# Дополнение к проекту: Государства, Дипломатия, Редкости Исследований, Аномалии, Автоматизация, Экономика, Логистика и Фортификации

Версия: 1.0 • Ядро: Unity (C#) • Архитектура: Domain/Data/Application/AI/Presentation/Infrastructure  
Совместимость: off‑screen LOD‑симуляция, EventBus, сохранения JSON, Pathfinding

Цель: объединить в единый план архитектуры все подсистемы 2D RTS с паузой – включительно несколько государств с дипломатией (посольства/союзы/слияние), расширенную экономику (двойной бюджет государства и игрока), логистику перемещений и фортификации, пересобранное древо исследований (обычные/редкие/аномальные), симуляцию заражения, аномалии и автоматизацию процессов через серверы/ядра. Все новые механики вписаны в существующую архитектуру (событийная модель, off-screen симуляция, сохранения, ScriptableObject-конфиги).

## 0) Коротко (TL;DR)

* Государства (States): старт за выбранное государство; уровни отношений → Нет контакта → Посольство → Союз → (возможное) Слияние.
* Видимость/интел:
* Нет контакта или вражда — AI не “видит” юниты/постройки другого государства (свой туман войны).
* Посольство — видны юниты/постройки, без экономики/очередей задач.
* Союз — видно всё (вкл. экономику и очереди), но каждый контролирует только свои войска.
* Слияние: ежедневно малая вероятность p (растёт со временем союза); при слиянии два государства объединяются под контролем одного AI; игрок остаётся лидером своей стороны.
* Экономика: введено 4 базовых ресурса (труда-часы, энергия, материалы, еда) с суточным циклом производства и потребления. Двойной бюджет: часть прироста автоматически резервируется под задачи гос-ИИ (настройка процентами), остальное в распоряжении игрока. Дефициты ресурсов снижают эффективность по приоритетам (отключение энергии, голод – падение прироста населения и пр.).
* Логистика: система узлов и коридоров снабжения (город, станция, порт, аэродром + дороги, ж/д, авиамаршруты). Игрок отдаёт приказы переброски (“откуда–куда”, приоритет, цель прибытия), юниты перемещаются автоматически. Время в пути рассчитывается с учётом пропускной способности; есть режим “рывок” (+50% скорость за повышенный расход). Узкие места и очереди на маршрутах визуализируются, поддерживается авто-маршрутизация подкреплений.
* Фортификации: юниты-инженеры могут возводить укрепления — временные (окоп, мешки с песком, наблюдательный пункт) и постоянные (ДОТ, бункер, рвы). У каждой позиции есть стоимость (труд, материалы, иногда энергия), время постройки, уровень защиты и износ. Временные позиции деградируют со временем и под огнём; постоянные – медленнее и поддаются ремонту. Доступны инструменты строительства линий (траншеи по полилинии) и узловых точек; возможно улучшение временных позиций до постоянных.
* Исследования: древо по категориям с “видимым одним шагом” вперёд; редкость исследований: Обычные / Редкие / Аномальные. Шанс редкого зависит от качества исследований (скрытый агрегатный показатель зданий науки). Стоимость шагов растёт на 10–15% за каждый завершённый шаг (инфляция прогресса).
* Здания науки: два типа апгрейдов — количественный (повышение скорости исследований, RP) и качественный (повышение качества, QP). Игрок не знает точного шанса появления редких исследований, но получает косвенные индикаторы уровня качества.
* Аномалии: особые технологии доступны только при наличии содержания заражённых образцов и команд по их поимке. Образцы расходуются в исследованиях; для прогресса по аномалиям требуется налаженный цикл автоматизации “поимка → доставка образцов → изучение”.
* Автоматизация: специальные серверные здания дают вычислительные единицы (CU); ядро автоматизации берёт под контроль один процесс (отряд или здание), потребляя много энергии и CU. Ядро полностью автоматизирует выбранный цикл (например, выполняет исследования или управляет отрядом без участия игрока).

## 1) Domain (модели и правила)

### 1.1 Сущности (Entities)

public enum IntelAccess { None, UnitsAndBuildings, Full } // уровни доступа к данным другого гос-ва  
public enum RelationKind { None, Hostile, Embassy, Alliance } // стадия отношений между гос-вами  
public enum ResearchRarity { Common, Rare, Anomalous } // тип исследования по редкости  
public enum ResourceType { Labor, Power, Materials, Food } // базовые ресурсы экономики  
  
public sealed record StateId(string Value);  
public sealed record SectorId(int X, int Y);  
  
public sealed class State  
{  
 public StateId Id;  
 public string Title;  
 public Dictionary<StateId, Relation> Relations = new();  
 public IntelLayer Intel; // знание о мире (свой FoW и обнаруженные объекты)  
 public EconomySnapshot Economy; // запасы ресурсов и бюджеты  
 public ResearchProgress Research; // прогресс по веткам исследований  
 public AutomationNetwork Automation; // доступные CU и привязанные ядра  
 // ... (прочие подсистемы: очереди логистики, строительство фортификаций и пр.)  
}  
  
public sealed class Relation  
{  
 public RelationKind Kind; // None/Hostile/Embassy/Alliance  
 public float DaysInAlliance; // накапливается, только когда Kind=Alliance  
 public float Stability; // 0..1 (доверие, история, общие цели)  
}  
  
public sealed class IntelLayer // "карта знаний" государства о других  
{  
 // Права доступа к данным другого гос-ва:  
 public IntelAccess AccessTo(StateId other);  
 // Получить известные данные (с учётом FoW):  
 public KnownUnits GetKnownUnits(SectorId sector, StateId other);  
 public KnownBuildings GetKnownBuildings(SectorId sector, StateId other);  
 public KnownEconomy GetKnownEconomy(StateId other); // экономика/очереди (только при Full)  
}  
  
// Прогресс исследований данного государства:  
public sealed class ResearchProgress  
{  
 public Dictionary<string, Branch> Branches; // активные ветки (например "economy.extraction")  
 public int TotalSteps; // сколько шагов исследования завершено (для инфляции стоимости)  
 public float CostMultiplier; // текущий мультипликатор стоимости (=(1+r)^TotalSteps)  
 public float QualityScore; // скрытый агрегат (отношение QP/RP от зданий науки)  
}  
  
public sealed class Branch  
{  
 public string Id; // например "economy.extraction"  
 public int Completed; // сколько шагов в ветке пройдено  
 public ResearchDef Current; // текущий (единственный видимый) исследуемый шаг  
}  
  
public sealed class ResearchDef  
{  
 public string Id;  
 public string BranchId;  
 public ResearchRarity Rarity; // Common/Rare/Anomalous  
 public float BaseCostRP; // базовая стоимость (в RP)  
 public string[] PrereqIds; // требования (предыдущие исследования)  
 public string[] RequiresAssets; // требуемые объекты для разблокировки (напр. ContainmentLab, CaptureTeam)  
}  
  
public sealed class AutomationNetwork  
{  
 public float TotalComputeUnits; // суммарно доступные CU (от всех ServerFarm)  
 public List<AutomationCore> Cores;  
 public void AttachCore(AutomationCore core, IAutomatable target);  
 // ...  
}

### 1.2 Формулы и правила (Diplomacy, Economy, Infection, Logistics, Fortification)

* Intel‑доступ:  
  None → (ничего не видно);  
  Embassy → Units+Buildings (разрешено видеть дислокацию/типы/прочность юнитов и построек союзника, без экономики и очередей задач);  
  Alliance → Full (полный доступ ко всем данным, включая экономику, прогрессы исследований, очереди строительства).  
  Примечание: уровень доступа влияет только на осведомлённость, но не даёт контроля над чужими войсками или строениями.
* Шанс слияния (ежедневный) – если государства находятся в союзе:
* p\_merge(day) = clamp( p0 + a \* ln(1 + DaysInAlliance/7) + b \* Stability - c \* Strain, 0, p\_max )
* Где: p0=0.0015 (0.15%), a≈0.002, b≈0.005, c≈0.003, p\_max=0.12.  
  Статистика: DaysInAlliance растёт каждый день в союзе; Stability (0..1) повышается при успешных совместных операциях и малых потерях; Strain – штраф за конфликты интересов и дефициты ресурсов. При успешном ролле слияния происходит объединение: одно государство поглощается другим, передаются территории, юниты, ресурсы, очереди (конфликты очередей решаются по приоритетам).
* Инфляция стоимости исследований: после каждого завершённого шага (глобально или в рамках ветки – настройка баланса) стоимость следующего увеличивается:
* Cost(next) = BaseCost \* (1 + r)^(i),
* где i – порядковый номер шага для данного гос-ва (или ветки); r – ставка инфляции (обычно 0.10–0.15 для Common, выше для Rare/Anomalous).
* Редкость следующего шага: после завершения текущего исследования вычисляется новый текущий шаг ветки:
* pRare = clamp( pRareBase + α \* QualityScore, 0, pRareMax )  
  if (Roll(pRare)) select Rare else select Common.  
  // Anomalous доступен, если выполнены все RequiresAssets (напр. ContainmentLab, CaptureTeam)   
  // и выполнен независимый Roll(pAnom), где pAnom << pRare и также зависит от QualityScore и накопленной статистики образцов.
* QualityScore – скрытый показатель качества исследований: QS = Σ(QP) / (Σ(RP) + ε), где RP (research points) и QP (quality points) дают научные здания. Высокий QS повышает шанс редких исследований. Аномальные технологии становятся возможны только при наличии спец. инфраструктуры и достаточного числа заражённых образцов.
* Экономика – ресурсы и суточный цикл: введены четыре базовых ресурса: трудовые часы населения, энергия (производство/потребление за день), материалы (стройматериалы, боекомплект) и еда (пропитание гражданских и военных). Экономический тик происходит раз в игровой день по фиксированному порядку (см. ниже). Для каждого здания заданы: стоимость постройки, время строительства, штатная потребность в труде, потребление/выработка энергии, ежедневное обслуживание и прирост/переработка ресурсов. Запрет отрицательных запасов: система автоматически снижает неприоритетные расходы, если ресурс на нуле (подробности – см. дефициты и приоритеты).
* Прирост населения и рекрутинг: каждый день рассчитывается прирост населения CivGrowth (чел./день) для каждого государства. Прирост добавляет в пул свободных гражданских; их трудовые часы вычисляются как LaborHours = CivFree \* HoursPerCitizen (константа, напр. 8 ч/день). Через постройки типа Казармы или Призывной пункт часть свободных гражданских можно конвертировать в военных: у таких построек есть лимит новобранцев в день, время подготовки и стоимость подготовки (еда, материалы, энергия, труд). По завершении обучения новый военный юнит поступает в резерв/гарнизон.
* Двойной бюджет (гос-ИИ и игрок): вводится механика передачи части ресурсов от игрока под управление стратегического государственного AI. Для каждого ресурса (включая CivGrowth) у игрока есть слайдер процента передачи в диапазоне 0–100%. Выбранный процент от суточного прироста данного ресурса автоматически передаётся в распоряжение гос-ИИ (до поступления на склад игрока). Округляется вниз до целых единиц; остаток получает игрок. Эти настройки можно задать глобально и при необходимости переопределить для каждого AI-союзника отдельно. Гос-ИИ расходует свой бюджет на приоритетные задачи (оборона, карантины, базовые исследования, поддержание экономики и логистики), автоматически перераспределяя средства по Utility-оценкам важности. Бюджет игрока используется им напрямую для точечных операций (ускоренные постройки, заказы технологий, возведение укреплений, нанятие отрядов). Внутри государства могут существовать механики налогов, субсидий, займов под проценты, лимитов капитальных/операционных расходов и т.п. для балансировки двух потоков бюджета.
* Контракты и штрафы: отношения между игроком и гос-ИИ в экономике оформляются как контракты – например, на строительство объекта, проведение исследования или ликвидацию очага инфекции. Контракт – это сущность с жизненным циклом (заключён, выполняется, выполнен/сорван). Сорванные контракты влекут штрафы: уменьшение доверия/стабильности, разовые вычеты из бюджета или падение рейтинга игрока. Выполненные – повышают доверие и могут приносить бонусы (бонус к финансированию, перки).
* Модель инфекции: выбрана гибридная схема – сочетание классической эпидемиологической модели (SIR/SEIR) с клеточным автоматом на мировой сетке. Заражение распространяется по ячейкам (соседство по сети ячеек – фиксированное Moore или Von Neumann)[[1]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D0%BB%D0%B8%D0%BC%D0%B8%D1%82%D1%8B%20%D0%BA%D0%B0%D0%BF%D0%B5%D0%BA%D1%81%D0%B0%2F%D0%BE%D0%BF%D0%B5%D0%BA%D1%81%D0%B0%29,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)[[2]](file://file_000000007c6861f4aefcba665986a990#:~: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), с диффузией влияния от каждой заражённой клетки. Параметры SIR (скорости заражения, восстановления, смертности) настраиваемые и могут меняться, например, при исследованиях вакцины. Распространение усиливается или сдерживается инфраструктурой и действиями: сооружение санитарных кордонов и карантинных зон снижает мобильность и вероятность заражения (но сокращает производство в этих зонах); наличие транспортных путей (дороги, железные дороги, авиа) может ускорять распространение инфекции между отдалёнными регионами. Выделяются “очаги” инфекции – кластеры клеток с высокими показателями заражённости; образуется динамическая “линия фронта” между заражёнными и чистыми территориями. Гос-ИИ ставит целью сдерживать фронт на границах очагов, а игрок может финансировать и проводить операции прорыва (очистки) в ключевых точках.
* Оптимизация инфекции: обновляются только “активные” клетки (фронтир) вместо всей сетки[[2]](file://file_000000007c6861f4aefcba665986a990#:~: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)[[3]](file://file_000000007c6861f4aefcba665986a990#:~: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). Реализация диффузии – в два шага: (1) сбор влияния соседей для всех активных клеток, (2) применение обновлений (используется двойной буфер для сетки, чтобы расчёты одного тика не влияли на него же). Возможен перенос тяжёлых вычислений инфекции в Unity Jobs (параллельно по чанкам IJobParallelFor, хранение полей заражения в структурах NativeArray<float> с Burst-компиляцией)[[2]](file://file_000000007c6861f4aefcba665986a990#:~: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)[[3]](file://file_000000007c6861f4aefcba665986a990#:~: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), либо использование ECS (системы в группе FixedStepSimulationSystemGroup с хранением сетки в компонентах). База данных соседей (для Moore/Neumann в сетке) фиксирована для воспроизводимости. Для визуализации и отладки — накладывается цветовая карта заражения (heatmap), позволяющая наблюдать прогресс и обнаруживать аномалии в расчётах.
* Логистика (транспорт): введён класс объектов LogisticNode (тип: город, станция, порт, аэродром и пр.) и связи LogisticLink (дорога, ж/д путь, авиамаршрут) между узлами. Игрок не управляет непосредственно транспортом юнитов, а создаёт приказы переброски: указывает отправной узел, пункт назначения, приоритет и желаемую группу/цель прибытия. Система логистики автоматически рассчитывает кратчайший по времени путь по сети узлов, учитывая пропускную способность звеньев и текущие загруженности. Юниты (или ресурсы) перемещаются в фоновом режиме, не требуя постоянного микроконтроля. Реализуется очередь на маршрутах: если пропускная способность звена ограничена, транспорт ожидает. Поддерживается режим ускоренной переброски (Rush) – юниты могут двигаться на ~50% быстрее ценой повышенного расхода материалов и энергии[[4]](file://file_0000000049b062438e06b107048784b7#:~: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). Для стабильной работы логистики предусматриваются механизмы авто-маршрутизации: игрок может обозначить для армий “домашние” базы (узлы), и тогда пополнение войск идёт автоматически по заданным маршрутам. При изменении сети (разрушены узлы, перекрыты дороги) — система частично инвалидирует кэш маршрутов и пересчитывает пути только затронутых направлений. Для эффективности маршрутизации на большой карте используется иерархическая навигация (например, алгоритм HPA: кластеризация сетки и вычисление путей между «порталами» кластеров)[[5]](file://file_000000007c6861f4aefcba665986a990#:~:text=9%29%20%D0%9F%D1%83%D1%82%D1%8C%20%D1%8E%D0%BD%D0%B8%D1%82%D0%B0%20,%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). Внутри кластера применяется быстрый алгоритм поиска (A с Jump Point Search ускорением для равномерной сетки)[[6]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%BF%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B8%D0%BD%D0%B3%D0%B0%3B%20%D0%B5%D1%81%D1%82%D1%8C%20%D1%83%D0%BB%D1%83%D1%87%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20JPS%2B%20%EE%88%80cite%EE%88%82turn1search3%EE%88%82turn1search11%EE%88%81). После прокладки маршрута выполняется сглаживание пути (string-pulling) для удаления лишних углов и избегания «красных зон» (например, заражённых секторов, если есть альтернатива).
* Фортификации: боевые инженерные юниты могут строить полевые укрепления. Укрепления бывают временные (окопы, мешки с песком, наблюдательные пункты) – возводятся быстро из подручных материалов, дают небольшой бонус к защите, но постепенно ветшают; и постоянные (ДОТы, бункеры, противотанковые рвы) – требуют больше ресурсов и времени, но обеспечивают серьёзную защиту и служат долго (нуждаются лишь в ремонте). Каждой постройке задана стоимость в ресурсах (трудо-часы, материалы, иногда энергия), необходимое время строительства, уровень защиты/укрытия, а также показатели деградации со временем и под обстрелом. Реализованы инструменты игрока для быстрого размещения фортификаций: линейное строительство (рисование линии траншеи произвольной формы) – система автоматически превращает линию в последовательность сегментов окопа с суммарной стоимостью пропорционально длине; узловые точки – отдельные укрепления (точки обороны) размещаются поодиночке; апгрейд позиции – возможность улучшить существующую временную позицию до постоянной (учитывается часть уже вложенных трудо-часов и материалов). Строительство выполняют доступные инженерные подразделения; несколько подразделений могут работать параллельно над разными задачами, либо над разными сегментами одной длинной траншеи. Временные укрепления со временем теряют эффективность (прочность, бонусы), особенно под огнём; постоянные – тоже, но медленнее, и могут быть восстановлены ремонтом.

### 1.3 Автоматизация (Automation)

public interface IAutomatable   
{   
 bool StartProcess(); // запустить процесс (если доступен ресурс)  
 bool IsRunning { get; } // сейчас выполняется  
 bool IsStalled { get; } // приостановлен (дефицит ресурсов)  
 bool CompleteStep(); // выполнить атомарный шаг процесса (например, один тик исследования или цикл постройки)  
}  
  
public sealed class AutomationCore   
{  
 public Guid Id;  
 public IAutomatable CurrentTarget; // текущий автоматизируемый объект (юнит/здание)  
 public bool IsActive;  
 public float EnergyConsumption; // потребление MW  
 public float ComputeConsumption; // потребление CU  
 // ...  
}

Принцип: ядро может быть привязано к одному объекту (например, отряду или лаборатории) и берёт на себя выполнение его операций. Для работы ядро требует большого расхода энергии и ComputeUnits (CU). Если ресурсов не хватает, ядро ставит процесс на паузу (состояние Stalled). Пример: ядро автоматизации прикреплено к лаборатории исследований – тогда все исследования в этой лаборатории выполняются автоматически без команд игрока; либо ядро к отряду захвата образцов – отряд сам по себе выполняет цикл поимки заражённых. Игрок ограничен числом имеющихся ядер и общим количеством вычислительных ресурсов, поэтому должен выбирать, какие задачи автоматизировать.

## 2) Data (конфиги, JSON, SO)

### 2.1 Примеры JSON конфигурации (упрощённо)

States & Relations (государства и отношения):

{  
 "states": [  
 { "id": "player", "title": "Союз Горного Края" },  
 { "id": "red", "title": "Красная Коммуна" },  
 { "id": "blue", "title": "Северный Консорциум" }  
 ],  
 "relations": [  
 { "a": "player", "b": "red", "kind": "Hostile", "stability": 0.2 },  
 { "a": "player", "b": "blue", "kind": "None", "stability": 0.0 }  
 ]  
}

ResearchDefs (исследования):

{  
 "research": [  
 { "id": "eco.extraction.1", "branch": "economy.extraction", "rarity": "Common",  
 "baseCostRP": 100, "prereq": [] },  
 { "id": "eco.extraction.R1","branch": "economy.extraction", "rarity": "Rare",  
 "baseCostRP": 160, "prereq": ["eco.extraction.1"] },  
 { "id": "anom.capture.1", "branch": "anomaly.capture", "rarity": "Anomalous",  
 "baseCostRP": 220, "requiresAssets": ["ContainmentLab","CaptureTeam"] }  
 ]  
}

Buildings (здания науки, аномалий, серверы, ядра):

{  
 "buildings": [  
 { "id": "Lab\_T1", "type": "Science", "rp": 1.0, "qp": 0.1, "power": 0.5 },  
 { "id": "Lab\_T2", "type": "Science", "rp": 2.2, "qp": 0.2, "power": 0.8,  
 "upgradeOf": "Lab\_T1" },  
 { "id": "Lab\_Spec", "type": "Science", "rp": 1.0, "qp": 1.4, "power": 1.2 },  
 { "id": "ContainmentLab","type": "Containment", "capacity": 20, "power": 1.5 },  
 { "id": "ServerFarm\_T1", "type": "Server", "computeUnits": 20, "power": 3.0 },  
 { "id": "AutomationCore","type": "Core", "consumeCU": 10, "power": 2.5 }  
 ]  
}

Economy & Logistics (экономика, логистика, фортификации):

{  
 "economy": {  
 "dailyTickHour": 0,  
 "transferToAI": { "laborPct": 20, "powerPct": 0, "materialsPct": 0, "foodPct": 0, "civGrowthPct": 0 }  
 },  
 "population": {  
 "civilians": 10000,  
 "baseDailyGrowth": 45,  
 "hoursPerCivPerDay": 8  
 },  
 "buildings": [  
 { "id": "farm\_t1", "type": "economic", "buildTimeDays": 3, "cost": {"materials":120},  
 "powerIn": 5, "laborIn": 10, "foodOut": 60 },  
 { "id": "barracks\_t1", "type": "unit\_prod", "buildTimeDays": 4, "cost": {"materials":200},  
 "powerIn": 10, "laborIn": 20,  
 "convert": { "civPerDay": 20, "trainDays": 2, "costPerRecruit": {"food": 2, "materials": 1} } }  
 ],  
 "logistics": {  
 "nodes": [  
 { "id": "city\_A", "type": "city" },  
 { "id": "front\_N", "type": "front" }  
 ],  
 "links": [  
 { "from": "city\_A", "to": "front\_N", "mode": "rail", "capacityUnitsPerDay": 50, "baseHours": 6 }  
 ]  
 },  
 "fortifications": [  
 { "id": "foxhole", "category": "temporary", "labor": 4, "materials": 2, "buildHours": 6, "cover": "light" },  
 { "id": "bunker", "category": "permanent", "labor": 40, "materials": 120, "powerIn": 2, "buildDays": 5, "cover": "heavy" }  
 ]  
}

Примечание: приведённые JSON-фрагменты иллюстрируют возможную структуру данных. В реальном проекте конфигурации могут храниться как ScriptableObject-ы и/или JSON в Addressables.

### 2.2 ScriptableObject каталоги и сохранения

* Каталоги (Assets): StateDef.asset, ResearchDef.asset, BuildingDef.asset, RecipeDef.asset, UnitDef.asset, InfectionParams.asset и др. – хранят статические данные игровых сущностей. При старте они используются для создания runtime-моделей (plain C# объекты), которые используются в симуляции (оригиналы SO остаются неизменными)[[7]](file://file_000000007c6861f4aefcba665986a990#:~: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).
* Версионирование сохранений: введён идентификатор версии SaveVersion (целый), сохраняемый вместе с данными. При загрузке проверяется версия; если она старее текущей – выполняются миграции (добавление новых полей с дефолтами, пересчёт устаревших параметров и т.д.)[[8]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%EE%88%80cite%EE%88%82turn2search10%EE%88%81). Старые файлы thus автоматически поддерживаются после обновлений.
* Секции сохранения: помимо базовых разделов (юниты, постройки, карта) сохраняются новые подсистемы: states[] (список государств с их ресурсами), relations[] (отношения), researchProgress (прогресс по веткам и QS), automation (сеть серверов и ядра), containment (содержимое лабораторий по аномалиям). В составе данных государств сохраняются DaysInAlliance, Stability по союзам, текущие ресурсы, очереди логистики, фортификаций и пр. Все случайные величины (роллы слияния, выпадение редких исследований) должны быть детерминированы через сохраняемый randomSeed[[9]](file://file_0000000014b86246bd27d0391214fa20#:~:text=3.3%20%D0%A1%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20,%D0%B4%D0%BB%D1%8F%20%D0%B1%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%B2%20%D1%80%D0%B5%D0%B4%D0%BA%D0%BE%D1%81%D1%82%D0%B8%2F%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F), чтобы при повторном прогоне результаты совпадали.

## 3) Application (циклы, события, системы)

### 3.1 EventBus – ключевые события

Основные события, генерируемые системами в ходе игры, для подписки логики UI, логирования и скриптов:

Diplomacy/RelationsChanged(StateId a, StateId b, RelationKind newKind)  
Diplomacy/MergeHappened(StateId winner, StateId absorbed)  
Intel/AccessChanged(StateId viewer, StateId target, IntelAccess newAccess)  
Research/StepCompleted(StateId state, string branchId, string defId, ResearchRarity rarity)  
Research/NextStepAssigned(StateId state, string branchId, string defId, ResearchRarity rarity)  
Anomaly/SamplesRequired(StateId state, int amount) // требуются образцы (для аномалии)  
Anomaly/Breach(StateId state, SectorId sector, int severity) // утечка/прорыв заражённых  
Automation/CoreAttached(Guid coreId, string targetId) // ядро автоматизации привязано к цели  
Automation/CoreStalled(Guid coreId, string reason) // пауза работы ядра (дефицит ресурсов)  
Power/GridOverload(StateId state, float deltaMW) // нехватка энергии в сетке (отключения)  
Logistics/TransferQueued(int transportId, string fromNodeId, string toNodeId)  
Logistics/TransferCompleted(int transportId)

(События сгруппированы по тематике. Кроме того, используется пакетирование повторяющихся событий: при всплеске множества однотипных событий система может склеивать их в одно – например, вместо 50 раздельных событий заражения ячеек за тик выдать один батч "InfectionSpread" списком секторов.)\*

### 3.2 Simulation Loop – игровой цикл и тайминг

* Фиксированный тик: игровая симуляция работает с фиксированной длительностью шага (например, 0.5 секунды реального времени). Реализация – собственный цикл SimulationLoop с использованием накопителя времени (unscaledDeltaTime)[[10]](file://file_000000007c6861f4aefcba665986a990#:~: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). Это обеспечивает независимость симуляции от фреймрейта и корректную паузу. Пауза реализована либо через флаг (при IsPaused все системные тик-апдейты пропускаются), либо глобально Time.timeScale = 0 – в последнем случае нужно убедиться, что вся UI-анимация и таймеры используют unscaled время[[10]](file://file_000000007c6861f4aefcba665986a990#:~: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). Также поддерживаются режимы скорости: ×0.5 / ×1 / ×2 / ×4 – изменение скорости масштабирует unscaledDeltaTime для симуляционных систем. Дополнительно, предусмотрена кнопка «Шаг» – выполнить вручную один тик при паузе[[11]](file://file_000000007c6861f4aefcba665986a990#:~:text=2,%D1%86%D0%B8%D0%BA%D0%BB%D1%83) (для отладки или пошаговой игры).
* Порядок систем на тик:
* Diplomacy (обработка заявок на отношения, расторжений)
* Intel (обновление уровней доступа, FoW слоёв)
* Research (генерация очков, проверка завершения шага)
* Automation (работа активных ядер, прогресс автоматизированных процессов)
* Economy/Production (суточный расчёт ресурсов и производство, если настал дневной тик)
* Infection/Quarantine (распространение инфекции, эффекты карантинов)
* Logistics (обновление статусов перебросок, прибытие/убытие юнитов)
* Combat/Anomaly (боевые системы, события аномалий)
* EventBus.Flush() (после завершения тика – отправка накопленных событий слушателям)[[12]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=6%29%20%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%3A%20,%EE%88%80cite%EE%88%82turn0search12%EE%88%81)[[13]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%2F%D1%81%D0%B5%D0%B9%D0%B2%D1%8B).

Примечания: Экономика и ряд систем обновляются раз в игровые сутки (может быть реализовано счетчиком тик-циклов или отдельным таймером). При паузе игровые системы не обновляются, однако UI и анимации работают (на unscaled time). Off-screen LOD: для отдалённых секторов или объектов стратегические системы (Diplomacy, Research, Automation, Infection) могут тикать с пониженной частотой[[14]](file://file_0000000014b86246bd27d0391214fa20#:~:text=,%D0%B4%D0%BB%D1%8F%20%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85%20%D1%81%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2%20%28%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%82%D1%8B) или в агрегированном режиме (например, раз в 10 тиков), чтобы снизить нагрузку. При большом количестве объектов алгоритмы вроде подсчёта влияний, обновления экономики или поиск пути могут распределяться на несколько кадров (cooperative multi-threading) или запускаться асинхронно в потоках/Jobs.

* Команды и задачи: взаимодействие игрока с игрой (построить здание, отдать приказ отряду, запустить исследование) реализовано через команды, которые помещаются в очередь и применяются либо сразу, либо в безопасной точке цикла. В ряде случаев поддерживается отмена (undo) команды, пока она не выполнена (например, можно снять с постройки здание, удалив команду до её исполнения). Все команды за тик логируются; при сохранении состояния важен тик-бариер – сохранение выполняется после полного завершения тика (и применения команд), чтобы не было «полуприменённых» действий[[15]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn1search6%EE%88%81).

### 3.3 Системы Pathfinding

* Pathfinding: Для поиска пути юнитов на сетке используется A с улучшениями. На малых дистанциях и в локальных боях работает Jump Point Search (JPS) – оптимизация A для равномерной сетки, пропускающая тривиальные промежуточные узлы[[6]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%BF%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B8%D0%BD%D0%B3%D0%B0%3B%20%D0%B5%D1%81%D1%82%D1%8C%20%D1%83%D0%BB%D1%83%D1%87%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20JPS%2B%20%EE%88%80cite%EE%88%82turn1search3%EE%88%82turn1search11%EE%88%81). На уровне всей карты применяется иерархический A (алгоритм HPA)[[5]](file://file_000000007c6861f4aefcba665986a990#:~:text=9%29%20%D0%9F%D1%83%D1%82%D1%8C%20%D1%8E%D0%BD%D0%B8%D1%82%D0%B0%20,%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) – сетка разделяется на области (кластеры), считаются порталы между ними; первый этап поиска строит маршрут по графу кластеров, второй этап – подробный путь внутри кластеров. Такой подход резко ускоряет поиск на больших картах. Поиск пути выполняется асинхронно (в Job-потоках) и позволяет пакетировать запросы – несколько запросов могут обрабатываться параллельно[[16]](file://file_000000007c6861f4aefcba665986a990#:~: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). При изменениях карты (появилось здание, карантин) система помечает только затронутые области как требующие обновления (partial invalidation)[[16]](file://file_000000007c6861f4aefcba665986a990#:~: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). После построения маршрута производится сглаживание траектории, а также накладываются запреты на движение через опасные зоны (например, через заражённые клетки или через чужие территории, если нет разрешения) – при наличии запрета система выберет альтернативный маршрут либо выдаст предупреждение, если путь невозможен.
* Debug & Visualize: Для отладки алгоритма пути предусмотрено отображение: при расчёте пути можно визуализировать проход алгоритма – отображать открытые и закрытые узлы, итоговый маршрут (например, с помощью OnDrawGizmos или линий)[[17]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn3search5%EE%88%82turn3search20%EE%88%81). Это помогает убедиться, что алгоритм работает корректно и эвристика согласована с расстоянием (иначе путь будет suboptimal или не найден).

### 3.4 Сохранения и репликация

* JSON-сохраняемость: игровые модели структурированы так, чтобы быть сериализуемыми в JSON с помощью JsonUtility. Следует избегать несериализуемых полей (например, Dictionary без кастомного конвертера)[[18]](file://file_000000001f0461f4847b90af8dfa8700#:~: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) и ссылок на ScriptableObject в полях состояния. Полиморфные поля (напр. интерфейсы) требуют атрибута [SerializeReference] и проверки, что ToJson/FromJson корректно восстанавливает подтипы[[18]](file://file_000000001f0461f4847b90af8dfa8700#:~: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). При сложных случаях возможен переход на стороннюю JSON-библиотеку, но базовый подход – плоские структуры данных. Сохранения больших массивов (например, состояние сетки заражения) допустимо переводить в бинарный формат для компактности (с обёрткой в Base64 в JSON либо отдельным файлом).
* Addressables (память): все крупные ассеты (спрайты, звуки) загружаются через Addressables. Важно тщательно отслеживать выгрузку: после использования объектов необходимо вызывать Addressables.Release – иначе объекты останутся в памяти[[19]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=1%29%20MonoBehaviour,%EE%88%80cite%EE%88%82turn0search12%EE%88%82turn0search20%EE%88%82turn0search24%EE%88%81)[[20]](file://file_000000007c6861f4aefcba665986a990#:~: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). Рекомендуется вести счётчики загруженных ресурсов для отладки, а также группировать ассеты в Addressables-группы по используемому режиму (для разных стратегий кеширования и подгрузки)[[21]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%BC%D0%B8%20%D0%B4%D0%BE%D0%BB%D0%B6%D0%BD%D1%8B%20%C2%AB%D0%BF%D1%80%D0%BE%D1%85%D0%BE%D0%B4%D0%B8%D1%82%D1%8C%C2%BB%20%D1%81%D0%BF%D1%80%D0%B0%D0%B9%D1%82%D1%8B%2C%20%E2%80%94,%D0%BF%D0%BE%20%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B0%D0%BC%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%2F%D1%87%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B5%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%EE%88%80cite%EE%88%82turn2search10%EE%88%82turn2search16%EE%88%81).

## 4) AI (стратегия, тактика, дипломатия)

### 4.1 Дипломатические решения (AI Diplomacy)

* Utility‑оценки выгод/рисков: AI оценивает целесообразность установления или разрыва отношений через utility-функции. Примеры выгод для Embassy/Alliance: бонусы торговли, совместный фронт обороны, наличие общих врагов[[22]](file://file_0000000014b86246bd27d0391214fa20#:~:text=,%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%20%D0%B1%D1%8B%D1%82%D1%8C%20%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%2F%D0%BD%D0%B5%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0). Примеры “стоимости”: риск утечки информации союзнику, вероятность нежелательного слияния (если AI не хочет потерять независимость)[[22]](file://file_0000000014b86246bd27d0391214fa20#:~:text=,%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%20%D0%B1%D1%8B%D1%82%D1%8C%20%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%2F%D0%BD%D0%B5%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0). Разные ИИ-фракции имеют разную политику: агрессивные стремятся к войнам, прагматичные – к локальным посольствам по границам, коалиционные – к союзам против общего противника[[23]](file://file_0000000014b86246bd27d0391214fa20#:~:text=%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%2F%D0%BD%D0%B5%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%29.%20,%D0%BA%20%D1%81%D0%BE%D1%8E%D0%B7%D0%B0%D0%BC%20%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2%20%D0%BE%D0%B1%D1%89%D0%B5%D0%B3%D0%BE%20%D0%B2%D1%80%D0%B0%D0%B3%D0%B0).
* Обновление отношений: решения о смене отношений принимаются периодически (на стратегическом тикере). Если utility выгоды союза превышает порог – AI предложит союз (или согласится на предложение игрока); если наоборот риск превысил выгоду – может разорвать союз/посольство. Примечание: игрок инициирует дипломатию через UI, а AI – через внутренний механизм, результат всегда оформляется событием RelationsChanged.

### 4.2 Интеллект и FoW (Intel & Fog-of-War)

* У каждого государства собственный слой IntelLayer, где хранится информация о других. При отсутствии контакта или вражде – IntelAccess = None, данные отсутствуют (юниты и постройки противника скрыты, если не находятся в прямой видимости игрока). При Embassy – устанавливается ограниченный доступ (юниты и постройки союзника видны и отображаются, но информация о его экономике, запасах ресурсов, очередях производства недоступна)[[24]](file://file_0000000014b86246bd27d0391214fa20#:~:text=4.2%20%D0%98%D0%BD%D1%82%D0%B5%D0%BB%20%D0%B8%20FoW%20,%D0%BD%D0%BE%20%D0%BD%D0%B5%20%D0%BF%D0%BB%D0%B0%D0%BD%D0%B8%D1%80%D1%83%D0%B5%D1%82%20%D0%B8%D1%85%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). При Alliance – доступ Full: союзники полностью прозрачны друг для друга по данным, хотя управление остаётся раздельным.
* Тактическое поведение AI также учитывает Intel: при Embassy союзник видит войска игрока, может реагировать на их положение (например, подтягивать свои войска к участку фронта, где игрок сконцентрировался), но не планирует использование чужих войск напрямую[[24]](file://file_0000000014b86246bd27d0391214fa20#:~:text=4.2%20%D0%98%D0%BD%D1%82%D0%B5%D0%BB%20%D0%B8%20FoW%20,%D0%BD%D0%BE%20%D0%BD%D0%B5%20%D0%BF%D0%BB%D0%B0%D0%BD%D0%B8%D1%80%D1%83%D0%B5%D1%82%20%D0%B8%D1%85%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). При Alliance AI координирует действия, предполагая помощь союзных войск, но прямого контроля всё равно не имеет – т.е. союзники действуют синхронно по целям, но каждый своими отрядами.

### 4.3 Слияние государств (Merge Logic)

* Если два государства находятся в союзе и наступает событие Roll(p\_merge) – инициируется процесс слияния. Генерируется событие MergeHappened, после чего один из AI назначается главным (например, выбирается “старший союзник” с большим вкладом в войну или с наибольшей экономикой)[[25]](file://file_0000000014b86246bd27d0391214fa20#:~:text=%D0%B2%D0%BB%D0%B0%D0%B4%D0%B5%D0%BD%D0%B8%D1%8F%2F%D1%8E%D0%BD%D0%B8%D1%82%D1%8B%2F%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0%3B%20%D0%BA%D0%BE%D0%BD%D1%84%D0%BB%D0%B8%D0%BA%D1%82%D1%8B%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%20%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%B0%D0%BC%D0%B8,%E2%80%9C%D1%81%D1%82%D0%B0%D1%80%D1%88%D0%B8%D0%B9%20%D1%81%D0%BE%D1%8E%D0%B7%D0%BD%D0%B8%D0%BA%E2%80%9D%20%D0%BB%D0%B8%D0%B1%D0%BE%20%E2%80%9C%D0%BD%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B0%D1%8F%20%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0%E2%80%9D). Второе государство прекращает самостоятельное существование: его территории, постройки, юниты переходят под контроль нового объединённого государства. Экономики и бюджеты объединяются, очереди строительства и исследований сливаются (конфликтующие приоритеты решаются по заранее настроенным коэффициентам приоритетов). Игрок, если он был лидером одного из объединяемых государств, остаётся лидером своей стороны (его статус не теряется). После слияния у нового объединённого государства пересчитывается баланс (например, Stability может снизиться из-за объединения разнородных наций, а Strain повыситься). Сохранение/загрузка проверяется на корректную поддержку слияния (могут быть миграции данных при переходе версии).

### 4.4 Стратегическое и тактическое поведение (Utility AI + GOAP + BT)

* Strategic Utility AI: глобальный стратегический AI каждой фракции оценивает возможные цели каждые N тиков (реже, например раз в 0.5–1 секунду реального времени)[[26]](file://file_000000007c6861f4aefcba665986a990#:~: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). Цели могут включать: сдержать распространение инфекции, закрыть стратегическое горлышко, исследовать технологию X, поддержать союзника (игрока)[[27]](file://file_000000007c6861f4aefcba665986a990#:~: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) и т.д. Каждой цели соответствует utility-функция, нормализованная в диапазон [0..1], которая вычисляет важность этой цели на основе текущего состояния (например, близости врага к базе, количества доступных ресурсов, уровня угрозы инфекции и пр.). AI выбирает цель с максимальным utility как приоритетную.
* GOAP (Goal-Oriented Action Planning): выбрав стратегическую цель, AI строит план действий для её достижения с помощью планировщика GOAP[[27]](file://file_000000007c6861f4aefcba665986a990#:~: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)[[28]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%B4%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B2%D1%80%D0%B0%D0%B3%D0%B0%2F%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%2F%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B%29%20%E2%80%94%20%D0%BB%D0%B5%D0%B3%D0%BA%D0%B0). Например, цель "оборона рубежа" может привести к плану: собрать ресурсы → исследовать технологию «Щит-1» → построить 3 турели по дуге X. Планировщик ограничен по глубине и времени поиска, чтобы не зависнуть; кэшируются типовые под-планы для повторного использования[[29]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D1%80%D0%B0%D0%B7%D0%B1%D0%B8%D0%B2%D0%B0%D0%B9%D1%82%D0%B5%20%D0%BF%D0%BE%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B0%D0%BC%20%28budgeted%20AI%29,%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). При каждом стратегическом тике, если текущая цель выполнена или изменилась ситуация, AI может перепланировать (но чтобы избежать дергания, вводится минимальный интервал или inertia цели). Референс: реализация GOAP вдохновлена классической работой Jeff Orkin (F.E.A.R. AI Planning)[[28]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%B4%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B2%D1%80%D0%B0%D0%B3%D0%B0%2F%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%2F%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B%29%20%E2%80%94%20%D0%BB%D0%B5%D0%B3%D0%BA%D0%B0).
* Behavior Trees (BT): на тактическом уровне (для конкретных отрядов) AI использует древья поведения. BT обрабатывает микро-задачи вроде патрулирования, ремонта, перестройкиFormation. В общем случае, стратегический AI (Utility/GOAP) не микроменеджит бой, а задаёт цели, тогда как Behavior Tree выполняет их, разбивая на примитивные действия (двигаться в точку, атаковать, отступить, залечь и т.д.). BT узлы могут содержать ограничители (например, выполнять поведение не больше N тактов) и условия успеха/фейла. Важно избегать принятия крупных решений непосредственно внутри BT – за этим следят Utility/GOAP, а BT исполняет план[[30]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81). Для отладки AI можно логировать активные ветки BT с каждым тиком и их результаты (Success/Fail/Running)[[30]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81).
* Influence Maps (карты влияния): для ускорения тактических оценок AI и придания им “осведомлённости” о ситуации используются карты влияния. Различные факторы (сила врага в районе, наличие укреплений, проходимость, покрытие артиллерии и т.д.) представляются в виде полей на сетке (например, float-значение на каждый тайл)[[31]](file://file_000000007c6861f4aefcba665986a990#:~: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). Эти карты обновляются инкрементально: при изменениях (перемещение войск, строительство укрепления) пересчитывается локально вокруг изменения или по кольцевым зонам[[32]](file://file_000000007c6861f4aefcba665986a990#:~: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). Utility-оценки AI могут напрямую читать значения influence-карт вместо того, чтобы сканировать все объекты, что значительно экономит время. Примеры карт: карта риска заражения, карта оборонительного покрытия (где свои фортификации), карта давления противника на фронте. Influence-карты хранятся в массивах, пригодных для Burst, и могут обновляться параллельно (Jobs) при необходимости[[32]](file://file_000000007c6861f4aefcba665986a990#:~: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). Для визуализации и отладки эти карты могут отображаться на UI (heatmap с изолиниями)[[33]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5).
* Оптимизация AI: стратегический AI тикает разреженно (раз в несколько симуляционных тиков), а тактические компоненты – чаще (каждый тик или через тик)[[26]](file://file_000000007c6861f4aefcba665986a990#:~: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). Расчёты Utility желательно распределять по кадрам, если целей много (budgeted AI). Планировщик GOAP ограничивается по ресурсам (глубине поиска, времени), а Behavior Trees – по количеству шагов за тик (для долгих последовательностей может растягивать выполнение).

### 4.5 Координация союзников

* Союзный AI: при нахождении в союзе два AI обмениваются данными (как указано в IntelAccess) и стараются координировать стратегии. Например, цели типа “поддержать союзника” получают повышенный utility, если союзник запрашивает помощь. Механика запросов может быть реализована через события (например, игрок посылает Signal о нужде в помощи в секторе, AI-союзник усиливает туда войска). Однако прямого подчинения войска не получают – союзник решает сам, выделять ли силы. В случае, если один из союзников – человек, а другой – AI, AI будет стараться компенсировать слабые места игрока (например, перекрывать незащищённые направления, снабжать ресурсами при критическом дефиците через дипломатию). При подготовке к возможному слиянию AI может сознательно избегать конфликтов, чтобы не повышать Strain и увеличить шансы на успешное объединение.

## 5) Presentation (UI/UX)

* Экран старта: выбор начального государства для игрока (иконка, название, краткое описание бонусов/особенностей).
* Дипломатия: экран/панель отношений – список известных государств, текущий уровень (цветом или иконкой: None/Hostile/Embassy/Alliance), индикаторы доверия/стабильности. Кнопки действий: предложить посольство/союз, разорвать отношения. При поступлении предложения от AI – всплывающее уведомление с выбором ответа.
* Интел‑слои: переключатели на тактической карте: “Показать свои войска”, “Показать войска с посольствами”, “Показать войска союзников”. Отображаются только объекты, которые соответствуют текущему уровню IntelAccess. Всплывающие подсказки при наведении объясняют, что игрок видит/не видит в каждом режиме.
* Событие слияния: при слиянии государств – диалог-событие (с оповещением, звуковым сигналом), описывающий изменения. После подтверждения – временная лента/лог показывает передачу территорий, объединение армий, перерасчёт экономики.
* Экран исследований: для каждой ветки отображается ровно один доступный сейчас шаг (с описанием и требуемыми RP), а далее значки неопределённых шагов (“?” или затемнённые), указывающие, что исследования будут открываться последовательно. Шанс появления редкого шага не показывается явно, но UI может давать косвенные подсказки (например, текст о том, что “качество исследований на высоком уровне, есть шанс на редкий прорыв”). После завершения шага появляется следующий.
* Аномалии: индикатор количества доступных заражённых образцов (например, на HUD или в экране исследований). Предупреждения о утечках (Breach) – всплывающие уведомления с деталями (сектор, тяжесть). Отдельная панель автоматизации цикла аномалий, показывающая статус: сколько образцов в containment, сколько нужно для следующего шага, активна ли команда захвата.
* Автоматизация: экран сети автоматизации – отображаются построенные ServerFarm (и суммарные CU), а также все ядра. Для каждого ядра указано, к какому объекту оно прикреплено, его состояние (Active/Paused/Stalled), потребляемые ресурсы (энергия, CU). Игрок может отсюда переназначить ядро на другую цель, если свободны ресурсы.
* Экран экономики/суточного отчёта: отдельный UI, который открывается кнопкой. Показывает баланс ресурсов за последний день: произведено/потреблено каждого ресурса, куда ушло (диаграммы или списки затрат, например, сколько на содержание, сколько на строительство и пр.), узкие места (например, “не хватило энергии, X отключений”, “нехватка еды, прирост населения снижался”). Отсюда же можно понять эффективность экономики, принять решения об изменении приоритетов.
* Панель процентов передачи ИИ: UI элемент с 5 слайдерами (Labor, Power, Materials, Food, Population Growth), устанавливающими долю, которая автоматически отдаётся гос-ИИ ежедневно. Есть кнопка “Сброс к 0” или “По умолчанию”, выставляющая все 0%. Слайдеры доступны только для ресурсов, которые вообще генерируются (если, например, генерации материала нет – слайдер не активен).
* Строительное меню: улучшено фильтрами по типам зданий (экономические, военные, научные, инфраструктура). Каждое здание показывает требования (ресурсы, энергия, время), окупаемость или ROI (для экономических), ограничения по размещению. Добавлена индикация, если ресурс на постройку временно недоступен (например, не хватает энергии в сети – предупреждение).
* Логистика: окно управления перебросками – список всех активных приказов переброски. По каждому показано: откуда и куда, тип/группа юнитов, прогресс (ETA или статус “в очереди”), приоритет. Есть возможность ускорить конкретный приказ (галочка “Rush” – тратится больше ресурсов, пересчитывается ETA), либо приостановить/отменить. При наведении на приказ на карте подсвечивается соответствующий маршрут и юниты. Узлы логистики на карте отображают значок, показывающий загруженность (цветом или проценты использования).
* Фортификации: при выборе инженеров появляются инструменты постройки: кнопка “Линия окопов” (после активации игрок рисует линию по земле), кнопка “Точка укрепления” (установка одиночного сооружения). UI показывает для текущего чертежа ожидаемую стоимость (в реальном времени, пока тянется линия или выбран тип), время строительства. После подтверждения заказ добавляется в очередь строительства фортификаций этого подразделения/армии. Очередь видна в UI (можно отменить задачи). Также, при наведении на построенную фортификацию, отображается ее состояние (прочность, уровень защиты, остаток времени до износа для временных).

## 6) Infrastructure (производительность, репозиторий, масштабирование)

* Чистые модели: Domain-сущности реализованы простыми C# классами/структурами (POCO), без наследования от UnityEngine.Object. Это облегчает их сериализацию и тестирование. Статические данные (конфиги) хранятся в ScriptableObject или JSON, а при запуске копируются в эти runtime-модели[[7]](file://file_000000007c6861f4aefcba665986a990#:~: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). Благодаря этому, работа с данными не приводит к ненужным аллокациям или модификации ассетов на лету.
* Event System: для слабосвязанности используется глобальный EventBus (или посредник) – системы публикуют события, UI и другие части подписываются. Это позволяет поддерживать off-screen симуляцию (события могут накапливаться и обрабатываться, когда объект вне поля зрения), а также упрощает сохранение (состояние можно сохранять, не беспокоясь о ненужных ссылках между системами – достаточно данных Domain).
* Performance Scaling: архитектура рассчитана на постепенное повышение производительности. На начальном этапе логика может быть реализована на MonoBehaviour и обновляться на каждом тик; по мере роста нагрузки “горячие” участки (например, расчёт диффузии инфекции, генерация карт влияния, массовый поиск путей) переносятся на Jobs + Burst[[2]](file://file_000000007c6861f4aefcba665986a990#:~: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). Планируется использование DOTS (ECS) для наиболее тяжёлых подсистем (сетка заражения, массовые боевые объекты) – с фиксированным FixedStepSimulationSystemGroup для синхронизации с остальной игрой[[2]](file://file_000000007c6861f4aefcba665986a990#:~: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). Все структурные изменения в ECS делаются через EntityCommandBuffer (однократное воспроизведение изменений в конце кадра)[[34]](file://file_000000007c6861f4aefcba665986a990#:~: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), чтобы избежать затратных sync-point’ов.
* Object Pooling: для частых появляющихся и исчезающих объектов (выстрелы, эффекты, маркеры на UI) применяется пуллинг – повторное использование экземпляров вместо уничтожения/создания[[35]](file://file_000000007c6861f4aefcba665986a990#:~: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). Это особенно важно, если объекты загружаются через Addressables (Instantiate может быть медленным, но предзагрузка и повторное использование решает проблему).
* Tilemap оптимизация: для отрисовки большой карты используются Tilemap-слои. Рекомендуется базовые слои тайлов (ландшафт) держать в режиме Chunk (крупные чанки, минимизирующие draw calls), а слои, между которыми проходят спрайты (юниты, эффекты) – в режиме Individual с использованием Sprite Atlas[[36]](file://file_000000007c6861f4aefcba665986a990#:~: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). Это обеспечивает баланс между батчингом и правильным сортингом. Всегда проверять эффект настроек через Frame Debugger (Unity)[[36]](file://file_000000007c6861f4aefcba665986a990#:~: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).
* Profiling & Debug: команда начинает профилировать игру с самых ранних этапов разработки. Уже на пустой сцене с Tilemap проводится тест производительности (сравнение режимов рендера тайлов)[[37]](file://file_000000007c6861f4aefcba665986a990#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81). В ходе разработки на ключевых этапах делаются “контрольные точки” – замеры в Profiler и анализ Frame Debugger, чтобы отслеживать влияние новых систем[[38]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D0%9A%D0%BE%D1%80%D0%BE%D1%82%D0%BA%D0%B8%D0%B5%20%D0%B4%D0%BE%D0%BF,%D0%B8%D0%BD%D0%B0%D1%87%D0%B5%20%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C%20%D1%80%D0%B0%D1%81%D0%BF%D1%83%D1%85%D0%B0%D0%B5%D1%82). В сборке для тестеров можно предусмотреть debug-меню с быстрыми переключателями оверлеев (показать тики, скорости, heatmap’ы влияния, пути AI и т.п.)[[33]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5)[[39]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=UI%2F%D0%B2%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D0%B4%D0%B5%D0%B1%D0%B0%D0%B3%D0%B0%20%28%D0%B1%D1%8B%D1%81%D1%82%D1%80%D0%BE%D0%B5%20%D1%87%D0%B5%D0%BA,%D0%BC%D0%B5%D0%BD%D1%8E). Это ускоряет выявление проблем производительности и логики.
* Сетевая и DI-инфраструктура: (На будущее) Код организован по фиче-модулям (см. Integration Hooks ниже). Для подключения зависимостей между подсистемами может использоваться простой Service Locator или DI-контейнер. Например, Pathfinding-сервис, EventBus, SavingService регистрируются при старте и доступны всем системам. Это снижает объем жёсткого подключения через инспектор и облегчает тестирование (можно подменять сервисы).

## 7) Баланс и константы (первоначальный набор)

* p0 = 0.15%/day, p\_max = 12%/day, α = 0.25 – базовые коэффициенты для расчёта шанса слияния (см. формулу p\_merge выше).
* pRareBase = 8%, pRareMax = 40% – базовый и максимальный шансы выпадения редкого исследования; pAnomBase = 1% – базовый шанс аномального (ещё реже).
* Инфляция исследований: r\_common = 0.12, r\_rare = 0.14, r\_anom = 0.15 – ставки удорожания шагов для каждого типа. Применяются в формуле стоимости через (1+r)^i.
* ROI апгрейдов зданий: одно улучшенное здание должно давать прирост ≥, чем 2 базовых того же типа × коэффициент. Для категорий: Economy +10%, Military +15%, Science +30%, Containment +20%. (Напр., улучшенная электростанция должна давать хотя бы ~2.2× выход энергии относительно двух обычных).
* ServerFarm/Core: ServerFarm\_T1 даёт 20 CU (compute units); одно AutomationCore потребляет 10 CU и ~2.5 MW энергии.
* Целочисленные расчёты: все ресурсы и их изменения хранятся целыми числами (без плавающих). Дробные коэффициенты (например, прирост с процентами) рассчитываются с использованием фиксированной точки – умножаются ×1000 (для трёх знаков) и по завершении вычислений округляются вниз до целого. Это гарантирует отсутствие расхождений и постепенного накопления ошибок, а также упрощает сохранение.
* Детерминированность: симуляционный цикл должен быть детерминированным при заданном сид генератора. Все случайные события (роллы) идут от единого генератора со стартовым randomSeed, сохраняемого в сейве. Это позволяет повторять сценарии (при отладке) и синхронизировать off-screen вычисления с on-screen результатами (например, автоматизация vs ручное управление дают идентичный итог с погрешностью ≤5%).

## 8) План внедрения (5 спринтов: базовый + доп)

### Спринт 0 — Базовая структура и системы (первая 1–2 недели)

Дни 1–2: Настройка проекта. Создать базовые папки и пустую сцену. Настроить Tilemap (проверить режимы Chunk vs Individual для слоёв). Реализовать минимальный цикл SimulationLoop с поддержкой паузы, разных скоростей и шагового режима[[11]](file://file_000000007c6861f4aefcba665986a990#:~:text=2,%D1%86%D0%B8%D0%BA%D0%BB%D1%83). Добавить базовые интерфейсы времени и событий (Clock, EventBus). Тест производительности: подключить Unity Profiler и Frame Debugger сразу на этом этапе, убедиться, что пустая сцена рендерится оптимально (тайлы объединены в батчи, нет лишних draw calls)[[37]](file://file_000000007c6861f4aefcba665986a990#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81).

Дни 3–5: Сделать начальные ScriptableObject каталоги: Units, Buildings, Techs, InfectionParams – заполнить парой примеров. На их основе реализовать создание runtime-объектов (например, UnitType, BuildingType, TechDef, InfectionModel) при старте. Разработать EconomySystem v1: базовые классы ресурсов, ежедневный прирост и потребление (пока с упрощённой логикой). InfectionSystem v1: реализовать упрощённую SIR-модель на клеточной сетке (можно без диффузии, только подсчёт заражённых соседей). Saving v1: внедрить сохранение/загрузку в JSON, учитывая ограничения JsonUtility (избегать Dictionary, использовать [Serializable] классы)[[40]](file://file_000000007c6861f4aefcba665986a990#:~: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,%EE%88%80cite%EE%88%82turn2search10%EE%88%81). Добавить SaveVersion = 1. End-to-end тест: запустить игру, подождать несколько тиков, сохранить, загрузить – убедиться, что состояние восстанавливается (юниты на тех же позициях, ресурсы не отрицательные и т.д.)[[41]](file://file_000000007c6861f4aefcba665986a990#:~: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).

Дни 6–8: AI v1: внедрить 3–5 простых Utility-метрик (например, “угроза инфекции в стране”, “врагов на границе”, “ресурсов накоплено”) и простейший стратегический тикер, который каждые N тиков выводит в лог наиболее приоритетную цель. Добавить прототип Influence-карты для инфекции (float-поле на карте, обновляется от InfectionSystem) и расчет условной “линии фронта” (массив граничных клеток). Object Pool: подключить пул объектов (Unity ObjectPool<T>) для повторного использования хотя бы простых эффектов (например, маркеры заражения или пули)[[42]](file://file_000000007c6861f4aefcba665986a990#:~:text=%E2%80%94%20Utility,T%3E%60%29%20%EE%88%80cite%EE%88%82turn6search6%EE%88%81).

Дни 9–10: AI v2: добавить GOAP-планировщик на 2–3 типа действий (например, “собрать ресурсы”, “построить турель”, “исследовать технологию X”) и пару узлов Behavior Tree для юнитов (патруль района, ремонт ближайшего здания). Pathfinding v1: реализовать A\* поиск пути на сетке, интегрировать Jump Point Search для ускорения на прямых участках[[17]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn3search5%EE%88%82turn3search20%EE%88%81). Сделать API для пакетного запроса путей (список целей → список маршрутов), результаты которого можно получать асинхронно (корутина или Task).

Дни 11–14: Оптимизация шаг 1: найти самые затратные места (ожидаемо InfectionSystem, InfluenceMap). Разбить их обновление на Jobs + Burst – например, расчёт диффузии инфекции через IJobParallelFor по частям сетки, генерация influence-карты – тоже параллельно[[43]](file://file_000000007c6861f4aefcba665986a990#:~: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). Убедиться, что Burst-компиляция работает (времено включить Native Debug Mode для отладки)[[44]](file://file_000000001f0461f4847b90af8dfa8700#:~: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). Addressables: перевести загрузку тяжёлых ресурсов (спрайты юнитов, текстуры) на Addressables, удостовериться в паре мест, что Release вызывается после выгрузки (память не растёт)[[45]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%BF%D0%BE%20%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B0%D0%BC%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%2F%D1%87%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B5%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%EE%88%80cite%EE%88%82turn2search10%EE%88%82turn2search16%EE%88%81). UI Dev-меню: добавить оверлей отображения текущей скорости/тикового индекса, heatmap заражения, простейшие кнопки быстрого вызова Profiler/FrameDebugger[[39]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=UI%2F%D0%B2%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D0%B4%D0%B5%D0%B1%D0%B0%D0%B3%D0%B0%20%28%D0%B1%D1%8B%D1%81%D1%82%D1%80%D0%BE%D0%B5%20%D1%87%D0%B5%D0%BA,%D0%BC%D0%B5%D0%BD%D1%8E). Если по результатам профилирования узкие места остались, сделать POC на ECS: например, вынести InfectionSystem в ECS (FixedStepSimulation) для дальнейшего тестирования производительности параллельно основной реализации[[46]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%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).

(По итогу двух недель должен работать базовый каркас RTS: пауза, тик, примитивная экономика и инфекция, простые AI-решения, сохранения. Далее – внедрение расширенных механик дополняющих модулей.)

### Спринт 1 — Государства и видимость (1–1.5 недели)

Что делаем: добавляем поддержку нескольких государств: создаём модели State/Relation/IntelLayer, интегрируем их в существующие системы (юниты теперь принадлежат государствам, экономики раздельны). Реализуем события и логику смены отношений (DiplomacySystem). Добавляем влияние уровня отношений на видимость: внедряем FoW (туман войны) на основе IntelLayer – враги не видны на карте без разведки или договорённостей. Настраиваем стартовый экран выбора государства игроком, базовый UI панели дипломатии (отображение отношений, кнопки предложить/разорвать). Проверяем EventBus-события при смене отношений.  
Готово, когда: AI-противники не видят чужие юниты/здания без соответствующих отношений; при уровне Embassy – видят юниты/постройки союзника, но не его ресурсы; при Alliance – видят всё, однако не могут управлять чужими юнитами. Сохранение/загрузка включает состояния государств и их отношений, корректно восстанавливает IntelLayer.

### Спринт 2 — Слияние и экономика совместимости (1 неделя)

Что делаем: реализуем формулу и процесс слияния государств: считаем p\_merge каждый день для союзов, добавляем таймеры отсчёта дней в союзе, Stability, Strain. При слиянии выполняем миграцию данных (территорий, юнитов, очередей) в коде DiplomacySystem. Дорабатываем экономику под много-государственный режим: проверяем, что суточный тик учитывает раздельные бюджеты и транзакции между игроком и AI (процент отчисления). Делаем UI-событие для слияния (всплывающее окно с информацией). Внедряем механику дефицитов и приоритетов: настраиваем списки приоритетных расходов (см. Domain: содержание населения и войск прежде всего, и т.д.) и логику отключений/снижения, когда ресурс в минусе[[47]](file://file_0000000049b062438e06b107048784b7#:~: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)[[48]](file://file_0000000049b062438e06b107048784b7#:~: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).  
Готово, когда: каждый день в союзе выполняется ролл слияния; повышенная Stability увеличивает шанс, а конфликты (Strain) уменьшают. При наступлении события оба государства объединяются в одно, AI выбирается по правилу, экономика и войска объединены. Сохранение после слияния корректно загружается (нет дублирования или пропажи объектов). В экономической системе: при нехватке энергии происходит отключение низкоприоритетных потребителей, при голоде населения – падает прирост CivGrowth и могут происходить события (бунты) вместо немедленного краха экономики. Все изменения отражаются в суточном отчёте.

### Спринт 3 — Исследования v2 (1.5 недели)

Что делаем: перерабатываем систему исследований на основе ResearchProgress/Branch: теперь у каждой ветки есть только один текущий шаг (ResearchDef), видимый игроку. После завершения шага присваивается следующий на основе редкости: рассчитывается QualityScore от зданий науки, на его основе pRare (формула выше) – иногда выпадают Rare вместо обычных. Добавляем учёт аномалий: если Branch содержит аномальные исследования, они становятся доступны только при наличии специальных условий (построен ContainmentLab и есть активная CaptureTeam, а также накоплено N заражённых образцов). Реализуем апгрейды лабораторий: добавляем параметры RP (скорость) и QP (качество), влияющие на скорость прогресса и QualityScore соответственно. В интерфейсе – обновляем экран исследований согласно новым правилам.  
Готово, когда: при завершении исследования ветки игрок видит новый доступный шаг; иногда вместо ожидаемого Common появляется Rare (с частотой в соответствии с качеством). Без построенной лаборатории содержания и отряда захвата все технологии с rarity = Anomalous помечены как недоступные (и в UI отображается, что нужны особые условия). Значения ROI для научных апгрейдов соблюдаются (например, Lab\_T2 даёт ~2.2× RP относительно Lab\_T1), QualityScore влияет на шанс редких технологий. Баланс проверен: стоимость исследований монотонно растёт, редкие не выскакивают слишком часто.

### Спринт 4 — Аномалии и Автоматизация (1.5 недели)

Что делаем: добавляем в игру новый тип зданий и юнитов для работы с аномалиями: лаборатория содержания заражённых (ContainmentLab) и команда захвата (CaptureTeam). При наличии этих объектов появляются аномальные исследования (ветки типа "anomaly."). Реализуем механику расхода образцов: каждое исследование аномалии требует определённое количество заражённых образцов; образцы попадают в ContainmentLab, когда CaptureTeam успешно выполнит миссию (можно упростить: раз в N дней команда добывает M образцов, если рядом есть заражённый сектор). Настраиваем события Breach – если накоплено слишком много образцов или отключилась энергия, есть шанс побега/утечки (событие, влияющее на заражение вокруг). Автоматизация: вводим здания ServerFarm (дают ComputeUnits) и объекты AutomationCore. Реализуем AutomationSystem, позволяющий привязать ядро к конкретному процессу (через интерфейс IAutomatable – например, к конкретной ветке исследований или отряду CaptureTeam). Ядро автоматически вызывает методы Start/CompleteStep у цели, имитируя игрока. В интерфейсе – экран автоматизации сети. Off-screen симуляция: убеждаемся, что автоматизированные процессы дают тот же результат, что и ручные, при перемотке (погрешность не более 5%). Оптимизируем нагрузку: automation tick происходит раз в несколько кадров, если объектов много.  
Готово, когда:\* цикл “поимка → доставка → исследование” работает в игре. Игрок может вручную отправлять CaptureTeam, получать образцы и тратить их на исследования аномалий. При установке ядра автоматизации на CaptureTeam и на лабораторию, цикл выполняется без вмешательства игрока (пока хватает энергии и CU). Если энергия или вычислительные мощности заканчиваются, ядро ставит процесс на паузу (UI показывает статус Stalled). Интерфейс автоматизации позволяет отслеживать все ядра и их цели. Off-screen тест: если запустить игру на x10 скорости с автоматизацией и сравнить с пошаговым выполнением тех же действий, итоги совпадают (например, за 30 дней собрали X образцов и выучили технологию Y в обоих случаях ±5%).

(После этих спринтов все новые механики интегрированы. Далее – полировка, тестирование, баланс.)

## 9) Тест-план и инварианты

* Дипломатия/Intel: убедиться, что без отношений AI имеет нулевое знание о другом (при отладке – списки KnownUnits пусты). При Embassy – AI видит юниты/постройки союзника (проверить, что KnownUnits/Buildings заполняются), но не видит его Economy (KnownEconomy = null). При Alliance – KnownEconomy не null, доступны очереди. Контроль юнитов остаётся раздельным: попытка AI приказать юниту игрока игнорируется.
* Слияние: проверить детерминизм – при фиксированном сид и условиях слияние происходит на той же день с тем же вероятностным исходом. После слияния все юниты поглощённого гос-ва имеют новый StateId, все его постройки и территории также переключились. AI продолжает работать (не теряет контроль). Сохранение сразу после слияния и загрузка – состояние объединённого государства корректно (нет дубликатов юнитов, ресурсы суммированы).
* Экономика/Ресурсы: инвариант баланса ресурсов – ресурсы не возникают из воздуха и не уходят в отрицательные: суммарный приход-расход за любой период ≥ 0. Написать unit-тесты на методы экономического резерва: Reserve не изменяет баланс при попытке зарезервировать, Commit списывает и сумма всех Commit равна сумме Reserve (деньги не исчезают)[[49]](file://file_000000001f0461f4847b90af8dfa8700#:~: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 суточных тиков с хаотичными тратами гос-ИИ и игрока – проверить, что итоговые балансы предсказуемы и не отрицательны. Дефицит-тест: установить потребление энергии выше генерации – убедиться, что сначала отключаются низкоприоритетные здания, и ни одно потребление не продолжает расти, когда ресурс 0. Аналогично для еды – при 0 еды прирост CivGrowth падает к 0, начинаются события (бунты) если держать долго.
* Логистика: тест “Логистический пик” – задать множество перебросок по одному узкому коридору, убедиться, что они выстраиваются в очередь, суммарно не превышают capacityUnitsPerDay[[4]](file://file_0000000049b062438e06b107048784b7#:~: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), ETA рассчитывается с учётом ожидания. При отключении режима Rush все прибывают в срок T, при включении Rush у некоторых – быстрее, но списались дополнительные материалы/энергия. Разрушить узел на пути – проверить, что существующие приказы перенаправляются (или выдают предупреждение “маршрут недоступен”).
* Фортификации: проверить, что временные укрепления теряют прочность со временем (смоделировать 30 дней, убедиться, что параметр cover снижается у окопа). Постоянные – не снижаются (или гораздо медленнее). Строительство: разместить длинную линию траншеи длиной L, послать 2 инженерных отряда – убедиться, что задача разделилась (каждый строит свой участок)[[50]](file://file_0000000049b062438e06b107048784b7#:~: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,%D1%87%D0%B0%D1%81%D0%BE%D0%B2%2F%D0%BC%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D0%BE%D0%B2) и общее время ~ равно L / (сумма скоростей). Попробовать улучшить временную позицию до постоянной – затраченные ранее трудо-часы учтены (например, если окоп был наполовину достроен до бункера, то на бункер осталось 50%).
* Исследования: проверить рост стоимости – завершить несколько шагов подряд, убедиться, что новый CostMultiplier вырос по формуле. Редкие исследования выпадают приблизительно с заданной частотой: можно выполнить 100 шагов при высоком QualityScore и подсчитать долю Rare – она должна быть близка к рассчитанному pRare (например, при QS, дающем 20% шанс, из 100 исследований ~20 оказались Rare). Шанс не показывается в UI напрямую – убедиться, что нигде не раскрывается игроку точное число (только описания).
* Аномалии: без ContainmentLab и CaptureTeam все Anomalous-технологии недоступны (в UI – серые, с пометкой о требовании). При наличии инфраструктуры – становятся исследуемыми. Проверить, что при начале исследования аномалии списываются требуемые образцы (и что их надо предварительно накопить). Инцидент-тест: хранить много образцов (больше capacity) и отключить лаборатории – должен сработать event Breach (можно подкрутить вероятность для теста). После Breach – infection в секторе вырос, событие записано.
* Автоматизация: убедиться, что AutomationCore не может быть прикреплено более чем к одной цели одновременно. Попробовать назначить два ядра на одну цель – второе должно отказать (или заменить первое – определиться в дизайне, но нельзя дублировать эффект). Проверить, что при нехватке энергии или CU процесс ставится на паузу: имитировать отключение энергии – AutomationCore генерирует событие CoreStalled, цель ничего не делает до возобновления. Сравнить результат длительного процесса (например, исследований) с ядром и без – за равный промежуток (например, 10 дней перемотки) разница ≤5%.
* Производительность: запустить масштабный сценарий: большая карта, 1000+ юнитов, десятки построек, активная инфекция. Проверить в Profiler использование CPU: SimulationLoop не превышает допустимые 16ms на кадр (при необходимости снизить частоту тиков для некоторых систем). Память: сделать несколько циклов загрузки-выгрузки сцены – убедиться, что нет утечек (Addressables все освобождены, объекты уничтожены). Профтест на pathfinding: вызвать 100 одновременных поисков пути (например, по логистике) – время должно масштабироваться под строку (если слишком долго, задуматься о дроблении на несколько кадров).

## 10) Риски и меры

* Сложность UI (дипломатия, исследования): риск перегрузить игрока информацией. Меры: начинаем с простого интерфейса “одна карточка на ветку” для исследований и упрощённой панели дипломатии. Постепенно усложняем, если нужно, ориентируясь на отзывы. Добавляем лог событий, чтобы игрок мог отслеживать скрытые процессы.
* Баланс вероятностей: вероятность слияния и выпадения редких технологий трудно подбирается “на глаз”. Меры: встроить телеметрию – собирать статистику по частоте слияний и редких технологий. Иметь возможность в патче легко изменить p\_max, pRareMax и др. Ограничить сверху экстремальные значения (например, даже при супер-качестве pRareMax = 40%, не 100%).
* Фризы при слиянии: объединение государств – тяжёлый процесс (много объектов перемещается за один кадр). Меры: реализовать пошаговую миграцию (например, переносить не более N юнитов за кадр, растягивая процесс на несколько кадров, с красивой анимацией). Использовать корутины или ECS-команды с разбиением.
* Эксплойт редкости исследований: игрок может пытаться манипулировать QualityScore (например, отключать/включать лаборатории) чтобы повысить шанс Rare. Меры: QualityScore рассчитывается по накопленным показателям и сглаженно, не давая резко поднять шанс. Шанс редкого всегда скрыт, а при резком росте качества можно ввести кулдаун на выпадение Rare (например, не чаще чем 1 раз в 3 шага).
* Дисбаланс ресурсов и войны: есть риск, что экономика станет слишком сложной и отвлечёт от военной части, или наоборот, война сделает экономику неважной. Меры: настроить коэффициенты ROI так, чтобы без минимальной экономики войну не выиграть (нужно снабжение), а без побед в боях экономика быстро попадёт под удар. Ввести события типа “беженцы” или “мобилизация”, связывающие военные успехи и экономику (например, большие потери снижают Stability и CivGrowth). Отслеживать метрики узких мест: если игроку постоянно не хватает какого-то ресурса, подсказать через UI, какую постройку или действие сделать.
* Производительность на масштабах: сотни юнитов, infection на всю карту, расчёт путей – всё это может превысить возможности CPU. Меры: активно использовать оптимизации: влияние – инкрементно, infection – обновление только фронта, pathfinding – иерархический и batched, тяжелые вычисления выносить в Jobs/Burst. Предусмотреть настройки масштаба: например, на большой карте стратегические тики реже, максимум N юнитов одновременно активно сражаются (остальные “спят” вне экрана), возможность понижения качества симуляции (LOD) для фоновый действий.

## 11) Hooks интеграции (структура кода)

/Game/Domain/States/\*.cs // State, Relation, IntelLayer (дипломатия, видимость)  
/Game/Domain/Research/\*.cs // ResearchProgress, Branch, ResearchDef (ветки исследований, редкости)  
/Game/Domain/Anomaly/\*.cs // Anomaly & Containment (образцы, события Breach)  
/Game/Domain/Automation/\*.cs // AutomationNetwork, AutomationCore, IAutomatable  
/Game/Domain/Economy/\*.cs // EconomyModel, Resource, Budget, Contracts (экономика и бюджеты)  
/Game/Domain/Infection/\*.cs // InfectionModel, InfectionCell, Quarantine (заражение и карантины)  
/Game/Domain/Logistics/\*.cs // LogisticNode, LogisticLink, TransferOrder (узлы и коридоры снабжения)  
/Game/Domain/Fortification/\*.cs // Fortification definitions, construction task (полевые укрепления)  
  
/Game/Application/Systems/\*.cs // DiplomacySystem, MergeSystem, ResearchSystemV2, AutomationSystem,  
 // EconomySystem (daily tick), InfectionSystem, LogisticsSystem, FortificationSystem  
/Game/AI/Strategic/\*.cs // DiplomacyPlanner, MergeDecider, StrategicPlanner (Utility AI), GoalPlanner (GOAP)  
/Game/AI/Tactical/\*.cs // BehaviorTrees, UnitAI (поведение отдельных юнитов)  
/Game/Presentation/UI/\*.cs // UI дипломatii, экран исследований, панель автоматизации, экономика/логистика UI  
/Game/Data/Defs/\*.asset // SO каталоги: StateDefs, ResearchDefs, BuildingDefs, UnitDefs, InfectionParams, RecipeDefs, FortificationDefs ...  
/Game/Infrastructure/Saving/\*.cs // SaveManager, SaveMigrator (обработка SaveVersion)  
/Game/Infrastructure/Pathfinding/\*.cs // Pathfinding service (A\*, JPS, HPA\* algorithms)

Примечание: архитектурные слои разделены по папкам. Domain содержит логику и модели; Application – системы, управляющие этими моделями; AI – алгоритмы принятия решений; Presentation – MonoBehaviour-скрипты для UI и визуалов; Data – скриптабл-объекты и конфиги; Infrastructure – утилиты сохранения, загрузки, внешние сервисы. Такой organization облегчает навигацию и поддержку кода, а также подключение/отключение модулей (например, можно модульно тестировать Domain-логику, подменяя Application-слой).

### Приложение A — Псевдокод выбора следующего исследования

void CompleteResearchStep(State s, string branchId) {  
 var branch = s.Research.Branches[branchId];  
 var def = branch.Current;  
 // Mark current step as completed:  
 branch.Completed += 1;  
 s.Research.TotalSteps += 1;  
 ApplyEffects(def); // применить эффекты технологии (если есть)  
 // Calculate new cost multiplier due to inflation:  
 float r = GetInflationRate(def.Rarity);  
 s.Research.CostMultiplier = Mathf.Pow(1 + r, s.Research.TotalSteps);  
  
 // Determine next step:  
 float QS = s.Research.QualityScore;  
 float pRare = Mathf.Clamp(pRareBase + alpha \* QS, 0, pRareMax);  
 ResearchDef next;  
 if (def.Rarity == ResearchRarity.Anomalous) {  
 // Anomalous branch might end or loop differently  
 next = null; // (можно предусмотреть конец ветки после аномалии)  
 } else if (Random.value < pRare) {  
 next = PickRandom(def.BranchId, rarity: Rare, excludeCompleted:true);  
 } else {  
 next = PickRandom(def.BranchId, rarity: Common, excludeCompleted:true);  
 }  
 // If Anomalous possible:  
 float pAnom = CalcAnomalyChance(QS, s.AnomalyStats);  
 if (next == null && Random.value < pAnom && AllAssetsPresent(def.BranchId)) {  
 next = GetAnomalousDef(def.BranchId);  
 }  
 branch.Current = next;  
 EventBus.Publish(new Research.NextStepAssigned(s.Id, branchId, next?.Id, next?.Rarity));  
}

(Этот псевдокод иллюстрирует логику прогрессии исследований: инфляция стоимости, выбор следующего шага с учётом QualityScore и доступности аномальных технологий.)

[[1]](file://file_000000007c6861f4aefcba665986a990" \l ":~:text=%D0%BB%D0%B8%D0%BC%D0%B8%D1%82%D1%8B%20%D0%BA%D0%B0%D0%BF%D0%B5%D0%BA%D1%81%D0%B0%2F%D0%BE%D0%BF%D0%B5%D0%BA%D1%81%D0%B0%29,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) [[2]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[3]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[5]](file://file_000000007c6861f4aefcba665986a990#:~:text=9%29%20%D0%9F%D1%83%D1%82%D1%8C%20%D1%8E%D0%BD%D0%B8%D1%82%D0%B0%20,%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) [[6]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%BF%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B8%D0%BD%D0%B3%D0%B0%3B%20%D0%B5%D1%81%D1%82%D1%8C%20%D1%83%D0%BB%D1%83%D1%87%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20JPS%2B%20%EE%88%80cite%EE%88%82turn1search3%EE%88%82turn1search11%EE%88%81) [[7]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[8]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%EE%88%80cite%EE%88%82turn2search10%EE%88%81) [[10]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[11]](file://file_000000007c6861f4aefcba665986a990#:~:text=2,%D1%86%D0%B8%D0%BA%D0%BB%D1%83) [[16]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[20]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[21]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%BC%D0%B8%20%D0%B4%D0%BE%D0%BB%D0%B6%D0%BD%D1%8B%20%C2%AB%D0%BF%D1%80%D0%BE%D1%85%D0%BE%D0%B4%D0%B8%D1%82%D1%8C%C2%BB%20%D1%81%D0%BF%D1%80%D0%B0%D0%B9%D1%82%D1%8B%2C%20%E2%80%94,%D0%BF%D0%BE%20%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B0%D0%BC%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%2F%D1%87%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B5%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%EE%88%80cite%EE%88%82turn2search10%EE%88%82turn2search16%EE%88%81) [[26]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[27]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[28]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%B4%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B2%D1%80%D0%B0%D0%B3%D0%B0%2F%D1%83%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%2F%D0%BA%D0%BE%D1%80%D0%B8%D0%B4%D0%BE%D1%80%D1%8B%29%20%E2%80%94%20%D0%BB%D0%B5%D0%B3%D0%BA%D0%B0) [[29]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D1%80%D0%B0%D0%B7%D0%B1%D0%B8%D0%B2%D0%B0%D0%B9%D1%82%D0%B5%20%D0%BF%D0%BE%20%D0%BA%D0%B0%D0%B4%D1%80%D0%B0%D0%BC%20%28budgeted%20AI%29,%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) [[31]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[32]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[34]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[35]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[36]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[37]](file://file_000000007c6861f4aefcba665986a990#:~:text=,Chunk%2FIndividual%29%20%EE%88%80cite%EE%88%82turn3search11%EE%88%82turn3search3%EE%88%81) [[38]](file://file_000000007c6861f4aefcba665986a990#:~:text=%D0%9A%D0%BE%D1%80%D0%BE%D1%82%D0%BA%D0%B8%D0%B5%20%D0%B4%D0%BE%D0%BF,%D0%B8%D0%BD%D0%B0%D1%87%D0%B5%20%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C%20%D1%80%D0%B0%D1%81%D0%BF%D1%83%D1%85%D0%B0%D0%B5%D1%82) [[40]](file://file_000000007c6861f4aefcba665986a990#:~: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,%EE%88%80cite%EE%88%82turn2search10%EE%88%81) [[41]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[42]](file://file_000000007c6861f4aefcba665986a990#:~:text=%E2%80%94%20Utility,T%3E%60%29%20%EE%88%80cite%EE%88%82turn6search6%EE%88%81) [[43]](file://file_000000007c6861f4aefcba665986a990#:~: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) [[45]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%D0%BF%D0%BE%20%D1%81%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B0%D0%BC%20%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8%2F%D1%87%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B5%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%EE%88%80cite%EE%88%82turn2search10%EE%88%82turn2search16%EE%88%81) [[46]](file://file_000000007c6861f4aefcba665986a990#:~:text=,%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) document-2.txt

<file://file_000000007c6861f4aefcba665986a990>

[[4]](file://file_0000000049b062438e06b107048784b7#:~: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) [[47]](file://file_0000000049b062438e06b107048784b7#:~: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) [[48]](file://file_0000000049b062438e06b107048784b7#:~: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) [[50]](file://file_0000000049b062438e06b107048784b7#:~: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,%D1%87%D0%B0%D1%81%D0%BE%D0%B2%2F%D0%BC%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D0%BE%D0%B2) document-4.txt

<file://file_0000000049b062438e06b107048784b7>

[[9]](file://file_0000000014b86246bd27d0391214fa20#:~:text=3.3%20%D0%A1%D0%BE%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20,%D0%B4%D0%BB%D1%8F%20%D0%B1%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%B2%20%D1%80%D0%B5%D0%B4%D0%BA%D0%BE%D1%81%D1%82%D0%B8%2F%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F) [[14]](file://file_0000000014b86246bd27d0391214fa20#:~:text=,%D0%B4%D0%BB%D1%8F%20%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85%20%D1%81%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2%20%28%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%82%D1%8B) [[22]](file://file_0000000014b86246bd27d0391214fa20#:~:text=,%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%20%D0%B1%D1%8B%D1%82%D1%8C%20%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%2F%D0%BD%D0%B5%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0) [[23]](file://file_0000000014b86246bd27d0391214fa20#:~:text=%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%2F%D0%BD%D0%B5%D0%B6%D0%B5%D0%BB%D0%B0%D0%BD%D0%BD%D0%B0%29.%20,%D0%BA%20%D1%81%D0%BE%D1%8E%D0%B7%D0%B0%D0%BC%20%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2%20%D0%BE%D0%B1%D1%89%D0%B5%D0%B3%D0%BE%20%D0%B2%D1%80%D0%B0%D0%B3%D0%B0) [[24]](file://file_0000000014b86246bd27d0391214fa20#:~:text=4.2%20%D0%98%D0%BD%D1%82%D0%B5%D0%BB%20%D0%B8%20FoW%20,%D0%BD%D0%BE%20%D0%BD%D0%B5%20%D0%BF%D0%BB%D0%B0%D0%BD%D0%B8%D1%80%D1%83%D0%B5%D1%82%20%D0%B8%D1%85%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) [[25]](file://file_0000000014b86246bd27d0391214fa20#:~:text=%D0%B2%D0%BB%D0%B0%D0%B4%D0%B5%D0%BD%D0%B8%D1%8F%2F%D1%8E%D0%BD%D0%B8%D1%82%D1%8B%2F%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0%3B%20%D0%BA%D0%BE%D0%BD%D1%84%D0%BB%D0%B8%D0%BA%D1%82%D1%8B%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%20%D1%80%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B0%D1%8E%D1%82%D1%81%D1%8F%20%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%B0%D0%BC%D0%B8,%E2%80%9C%D1%81%D1%82%D0%B0%D1%80%D1%88%D0%B8%D0%B9%20%D1%81%D0%BE%D1%8E%D0%B7%D0%BD%D0%B8%D0%BA%E2%80%9D%20%D0%BB%D0%B8%D0%B1%D0%BE%20%E2%80%9C%D0%BD%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B0%D1%8F%20%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B0%E2%80%9D) ADDON\_STATES\_RESEARCH\_AUTOMATION\_PLAN.md

<file://file_0000000014b86246bd27d0391214fa20>

[[12]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=6%29%20%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%3A%20,%EE%88%80cite%EE%88%82turn0search12%EE%88%81) [[13]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D1%8F%2F%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%2F%D1%81%D0%B5%D0%B9%D0%B2%D1%8B) [[15]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn1search6%EE%88%81) [[17]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn3search5%EE%88%82turn3search20%EE%88%81) [[18]](file://file_000000001f0461f4847b90af8dfa8700#:~: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) [[19]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=1%29%20MonoBehaviour,%EE%88%80cite%EE%88%82turn0search12%EE%88%82turn0search20%EE%88%82turn0search24%EE%88%81) [[30]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%EE%88%80cite%EE%88%82turn2search5%EE%88%81) [[33]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=,%D1%80%D0%B5%D0%B6%D0%B8%D0%BC%D0%B5) [[39]](file://file_000000001f0461f4847b90af8dfa8700#:~:text=UI%2F%D0%B2%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D0%B4%D0%B5%D0%B1%D0%B0%D0%B3%D0%B0%20%28%D0%B1%D1%8B%D1%81%D1%82%D1%80%D0%BE%D0%B5%20%D1%87%D0%B5%D0%BA,%D0%BC%D0%B5%D0%BD%D1%8E) [[44]](file://file_000000001f0461f4847b90af8dfa8700#:~: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) [[49]](file://file_000000001f0461f4847b90af8dfa8700#:~: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) document-3.txt

<file://file_000000001f0461f4847b90af8dfa8700>