УДК 004.94 : 794.5

ИНФОРМАЦИОННЫЕ И МАТЕМАТИЧЕСКИЕ МОДЕЛИ КОМПЬЮТЕРНОЙ ИНКРЕМЕНТАЛЬНОЙ ИГРЫ

Шабанов И.О.

студент 4го курса бакалавриата

Уфимский университет науки и технологий

Уфа, Россия

Каримов Р.Р.

к.т.н., доцент

Уфимский университет науки и технологий

Уфа, Россия

Аннотация

Рассматриваются информационные и математические модели, необходимые для разработки компьютерной инкрементальной игры с элементами Tower Defense. Разработаны модели игрового цикла, диаграмма прецедентов и диаграмма классов. Предложена и реализована математическая модель прогрессии параметров игрового цикла. Реализован вертикальный срез компьютерной игры в игровом движке Unity, который доказал корректность предложенных моделей и работоспособность программы.

Ключевые слова: инкрементальная компьютерная игра, Tower Defense, игровой цикл, UML, диаграмма прецедентов, диаграмма классов, математическая модель, игровой баланс, игровой движок Unity.

INFORMATION AND MATHEMATICAL MODELS OF COMPUTER INCREMENTAL GAME

Shabanov I.O.

4th year undergraduate student

Ufa University of Science and Technology,
Ufa, Russia

Karimov R.R.
Ph.D., Associate Professor,
Ufa University of Science and Technology,
Ufa, Russia

Abstract

The information and mathematical models for the development of an incremental computer game similar to Tower Defense genre. Game loop models, a precedent diagram, and a class diagram have been developed. A mathematical model of the progression of game loop parameters is proposed and implemented. A vertical slice of a computer game has been implemented in the Unity game engine, which proved proposed models and the program's performance.

Keywords: incremental computer game, Tower Defense, game loop, UML, precedents diagram, class diagram, mathematical model, game balance, Unity game engine.

Введение

Жанр инкрементальных игр относится к числу наиболее популярных жанров компьютерных игр. При этом несмотря на кажущуюся простоту визуального стиля, смысла игры и казуальность данный жанр обладает большим потенциалом с точки зрения применения математических методов и алгоритмов, сочетания тактики и стратегии, что способствует раскрытию интеллектуальных способностей пользователя и позволяет широко их использовать в обучении и деловых тренингах [6;7;8; 9].

В статье рассматривается разработка инкрементальной компьютерной игры «Guardo's Defense Quest», в которой игрок обороняет замок от волн противников, собирает и перерабатывает ресурсы и постепенно развивает

персонажа с помощью дерева улучшений, прокачки оружия и механики перерождения.

Актуальность работы обусловлена устойчивой популярностью жанра инкрементальных игр и спецификой их разработки. В отличие от большинства жанров, основная сложность инкрементальной игры заключается не в графической составляющей, а в математической модели прогрессии и согласованной работе множества взаимосвязанных подсистем.

Цель работы — обеспечение сбалансированного, предсказуемого и долгосрочного игрового прогресса в компьютерной инкрементальной игре за счет применения формализованных математических моделей экспоненциального роста, конвертации ресурсов и динамической балансировки характеристик [2].

К основным особенностям жанра инкрементальной игры относятся [1]:

– прогрессия через числа: игрок начинает с минимального прироста ресурсов (например, одна единица за одно действие) и путём последовательной покупки улучшений доводит показатели до величин, превышающих стартовые на много порядков;

– автоматизация игровых действий: со временем ручные действия игрока замещаются автоматическими генераторами ресурсов, в том числе продолжающими работу в отсутствие игрока (*offline-прогресс*);

– простота управления при глубине стратегических решений: на уровне ввода взаимодействие ограничено, как правило, несколькими клавишами или кнопками мыши, в то время как содержательная сложность достигается за счёт планирования последовательных покупок и улучшений;

– наличие замкнутых петель прогресса вида «получение ресурса → покупка улучшения → увеличение скорости получения ресурса», обеспечивающих ощущение непрерывного роста;

– механика «перерождения» (*prestige/reset*), при которой игрок добровольно сбрасывает значительную часть накопленного прогресса в обмен на

особую валюту, используемую для приобретения принципиально более сильных улучшений в следующем прохождении.

Вышеперечисленные особенности делают жанр инкрементльных игр удобным для коротких игровых сессий и фонового запуска, а также обеспечивают игроку удовлетворение от наблюдения за ростом числовых показателей. Важным следствием математических особенностей инкрементальных игр является то, что они нередко используются в образовательных целях — для демонстрации принципов сложного процента, экспоненциального роста и основ алгоритмизации [1].

К наиболее известным представителям жанра относятся *Cookie Clicker*, *Adventure Capitalist*, *Universal Paperclips*, *Melvor Idle*, *Realm Grinder*, *Antimatter Dimensions*. Несмотря на внешнюю простоту, многие из перечисленных игр содержат математические модели значительной сложности, включающие десятки связанных между собой подсистем.

Концепция игры и референсный анализ

Разрабатываемая игра «Guardo’s Defense Quest» представляет собой инкрементальную игру с видом камеры сверху, в которой игрок управляет стражем замка по имени Гардо. Основной задачей игрока является оборона замка от волн наступающих противников и последовательное развитие системы улучшений (апгрейдов), позволяющей переходить к более высоким уровням сложности. По мере прохождения игрок собирает ресурсы и опыт у поверженных противников: ресурсы обрабатываются и расходуются на приобретение глобальных апгрейдов, а опыт преобразуется в очки навыков, распределяемые в дереве выбранного оружия.

Разрабатываемая игра относится к гибридному типу: основной игровой цикл строится на повторяющихся забегах по обороне замка (loop-based подход), стратегическое развитие обеспечивается деревом глобальных улучшений и прокачкой оружия (upgrades-focused подход), а долгосрочный прогресс

поддерживается механикой перерождения, характерной для игр с экспоненциальным ростом.

Ключевыми элементами игрового опыта, обеспечивающими удержание игрока, являются [1]:

– визуальная составляющая — интерактивная графика и выраженная обратная связь на действия игрока (эффекты попаданий, визуализация обработки ресурсов в стиле «машины Руба Голдберга»);

– возможность построения индивидуальных «билдов» — игрок свободно распределяет очки навыков в дереве оружия и приоритезирует ветви дерева глобальных апгрейдов, что приводит к различным стратегиям прохождения;

– постепенное «развёртывание» механик — на начальных этапах игроку доступно ограниченное число механик, новые открываются по мере продвижения по дереву улучшений и качественно изменяют игровой процесс.

Модель игрового цикла и игровых подсистем

Игровой процесс организован в виде замкнутого цикла, состоящего из следующих фаз:

– оборона замка — активная фаза, в которой игрок непосредственно управляет персонажем и отражает волны противников на прямоугольной арене;

– обработка ресурсов — переходная фаза, в которой накопленные за забег ресурсы преобразуются в игровую валюту;

– распределение улучшений — фаза, в которой игрок расходует валюту и очки навыков на глобальные апгрейды и дерево прокачки оружия;

– переход к следующему забегу либо, по достижении определённого порога ресурсов, добровольный сброс прогресса (перерождение) в обмен на особую валюту, открывающую доступ к усиленным мета-улучшениям.

Игровой цикл объединяет тактический уровень (внутренний цикл боя), уровень забега – основная последовательность фаз игрового процесса, и стратегический уровень – петля перерождения (рис.1).

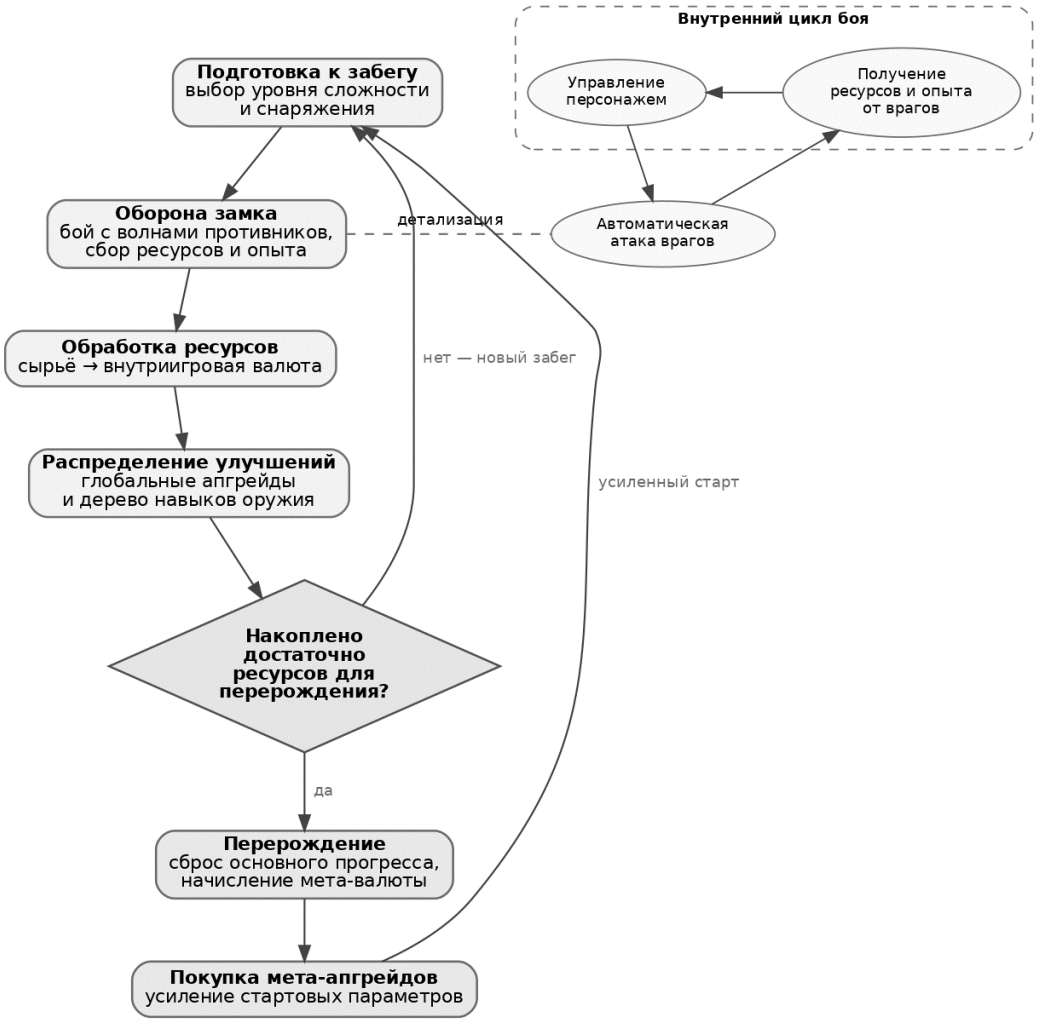

Рис.1 – Игровой цикл инкрементальной игры (авторский рисунок)

Внутренний цикл соответствует активной фазе боя и протекает на временных масштабах порядка нескольких секунд. Игрок управляет персонажем (перемещение по арене), автоматическая система прицеливания выбирает ближайшую цель и применяет текущее оружие, а уничтожение противника приводит к получению ресурсов и опыта [6].

Цикл забега объединяет четыре последовательные фазы: подготовку к забегу (выбор уровня сложности и снаряжения), оборону замка (внутри которой раскручивается внутренний цикл боя), обработку накопленных за забег ресурсов и распределение полученной валюты и очков навыков по дереву глобальных апгрейдов и дереву прокачки оружия. Блок решения выражает основное игровое решение — продолжить серию забегов с усиленным после улучшений

персонажем либо инициировать перерождение. Характерная продолжительность одного оборота этого цикла составляет от нескольких до десятков минут и определяется темпом игрового процесса.

UML диаграммы игровой системы

Диаграмма прецедентов. На основании референсного анализа разработана диаграмма прецедентов (use-case diagram) разрабатываемой игры (рис. 2) [9]. Диаграмма формализует функциональные требования к программному обеспечению, отражая полный набор вариантов использования, доступных игроку, и отношения между ними.

Варианты использования сгруппированы по шести функциональным пакетам, соответствующим подсистемам игры:

– пакет *«Управление игрой»* объединяет варианты использования, связанные с управлением игровой сессией: запуск игры, сохранение прогресса, загрузка сохранения, просмотр статистики;

– пакет *«Оборона замка»* включает варианты использования активной фазы игрового цикла: выбор оружия, управление персонажем, сражение с противниками, сбор ресурсов и опыта; вариант использования «Сражаться с врагами» включает в себя атаку противников, уклонение от их атак и защиту замка (отношение «включает»);

– пакет *«Обработка ресурсов»* объединяет варианты использования, связанные с переработкой сырья в игровую валюту; улучшение обрабатывающих машин включает в себя автоматизацию обработки;

– пакет *«Прокачка оружия»* содержит варианты использования, связанные с получением, распределением, а также свободным сбросом и перераспределением очков навыков, при этом вариант «Собирать ресурсы и опыт» пакета «Оборона замка» включает в себя повышение уровня игрока;

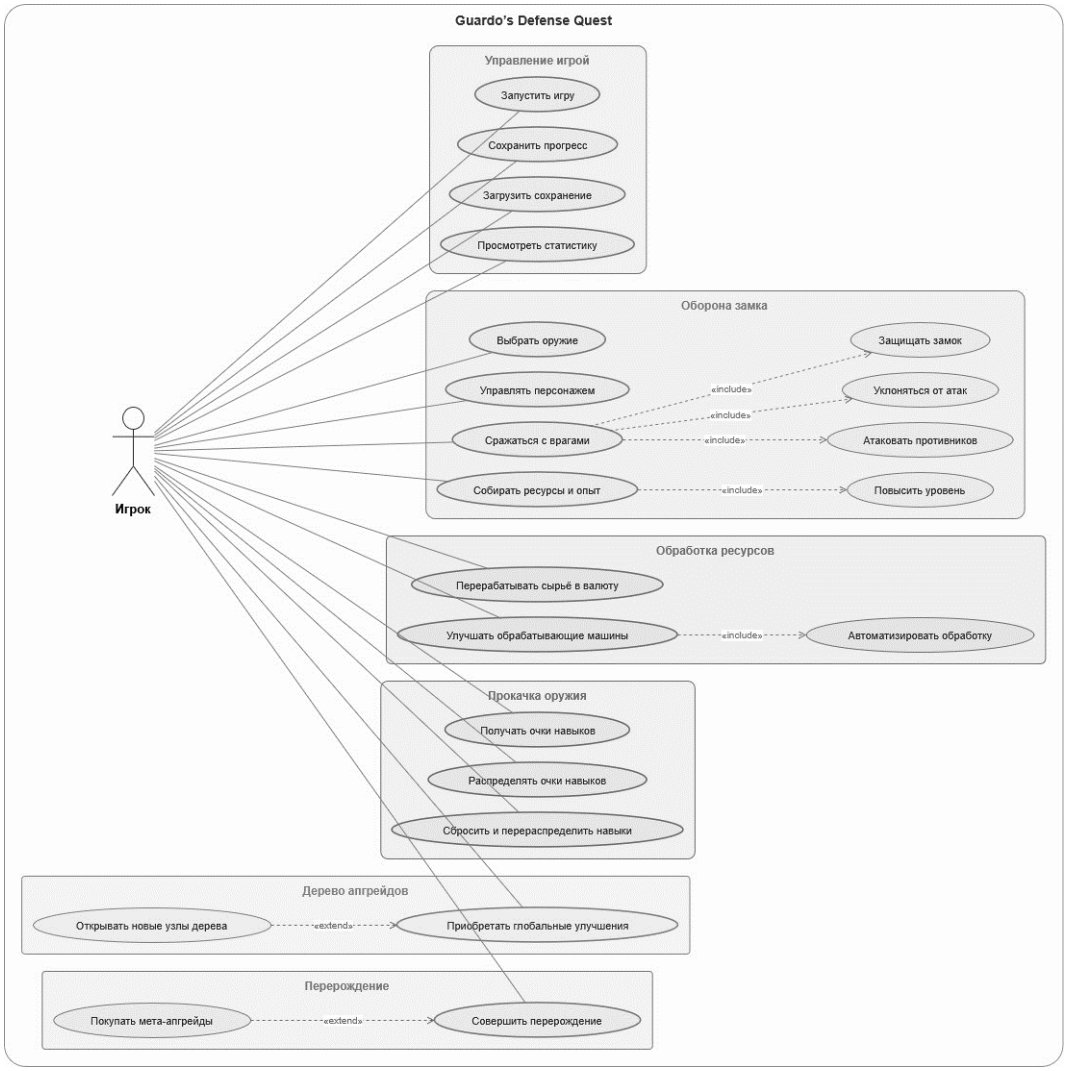

Рис. 2 – Диаграмма прецедентов игры (авторский рисунок)

– пакет *«Дерево апгрейдов»* описывает приобретение глобальных улучшений, причём каждая покупка расширяет набор доступных для открытия узлов дерева (отношение «расширяет»).

Диаграмма классов (рис. 3) формализует состав программных сущностей игры и статические связи между ними. Диаграмма классов служит основой для проектирования архитектуры программного обеспечения. При её построении учтены особенности выбранного игрового движка Unity и языка C#: классы снабжены стереотипами, отражающими их роль в архитектуре «Сущность — Компонент».

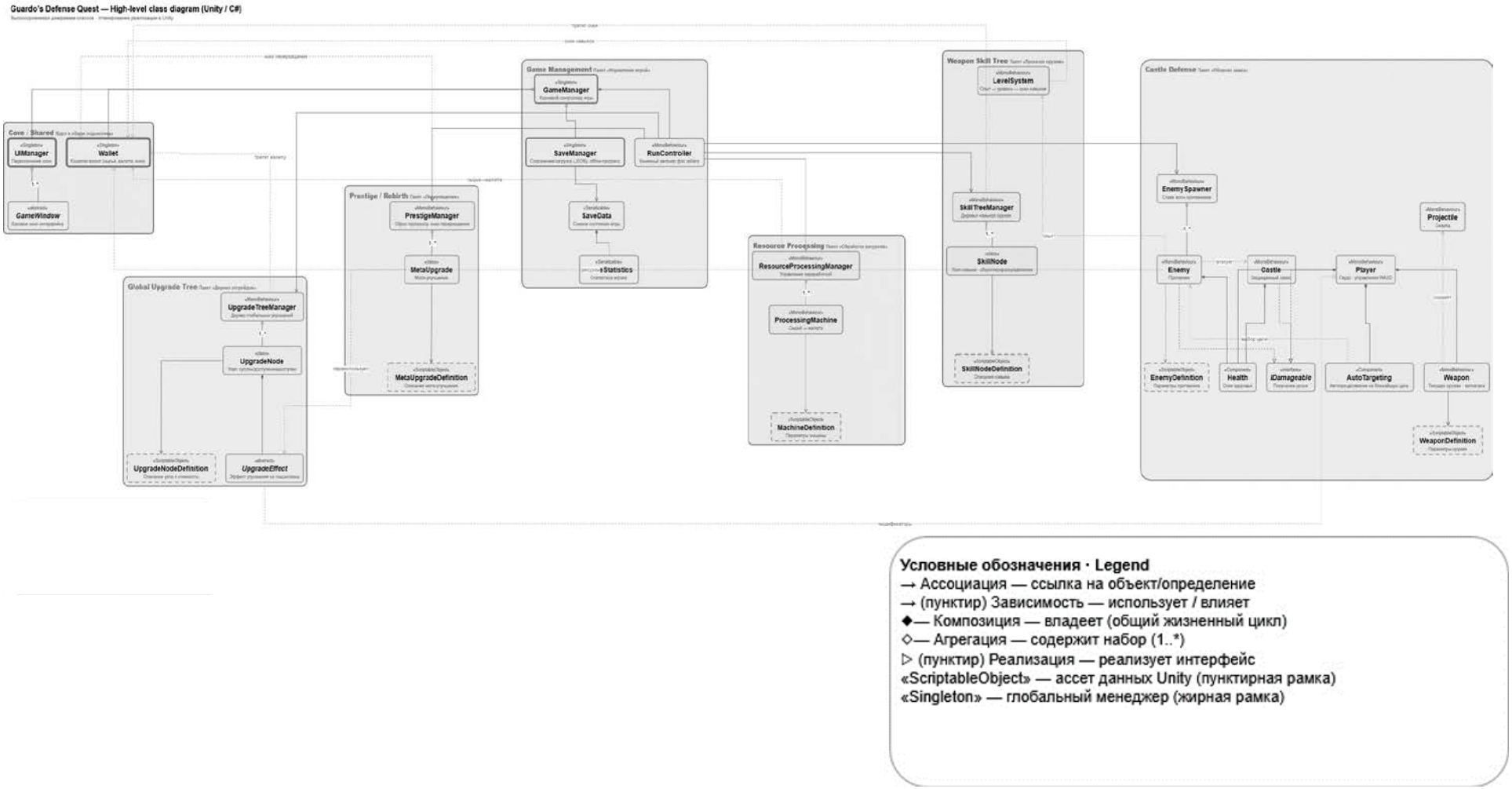

Рис. 3 – Диаграмма классов игры (авторский рисунок)

Использованы следующие стереотипы: «MonoBehaviour» — компоненты, присоединяемые к игровым объектам сцены и реализующие поведение во время выполнения; «ScriptableObject» — ассеты данных, хранящие конфигурацию сущностей, узлов деревьев и улучшений отдельно от логики; «Singleton» — глобальные менеджеры, существующие в единственном экземпляре; «interface» и «abstract» — абстрактные типы, задающие общий контракт для группы классов.

Классы сгруппированы в те же функциональные пакеты, что и варианты использования на рис. 2, что обеспечивает прослеживаемость требований — от вариантов использования к реализующим их программным сущностям:

– пакет *«Управление игрой»* включает корневой менеджер GameManager, конечный автомат фаз забега RunController и подсистему сохранения SaveManager с сериализуемыми классами SaveData и GameStatistics, обеспечивающими сохранение прогресса в формате JSON и расчёт offline-прогресса;

– пакет *«Оборона замка»* объединяет персонажа игрока Player (Гардо), его оружие Weapon и снаряды Projectile, систему автоприцеливания AutoTargeting, противников Enemy и их генератор EnemySpawner, а также защищаемый замок Castle; общий компонент Health и интерфейс IDamageable реализуются всеми поражаемыми сущностями;

– пакет *«Обработка ресурсов»* содержит менеджер ResourceProcessingManager и обрабатывающие машины ProcessingMachine, преобразующие накопленное сырьё в игровую валюту;

– пакет *«Дерево апгрейдов»* представлен менеджером UpgradeTreeManager, узлами дерева UpgradeNode и абстрактным классом эффекта улучшения UpgradeEffect;

– пакет *«Прокачка оружия»* объединяет менеджер SkillTreeManager, узлы навыков SkillNode и подсистему опыта и уровней LevelSystem, начисляющую очки навыков;

– пакет *«Перерождение»* включает менеджер PrestigeManager и мета-улучшения MetaUpgrade;

– ядро (Core/Shared) объединяет кошелёк игровых валют Wallet, менеджер пользовательского интерфейса UIManager и базовое окно GameWindow.

Каждому типу игровых сущностей, параметры которого подлежат балансировке, поставлен в соответствие класс-определение со стереотипом «ScriptableObject» (WeaponDefinition, EnemyDefinition, MachineDefinition, UpgradeNodeDefinition, SkillNodeDefinition, MetaUpgradeDefinition). Такой управляемый данными (data-driven) подход позволяет изменять баланс игры путём редактирования ассетов, без модификации исходного кода.

Математическая модель прогрессии игрового процесса

Рассмотрим ряд расчетных параметров игрового процесса [3; 5].

Боевая система. Боевая эффективность оружия определяется параметрами его определения (WeaponDefinition) и модификаторами из общего блока характеристик Stats, формируемого деревом улучшений. Урон одиночного снаряда dmg, темп стрельбы $rate$ (выстрелов в секунду) и интервал между выстрелами $cooldown$ вычисляются по формулам:

$$dmg = (baseDamage + FlatDamage) \cdot DamageMult$$

$$rate = baseFireRate \cdot FireRateMult$$

$$cooldown = 1/rate$$

Число снарядов за выстрел $count = projectileCount + ExtraProjectiles$.

Пробитие $pierce = basePierce + ExtraPierce$.

Урон в секунду по одиночной цели (без учёта перекрытия снарядов и пробития) равен: $DPS = rate \cdot$ dmg.

Коэффициенты $FireRateMult$ и $DamageMult$ возрастают линейно по сумме рангов соответствующих узлов дерева апгрейдов. Поэтому совокупная боевая мощь растёт мультипликативно относительно числа вложенных рангов. Это создаёт характерное для жанра ощущение ускоряющегося роста при линейных по виду улучшениях. Например, пистолет (*baseDamage* = 20, *baseFireRate* = 1) даёт 20 ед. урона в секунду; после прокачки «Базовой подготовки» до ранга 3 (+0,15 к *DamageMult* за ранг) и «Усиленных снарядов»

до ранга 5 (+0,1 за ранг) *DamageMult* = 1,95, а «Скорострельность» ранга 3 даёт *FireRateMult* = 1,45, что повышает урон в секунду до 56,6 (примерно в 2,8 раза).

Кривая стоимости улучшений. Дерево глобальных улучшений построено по ранговой модели: каждый узел приобретается многократно, повышая ранг r от 0 до *maxRank*. Стоимость перехода на ранг r задаётся функцией $C(r)$ (метод UpgradeNodeDefinition.CostForRank):

$$C(r) = baseCost \cdot costGrowth^{(r-1)}, r = 1 \dots rankMax$$

Суммарная стоимость прокачки узла до ранга R есть сумма геометрической прогрессии:

$$C_R = \sum_{r=1}^{R} C(r) = \frac{baseCost \cdot (costGrowth^R - 1)}{costGrowth - 1}$$

Геометрический рост цены является основным «клапаном слива» валюты для балансирования игрового процесса: он не позволяет игроку мгновенно выкупить дерево и растягивает прогресс, согласовывая его с растущим доходом за забег. Например, для узла «Базовая подготовка» (*baseCost* = 50, *costGrowth* = 1.6, *maxRank* = 3): C(1) = 50, C(2) = 80, C(3) = 128, суммарно 258 единиц валюты. Кривые стоимости рангов приведены на рисунке 4.

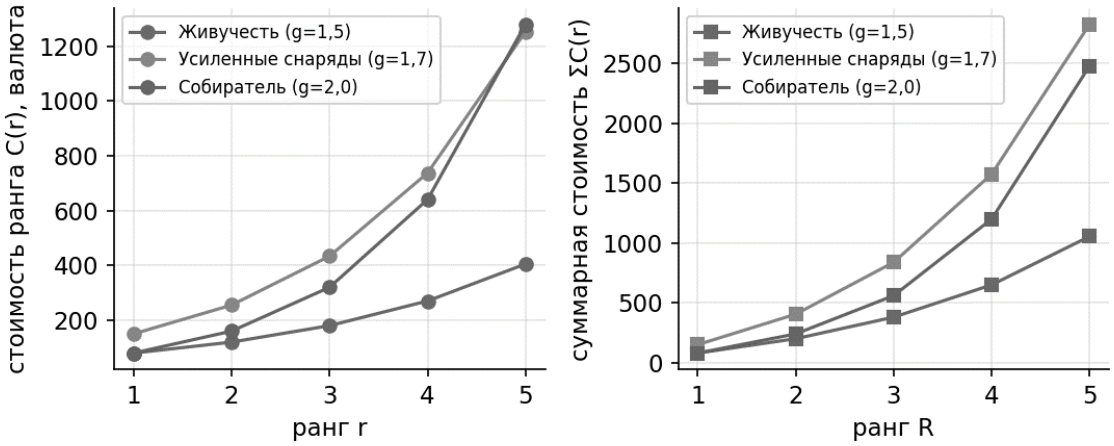

Рис.4 – Кривые стоимости рангов (авторский рисунок)

Кривая роста эффективности. Эффект узла линейно зависит от ранга: вклад узла в характеристику равен произведению $effectPerRank \cdot rank$. Все вклады агрегируются в едином блоке *Stats* суммированием по всем узлам с рангом больше нуля:

$$Stat = base + \sum effectPerRank \cdot rank$$

Тип эффекта задаётся перечислением UpgradeEffectType (13 типов: плоский и процентный урон, скорострельность, пробитие, дополнительный снаряд, HP Гардо и замка, скорость передвижения, регенерация Гардо и замка, сырьё за убийство, радиус подбора, коэффициент конвертации).

Полиморфная иерархия классов-эффектов не используется: применение эффекта реализовано выбором по типу (метод UpgradeEffect.Apply), что упрощает сериализацию и редактирование баланса в ассетах (управляемый данными подход). Сравнение линейного вклада характеристики и мультипликативного роста боевой мощи приведено на рис.5.

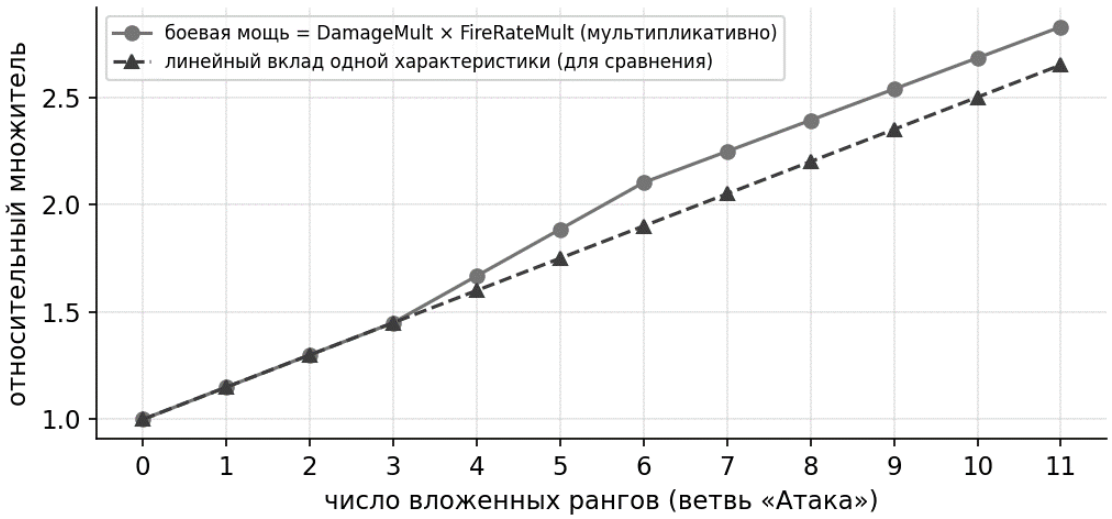

Рис. 5 – Линейный вклад одной характеристики и мультипликативный рост боевой мощи по мере вложения рангов (авторский рисунок)

Экономика забега. Экономика забега двухступенчатая. За забег накапливается сырьё; уничтоженный противник роняет сферы сырья в количестве:

$$drop = resDropAmount \cdot (1 + ResPerKillBonus) \cdot runResMultiplier$$

где *runResMultiplier* — множитель выбранного уровня сложности (выше уровень — больше сырья). Ресурсы за забег суммируются по всем уничтоженным противникам и по потерянным артефактам, автоматически

зачисляемым в конце забега. Ресурсы конвертируется в валюту (метод Converter.Convert): $currencyGain = S \cdot ConversionRate$.

В реализованном вертикальном срезе игре конвертация выполняется мгновенно, однако коэффициент *ConversionRate* сделан улучшаемым (узел «Переработка»), что сохраняет двухступенчатую экономику как точку дальнейшего роста. Ресурсы начисляются при любом исходе забега — поражение не штрафуется потерей собранных ресурсов.

Баланс темпа игровой сессии. Темп сессии определяется отношением дохода за забег $currencyGain$ к стоимости очередных рангов $C(r)$. Линейный рост дохода (через полученные ресурсы, множитель уровня и коэффициент конвертации) уравновешивается геометрическим ростом цен, благодаря чему один забег оплачивает несколько рангов на ранней стадии и доли ранга на поздней.

Структурно дерево апгрейдов состоит из корневой вершины «Базовая подготовка» с тремя тематическими ветвями (атака, выживание, экономика). Дочерний узел открывается, когда ранг родителя не меньше единицы. Зависимость множителя боевой мощи от суммарно потраченной валюты показана на рис. 6.

Реализация ключевых алгоритмов

Игровой цикл реализован в виде конечного автомата. Класс RunController реализует последовательность фаз: подготовка к забегу (выбор уровня и оружия), оборона, итоги (конвертация сырья в валюту) и распределение улучшений, после чего цикл повторяется. Условие победы — все волны уровня порождены и на сцене не осталось противников (рис.7).

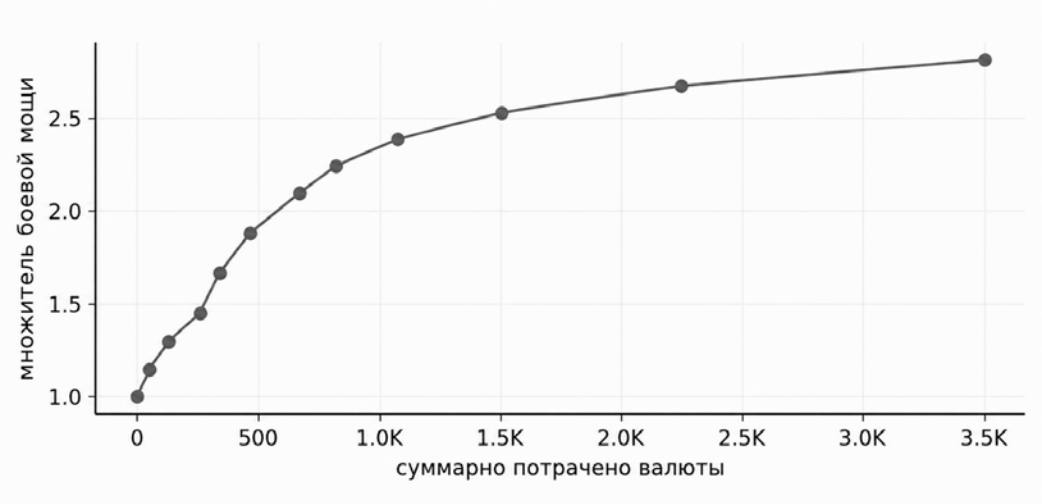

Рис.6 – Зависимость множителя боевой мощи от суммарно потраченной валюты – темп прогрессии (авторский рисунок)

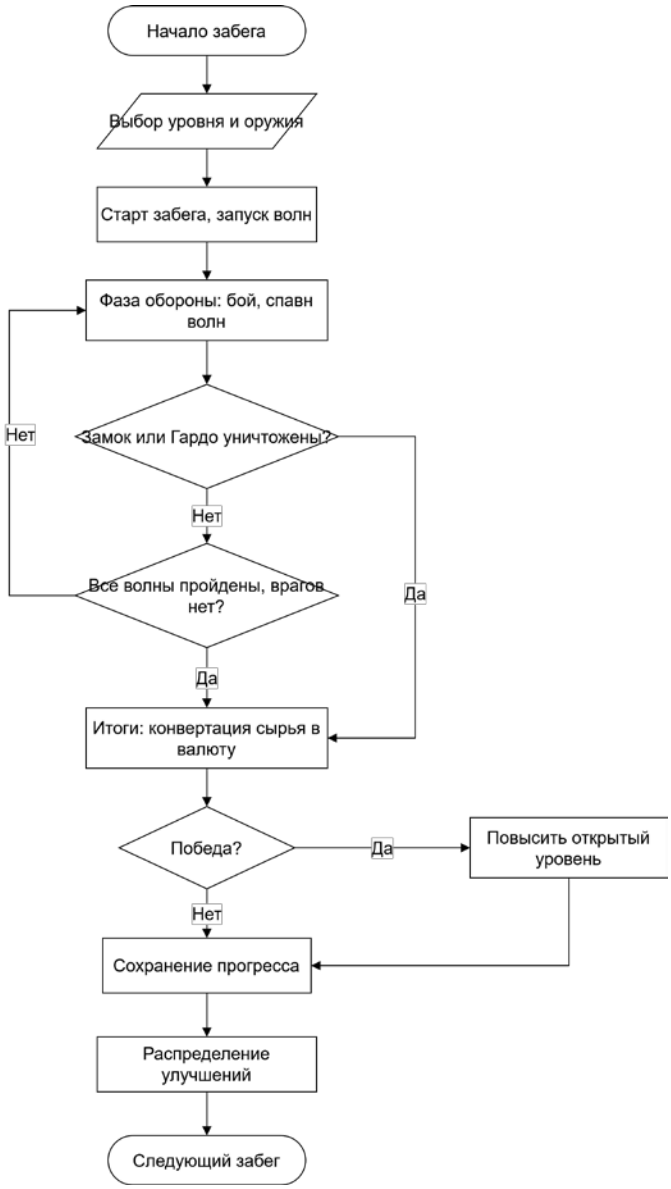

Рис.7 – Блок-схема основного игрового цикла (авторский рисунок)

Поражение наступает при наступлении одного из двух событий – гибели Гардо или замка. При любом завершении забега собранные ресурсы конвертируется и зачисляются в кошелёк; при победе повышается значение наибольшего открытого уровня, после чего выполняется сохранение.

Верификация разработанных моделей

Реализован вертикальный срез игры в игровом движке Unity. На рис.8 и 9 представлены – основное меню игры и реализация игрового процесса с пояснениями.

Рис.8 – Основное меню игры (авторский рисунок)

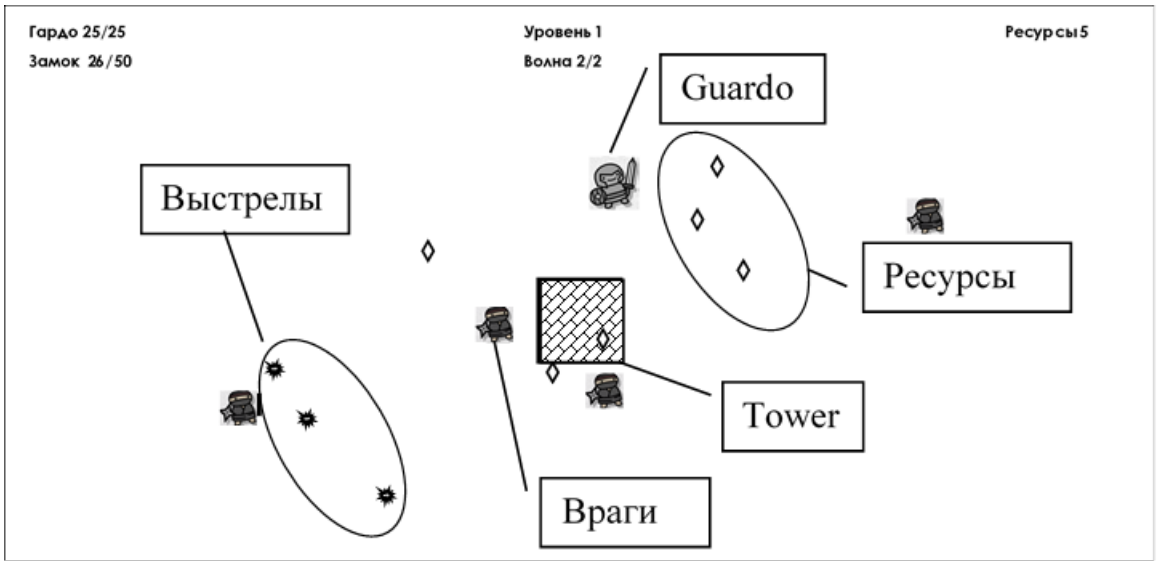

Рис.9 – Реализация игрового процесса с пояснениями (авторский рисунок)

Корректность математической модели прогрессии проверена методом контрольных сценариев. Каждый сценарий выполнялся детерминированно в редакторе Unity (режим Play): на вход подавались известные значения,

наблюдаемый результат сверялся с ожидаемым, вычисленным в соотвествии с математической моделью.

Контрольные сценарии охватывают: кривую стоимости $C(r)$; агрегацию эффектов рангов в блоке *Stats*, в том числе суммирование вкладов нескольких узлов; гейтинг разблокировки по рангу родителя, включая отказ покупки запертого узла; экономику преобразования ресурсов в валюту и эффект узла «Переработка»; поток фаз забега и персистентность сохранения, включая структуру JSON и кросс-сессионную загрузку; а также сквозную играбельность уровня. Во всех сценариях наблюдаемые значения совпали с ожидаемыми.

Вывод

Рассмотрена модель игрового цикла инкрементальной компьютерной игры «Guardo’s Defense Quest». Разработаны информационные модели компьютерной игры: UML диаграммы – диаграмма прецедентов и диаграмма классов. Разработана математическая модель прогрессии инкрементальной игры. Представлена схема алгоритма игрового процесса. Разработан вертикальный срез игры. Проведена верификация разработанных моделей игры, показавшая их корректность.

Библиографический список

1. How to Make an Idle game: Everything You Need to Know About Incremental Mobile Games. URL: https://www.gameanalytics.com/blog/how-to-make-an-idle-game-adjust (дата обращения: 20.06.2026).
2. Горовых И. И., Евич Л. Н. Описание баланса и механики мобильного приложения жанра Tower Defense // Молодой исследователь Дона. 2019. №5 (20). URL: https://cyberleninka.ru/article/n/opisanie-balansa-i-mehaniki-mobilnogo-prilozheniya-zhanra-tower-defense (дата обращения: 19.06.2026).
3. Ельчанинов М.Н. Оценка ресурсных затрат и оптимизация стратегий поэтапного улучшения игрового снаряжения на основе поглощающих цепей Маркова // Дневник науки. №6. 2026. URL: https://dnevniknauki.ru/images/publications/2026/6/technics/Elchaninov.pdf (дата обращения: 22.06.2026).

4. Колдаева Е.Ю. Математика игрового баланса на примере инкрементальных игр // Репозиторий БГУИР. URL: https://libeldoc.bsuir.by/handle/123456789/60352 (дата обращения: 20.06.2026).
5. Сахипов, А. Э. Разработка игры «Симулятор огородника» в жанре Tower Defense на платформе Unity / А. Э. Сахипов // Мавлютовские чтения : Материалы XIX Всероссийской молодёжной научной конференции. В 8-ми томах, Уфа, 24–28 ноября 2025 года. – Уфа: Уфимский университет науки и технологий, 2025. – С. 1283-1287. – EDN HCCQUQ.
6. Силин, К. К. Обучающие игры: возможности и сферы применения / К. К. Силин // Мавлютовские чтения : Материалы XVIII Всероссийской молодёжной научной конференции. В 9-ти томах, Уфа, 25–29 ноября 2024 года. – Уфа: Уфимский университет науки и технологий, 2024. – С. 953-959. – EDN ZSXRKA.
7. Соболева Е. В., Соколова А. Н., Исупова Н. И., Суворова Т. Н. Применение обучающих программ на игровых платформах для повышения эффективности образования // Вестник Новосибирского государственного педагогического университета. – 2017. – Т. 7, № 4. – С. 7-25. – DOI 10.15293/2226-3365.1704.01. – EDN ZFRMWN.
8. Соловьев И.В. Инкрементная компьютерная деловая игра как технология обучения // ИТС. 2015. №2 (79). URL: https://cyberleninka.ru/article/n/inkrementnaya-kompyuternaya-delovaya-igra-kak-tehnologiya-obucheniya (дата обращения: 25.06.2026).
9. Цифровые технологии и искусственный интеллект в организационно-технических системах / А. В. Воробьев, М. А. Верхотуров, С. В. Тархов [и др.]. – Уфа : Уфимский университет науки и технологий, 2024. – 258 с. – ISBN 978-5-7477-5910-7. – EDN LVMBMB.