F%D0%BE%D1%82%D0%BA%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%82%D1%8C%20%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%8F,%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%2C%20%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B1%D1%8E%D0%B4%D0%B6%D0%B5%D1%82%D0%B0) [[87]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0%3A%20,Publish%28order) [[88]](file://file_00000000bdd462439d54689f98b935a3#:~:text=public%20BuildOrder%20Place,Publish%28order%29%3B%20return%20order%3B) [[114]](file://file_00000000bdd462439d54689f98b935a3#:~:text=8%29%20%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B8%20%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20,%D1%82%D1%8F%D0%B6%D1%91%D0%BB%D1%8B%D0%B5%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%80%D0%B0%D0%B7%D0%B1%D0%B8%D0%B2%D0%B0%D0%B9%20%D0%BF%D0%BE%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B0%D0%BC) [[118]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,%D1%82%D1%8F%D0%B6%D1%91%D0%BB%D1%8B%D0%B5%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%80%D0%B0%D0%B7%D0%B1%D0%B8%D0%B2%D0%B0%D0%B9%20%D0%BF%D0%BE%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B0%D0%BC) [[136]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%201%E2%80%932%20,%D1%82%D0%B8%D0%BA%D0%BE%D0%BC%20%D0%B8%20%D0%BF%D0%B0%D1%83%D0%B7%D0%BE%D0%B9) [[139]](file://file_00000000bdd462439d54689f98b935a3#:~:text=,BT%20%D0%B4%D0%BB%D1%8F%20%D0%BF%D0%B0%D1%82%D1%80%D1%83%D0%BB%D1%8F%2F%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%B0) [[141]](file://file_00000000bdd462439d54689f98b935a3#:~:text=%D0%94%D0%B5%D0%BD%D1%8C%203%E2%80%935%20,%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0%20S%2FI%2FR) document.txt

<file://file_00000000bdd462439d54689f98b935a3>

[[13]](file://file_0000000077706243ab98f6cd120afa5f#:~:text=%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D0%B8%20%C3%970,10) [[14]](file://file_0000000077706243ab98f6cd120afa5f#:~:text=%D0%9E%D0%B4%D0%B8%D0%BD%D0%BE%D1%87%D0%BD%D1%8B%D0%B9%20%D1%82%D0%B8%D0%BA%20,11) [[129]](file://file_0000000077706243ab98f6cd120afa5f#:~:text=%D0%94%D0%BE%D0%B1%D0%B0%D0%B2%D1%8C%D1%82%D0%B5%20UI,%D0%BC%D0%B5%D0%BD%D1%8E%20%D0%B4%D0%BB%D1%8F%20%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%B0) Квартальный план разработки RTS-проекта (Unity 2D) для начинающего.docx

<file://file_0000000077706243ab98f6cd120afa5f>

[[18]](file://file_00000000a1386243b45d516491d7291b#:~:text=,tick) [[19]](file://file_00000000a1386243b45d516491d7291b#:~:text=1,%D0%B1%D0%BE%D0%BB%D0%B5%D0%B5%20%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%88%D0%B0%D0%B3%D0%B0%20%D0%BF%D1%80%D0%B8%20Step%C2%BB) [[20]](file://file_00000000a1386243b45d516491d7291b#:~:text=5,%EE%88%80cite%EE%88%82turn0search12%EE%88%81) [[27]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn1search10%EE%88%81) [[30]](file://file_00000000a1386243b45d516491d7291b#:~:text=1%29%20,%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%20%D0%B2%D0%BE%D1%81%D1%81%D1%82%D0%B0%D0%BD%D0%B0%D0%B2%D0%BB%D0%B8%D0%B2%D0%B0%D0%B5%D1%82%20%D0%BF%D0%BE%D0%B4%D1%82%D0%B8%D0%BF%D1%8B) [[52]](file://file_00000000a1386243b45d516491d7291b#:~:text=%2A%2A%D0%9F%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%BA%D0%B8%2F%D1%82%D0%B5%D1%81%D1%82%D1%8B%3A%2A%2A%201%29%20Unit,%C2%AB%D0%B4%D0%B5%D0%BD%D1%8C%D0%B3%D0%B8%20%D0%BD%D0%B5%20%D0%B8%D1%81%D1%87%D0%B5%D0%B7%D0%B0%D1%8E%D1%82%2F%D0%BD%D0%B5%20%D0%BF%D0%BE%D1%8F%D0%B2%D0%BB%D1%8F%D1%8E%D1%82%D1%81%D1%8F%C2%BB) [[53]](file://file_00000000a1386243b45d516491d7291b#:~:text=1%29%20Unit,%C2%AB%D0%B4%D0%B5%D0%BD%D1%8C%D0%B3%D0%B8%20%D0%BD%D0%B5%20%D0%B8%D1%81%D1%87%D0%B5%D0%B7%D0%B0%D1%8E%D1%82%2F%D0%BD%D0%B5%20%D0%BF%D0%BE%D1%8F%D0%B2%D0%BB%D1%8F%D1%8E%D1%82%D1%81%D1%8F%C2%BB) [[54]](file://file_00000000a1386243b45d516491d7291b#:~:text=2,%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D1%8C%2C%20%D1%87%D1%82%D0%BE%20%D0%BD%D0%B5%D1%82%20%D0%B4%D1%80%D0%BE%D0%B6%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%B2) [[60]](file://file_00000000a1386243b45d516491d7291b#:~:text=%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9%20%D0%B2%D1%8B%D0%B1%D1%80%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9%20%D0%BA%D0%BB%D0%B5%D1%82%D0%BA%D0%B8%20%28gizmos%29,%D0%BA%D0%B0%D1%80%D1%82%D1%8B%3A%20%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%B0%D0%B5%D1%82%20%C2%AB%D0%B3%D0%BB%D0%B0%D0%B7%D0%BE%D0%BC%C2%BB%20%D0%BB%D0%BE%D0%B2%D0%B8%D1%82%D1%8C%20%D1%81%D0%BA%D0%B0%D1%87%D0%BA%D0%B8%2F%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D0%B8) [[62]](file://file_00000000a1386243b45d516491d7291b#:~:text=%D0%98%D0%BD%D1%84%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%20) [[63]](file://file_00000000a1386243b45d516491d7291b#:~:text=%EE%88%80cite%EE%88%82turn4search0%EE%88%81%202,%D0%BA%D0%B0%D1%80%D1%82%D1%8B%3A%20%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%B0%D0%B5%D1%82%20%C2%AB%D0%B3%D0%BB%D0%B0%D0%B7%D0%BE%D0%BC%C2%BB%20%D0%BB%D0%BE%D0%B2%D0%B8%D1%82%D1%8C%20%D1%81%D0%BA%D0%B0%D1%87%D0%BA%D0%B8%2F%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D0%B8) [[64]](file://file_00000000a1386243b45d516491d7291b#:~:text=%E2%80%94%20%D1%82%D0%B5%D1%81%D1%82%20%C2%AB%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BD%D0%B5%20%D0%B7%D0%B0%D1%82%D0%B8%D1%80%D0%B0%D0%B5%D1%82%D1%81%D1%8F,%D0%BA%D0%B0%D1%80%D1%82%D1%8B%3A%20%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%B0%D0%B5%D1%82%20%C2%AB%D0%B3%D0%BB%D0%B0%D0%B7%D0%BE%D0%BC%C2%BB%20%D0%BB%D0%BE%D0%B2%D0%B8%D1%82%D1%8C%20%D1%81%D0%BA%D0%B0%D1%87%D0%BA%D0%B8%2F%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D0%B8) [[75]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81) [[82]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5) [[83]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search2%EE%88%81) [[91]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn1search6%EE%88%81) [[104]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn3search5%EE%88%82turn3search20%EE%88%81) [[124]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D0%BC%D0%B5%D0%BD%D1%8E.%20%EE%88%80cite%EE%88%82turn0search12%EE%88%82turn0search20%EE%88%81) [[127]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D0%B8%20%D0%B0%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D1%8B%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.%20%EE%88%80cite%EE%88%82turn0search13%EE%88%81) [[128]](file://file_00000000a1386243b45d516491d7291b#:~:text=%E2%80%94%20%D0%92%D1%8B%D0%BD%D0%B5%D1%81%D0%B8%20%D0%B4%D0%B8%D1%84%D1%84%D1%83%D0%B7%D0%B8%D1%8E%2F%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20Jobs,%D1%83%20%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B3%D0%BE%20job.%20%EE%88%80cite%EE%88%82turn1search11%EE%88%81) [[130]](file://file_00000000a1386243b45d516491d7291b#:~:text=2%29%20%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5%20%C2%AB%D0%B8%D0%BD%D1%84%D0%BB%D1%8E%D0%B5%D0%BD%D1%81,%EE%88%80cite%EE%88%82turn2search7%EE%88%82turn2search3%EE%88%81) [[132]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%EE%88%80cite%EE%88%82turn2search0%EE%88%82turn2search12%EE%88%81) [[143]](file://file_00000000a1386243b45d516491d7291b#:~:text=,%D1%83%20%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B3%D0%BE%20job.%20%EE%88%80cite%EE%88%82turn1search11%EE%88%81) document-3.txt

<file://file_00000000a1386243b45d516491d7291b>

[[35]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%C2%A79) [[36]](file://file_0000000004006243943cbb9888dd4b65#:~:text=9%29%20%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%82%D0%B8%D0%BA%D0%B0%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F) [[37]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,8) [[38]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1,%28%D0%BF%D1%80%D0%B8%20%D0%BD%D0%B0%D0%BB%D0%B8%D1%87%D0%B8%D0%B8%20%D0%BD%D0%B5%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%B8%D1%85%20%D0%98%D0%98) [[39]](file://file_0000000004006243943cbb9888dd4b65#:~:text=2.%20%2A%2A%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0%20,%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%2C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B0%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B5%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8%2C%20%D0%B7%D0%B0%D0%B1%D0%B8%D1%82%D1%8B%D0%B5%20%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B) [[40]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%D0%B2%20%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%2F%D0%B3%D0%B0%D1%80%D0%BD%D0%B8%D0%B7%D0%BE%D0%BD%20%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%20%D0%B7%D0%B0%D0%B2%D0%B5%D1%80%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) [[41]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,) [[42]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%29.%204.%20,%28%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%83%D1%81%D0%BE%D0%B2%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D1%80%D0%BE%D1%81%D0%BE%D0%BA%2C%20%D0%BF%D1%80%D0%B8%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%B2%D1%8B%D0%B1%D1%8B%D1%82%D0%B8%D1%8F) [[43]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%29.%204.%20,%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%2C%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%B0%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B5%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8%2C%20%D0%B7%D0%B0%D0%B1%D0%B8%D1%82%D1%8B%D0%B5%20%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B) [[44]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%2A%2A%D1%88%D1%82%D0%B0%D1%82%D0%BD%D1%83%D1%8E%20%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B2%20%D1%82%D1%80%D1%83%D0%B4%D0%B0,%28%D1%81%D0%BC.%20%C2%A710) [[45]](file://file_0000000004006243943cbb9888dd4b65#:~:text=10%29%20%D0%94%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D1%8B%20%D0%B8%20%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D1%8B%20,%D0%B2%D1%81%D1%91%20%D0%B2%D0%B8%D0%B4%D0%BD%D0%BE%20%D0%B2%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%BC%20%D0%BE%D1%82%D1%87%D1%91%D1%82%D0%B5) [[46]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D0%B5%20%C2%AB%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D0%BE%20%D0%B2%D0%B0%D0%B6%D0%BD%D0%BE%D0%B3%D0%BE%C2%BB%2C%203,%D0%B2%D1%81%D1%91%20%D0%B2%D0%B8%D0%B4%D0%BD%D0%BE%20%D0%B2%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%BC%20%D0%BE%D1%82%D1%87%D1%91%D1%82%D0%B5) [[47]](file://file_0000000004006243943cbb9888dd4b65#:~:text=2.%20%D0%9F%D1%80%D0%B8%20%D0%BD%D0%B5%D1%85%D0%B2%D0%B0%D1%82%D0%BA%D0%B5%20,%D0%B2%D1%81%D1%91%20%D0%B2%D0%B8%D0%B4%D0%BD%D0%BE%20%D0%B2%20%D1%81%D1%83%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%BC%20%D0%BE%D1%82%D1%87%D1%91%D1%82%D0%B5) [[48]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D1%87%D0%B0%D1%81%D1%8B%20%D0%B8%20%D0%B5%D0%B4%D1%83%20%D0%BD%D0%B0%20%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B8) [[49]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D1%8C%20%D0%BD%D0%B0%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC) [[50]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87) [[92]](file://file_0000000004006243943cbb9888dd4b65#:~:text=5,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2) [[93]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D1%81%D1%87%D0%B8%D1%82%D0%B0%D0%B5%D1%82%20,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2) [[94]](file://file_0000000004006243943cbb9888dd4b65#:~:text=,6) [[95]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20%D0%92%D0%B2%D0%B5%D1%81%D1%82%D0%B8%20,%D1%81%20%D0%BF%D0%BE%D0%B2%D1%8B%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%80%D0%B0%D1%81%D1%85%D0%BE%D0%B4%D0%BE%D0%BC%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2) [[96]](file://file_0000000004006243943cbb9888dd4b65#:~:text=11%29%20%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%3A%20%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8%20%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20,%D0%B0%D1%80%D0%BC%D0%B8%D0%B9%3B%20%D0%BF%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%B4%D1%91%D1%82%20%D0%BF%D0%BE%20%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E) [[97]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B0%D1%80%D0%BC%D0%B8%D0%B9%3B%20%D0%BF%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%B4%D1%91%D1%82%20%D0%BF%D0%BE%20%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E) [[98]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D0%B0%D1%80%D0%BC%D0%B8%D0%B9%3B%20%D0%BF%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%B4%D1%91%D1%82%20%D0%BF%D0%BE%20%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E) [[99]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87) [[100]](file://file_0000000004006243943cbb9888dd4b65#:~:text=13%29%20%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%20%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20,%D1%83%D0%B7%D0%BA%D0%B8%D0%B5%20%D0%BC%D0%B5%D1%81%D1%82%D0%B0%20%D0%BF%D0%BE%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%83%2F%D1%83%D0%B7%D0%BB%D1%83) [[101]](file://file_0000000004006243943cbb9888dd4b65#:~:text=12%29%20%D0%A4%D0%BE%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8%3A%20%D0%B4%D0%B5%D1%82%D0%B0%D0%BB%D0%B8%20%28SHOULD%29%201,%D0%BF%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D1%8B%D0%B5%20%E2%80%94%20%D0%BC%D0%B5%D0%B4%D0%BB%D0%B5%D0%BD%D0%BD%D0%B5%D0%B5%20%D0%B8%20%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%BE%D0%BF%D1%80%D0%B8%D0%B3%D0%BE%D0%B4%D0%BD%D1%8B) [[102]](file://file_0000000004006243943cbb9888dd4b65#:~:text=3.%20%D0%98%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B%20%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0%3A%20,%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F%20%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%BC%20%D0%B1%D1%80%D0%B8%D0%B3%D0%B0%D0%B4%2F%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0) [[103]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D0%BC%D0%B5%D1%81%D1%82%D0%BE%20%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D1%8F.%204.%20,%D0%B2%20%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8%2C%20%D0%BE%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87) [[135]](file://file_0000000004006243943cbb9888dd4b65#:~:text=%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B8%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%28DoD%29%20,%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B9%2C%20%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%BD%D1%8B%D0%B5%20%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B8%20%D0%B4%D0%B5%D1%84%D0%B8%D1%86%D0%B8%D1%82%D0%B0%D1%85) [[142]](file://file_0000000004006243943cbb9888dd4b65#:~:text=1.%20,%D1%87%D0%B0%D1%81%D1%8B%20%D0%B8%20%D0%B5%D0%B4%D1%83%20%D0%BD%D0%B0%20%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B8) [[144]](file://file_0000000004006243943cbb9888dd4b65#:~:text=14%29%20%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9%20%D1%81%D1%80%D0%B5%D0%B7%20,%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8) [[145]](file://file_0000000004006243943cbb9888dd4b65#:~:text=14%29%20%D0%9C%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9%20%D1%81%D1%80%D0%B5%D0%B7%20,%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8) document-4.txt

<file://file_0000000004006243943cbb9888dd4b65>

[[76]](https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai#:~:text=Something%20that%20I%20just%20thought,called%20the%20influence) [[77]](https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai#:~:text=So%20for%20example%2C%20if%20you,the%20points%20of%20highest%20influence) android - Help with Strategy-game AI - Game Development Stack Exchange

<https://gamedev.stackexchange.com/questions/7251/help-with-strategy-game-ai>

[[105]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=breaking%20algorithm%20which%20speeds%20up,including%20abstraction%20and%20memory%20heuristics) [[106]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=considered,existing%20speedup%20techniques%20%E2%80%94%20including) [[109]](https://harablog.wordpress.com/2011/09/07/jump-point-search/#:~:text=2.%20It%20involves%20no%20pre,%5B1) Jump Point Search | Shortest Path

<https://harablog.wordpress.com/2011/09/07/jump-point-search/>

[[107]](https://jair.org/index.php/jair/article/view/12255#:~:text=Regarding%20Goal%20Bounding%20and%20Jump,based%20optimal%20pathfinding) Regarding Goal Bounding and Jump Point Search

<https://jair.org/index.php/jair/article/view/12255>

[[110]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=Hierarchical%20Path,clusters%2C%20the%20maximal%20sized%20entrances) [[111]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=3,run%20can%20be%20used) [[112]](https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf#:~:text=the%20nodes%20in%20the%20abstract,graph%20instead%20of%20the) cs.ru.nl

<https://www.cs.ru.nl/bachelors-theses/2013/Linus_van_Elswijk___0710261___Hierarchical_Path-Finding_Theta_star_Combining_HPA_star_and_Theta_star.pdf>

[[131]](https://docs.unity3d.com/Packages/com.unity.test-framework@1.0/manual/edit-mode-vs-play-mode-tests.html#:~:text=Manual%20docs,Editor%20and%20test%20game%20code) Edit Mode vs. Play Mode tests | Test Framework - Unity - Manual

<https://docs.unity3d.com/Packages/com.unity.test-framework@1.0/manual/edit-mode-vs-play-mode-tests.html>

[[133]](https://samuel-asher-rivello.medium.com/best-practices-6-ci-cd-with-unity-buildalon-be3286f05274#:~:text=In%20today%E2%80%99s%20fast,work%20and%20catch%20issues%20early) Best Practices — 6 — CI/CD With Unity Buildalon | by Samuel Asher Rivello | Medium

<https://samuel-asher-rivello.medium.com/best-practices-6-ci-cd-with-unity-buildalon-be3286f05274>

[[134]](https://unity.com/topics/what-is-ci-cd#:~:text=CI%20is%20a%20DevOps%20best,can%20snowball%20into%20bigger%20issues) What is CI/CD? Continuous Integration & Delivery Explained | Unity

<https://unity.com/topics/what-is-ci-cd>