??????

Vision & Scope

??????-????? — Vision & Scope

1. ???????? ? ????????

На сервере существует режим строительства с ранговой системой, где строительство — это измеримый навык: игрок проходит оценку, получает ранг и открывает новые возможности. Паркур-режим расширяет эту концепцию на другой аспект Minecraft.

?????? ?????? ??? ?????? ???????????:

- Добавляет второй измеримый навык — соревновательное прохождение уровней с прогрессией, которая сохраняется между сессиями.
- Создаёт «режим отдыха» — метаигру между сессиями строительства, снижающую усталость от основного режима.
- Служит переходной стадией перед разработкой основного режима сервера, в который механика процедурной генерации уровней органично перерастёт. В дальнейшем паркур-режим станет частью метаигры основного режима.

2. ??????? ?????????

Первичная аудитория — игроки, уже присутствующие на сервере в строительном режиме. Им знакома ранговая система, они понимают ценность прогрессии. Паркур предлагает им альтернативное занятие без необходимости менять сервер.

Вторичная аудитория — новые игроки, которых привлекает именно паркур как жанр. Для них режим является точкой входа на сервер.

3. ???????? ??????? ???? (Core Loop)

Вход в режим → Лобби
↓
Автоматическая генерация уровня (по текущему рангу игрока)
↓
Прохождение уровня
↓ (успех) ↓ (провал)
Получение опыта Нет штрафа (обычный уровень)
↓
Накопление опыта → порог ранга
↓
Ранговый экзамен:
уровень повышенной сложности

с механиками следующего ранга
↓ (успех) ↓ (провал)
Получение нового ранга Штраф: снятие части опыта
Разблокировка механик Повторная попытка позже

?????? ?????? ?????:

- При входе в режим уровень генерируется **автоматически** — игрок попадает в него без ручного запуска.
- Штраф за провал (снятие опыта) действует **только на ранговом экзамене**, не на обычных уровнях. Это снижает барьер для casual-игроков и сохраняет напряжение именно в ключевой момент.
- Ранговый экзамен **заменяет** очередной обычный уровень при достижении порога опыта — отдельного вызова нет.
- При выходе из режима посреди уровня: **уровень удаляется**, при повторном входе игрок появляется в **лобби** и получает новый сгенерированный уровень. Прогресс незавершённого уровня не сохраняется.

4. ??????????? ???????

Режим состоит из четырёх полностью независимых систем. Каждая система разрабатывается, тестируется и обновляется отдельно. Взаимодействие между ними происходит **только через явно описанные контракты** (см. документ Subsystems Spec).

Система	Ответственность	Знает о других системах
Ранговая система	Прогрессия, очки, экзамены, награды	Нет
Процедурная генерация уровня	Построение геометрии по параметрам	Нет
Парсер уровней в конфиги	Перевод построенных карт в данные генерации	Нет
Система создания мира	Разворачивание мира со сгенерированным уровнем	Нет

Единственная точка связи между системами — **Subsystems Spec**, где описан маппинг ранг → параметры генерации.

5. Out of Scope — ?????? 1.0

Следующие возможности явно исключены из текущей разработки. Их добавление требует отдельного решения и обновления документации:

- Кооперативный паркур (несколько игроков на одном уровне)
- Редактор уровней для игроков
- Сезонная система / временные ивенты
- Рейтинговые таблицы между игроками (лидерборд)

- Повтор / запись прохождения

6. ???????? ??????

Режим считается успешным при достижении следующих показателей:

- **Retention D3:** не менее 40% игроков, попробовавших режим, возвращаются на третий день.
- **Конверсия в ранг 2:** не менее 60% игроков, начавших режим, достигают второго ранга.
- **Среднее число попыток на экзамен:** не более 5 — сигнал о сбалансированной сложности.
- **Время сессии:** средняя сессия 15–30 минут — режим работает как «отдых», а не затягивает на часы.

7. ??????? ????????????? ??????

Ни одна из четырёх систем не импортирует, не вызывает и не знает о внутреннем устройстве другой. Любое взаимодействие между системами проходит через контракт, описанный в Subsystems Spec. Нарушение этого принципа в коде требует обсуждения и обновления документации до мержа.

Этот принцип обеспечивает возможность замены любой системы (например, генератора уровней) без затрагивания остальных частей режима.

?????????? ??????? ?????????

? ?????????

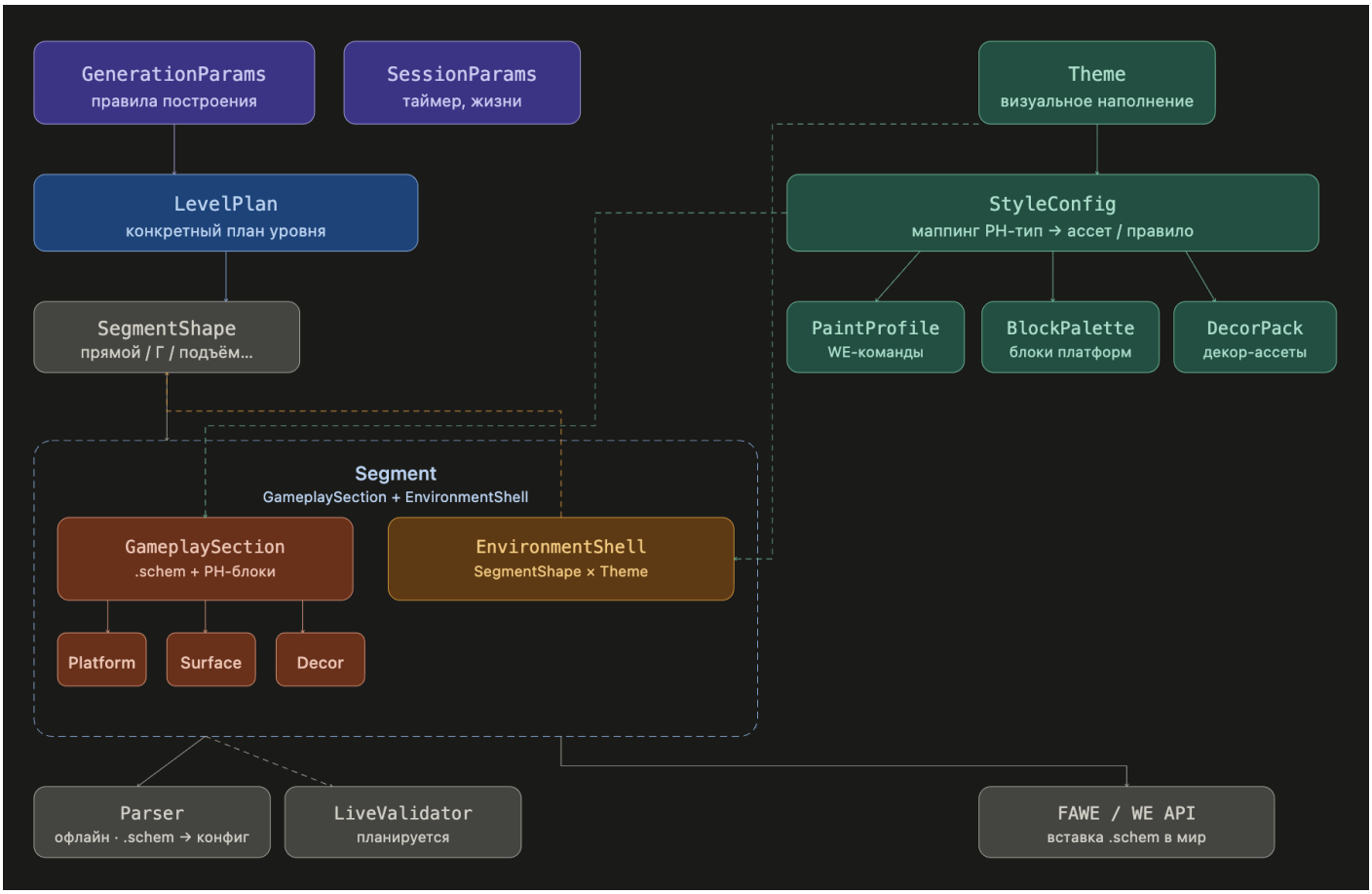
Этот документ фиксирует **единый словарь** компонентов системы процедурной генерации. Его цель — устранить разночтения: когда разработчик и билдер используют слово «секция» или «форма», они должны понимать под ним одно и то же.

Важно: система спроектирована независимо от конкретного режима. В терминах намеренно отсутствуют слова «ранг», «паркур», «экзамен» — это понятия игрового режима, а не системы генерации. Система генерации принимает параметры на вход и строит мир на выход, не зная зачем.

Документ будет обновляться многократно. В частности, подтипы плейсхолдеров (PH) — предмет отдельной дискуссии и отдельного документа (Placeholder Spec); здесь зафиксирован только верхний уровень.

????? ???????????

Иерархия от входных параметров до финальной геометрии



??????? ??????????

GenerationParams

Набор правил, определяющий **что генерировать**. Не привязан к конкретному уровню — это профиль параметров, который подаётся генератору на вход.

Включает:

- Общая сложность, минимальная и максимальная сложность сегмента
- Доступные типы сегментов
- Минимальная и максимальная длина уровня (количество сегментов)
- Доступные темы
- Правила размещения тем: минимум 1, максимум 2 темы на уровень; одна тема занимает минимум 5 сегментов подряд перед сменой

GenerationParams **не знает** ничего о таймерах, жизнях и других правилах игровой сессии — это зона ответственности SessionParams.

SessionParams

Набор правил, определяющий **как проходит игровая сессия**. Это runtime-параметры, не параметры построения геометрии.

Включает: таймер, количество жизней, прочие геймплейные теги настройки сессии.

> **Почему GenerationParams и SessionParams разделены:** построение геометрии и правила игры — разные ответственности. Генератор читает только GenerationParams и не должен знать о таймере. Это сохраняет принцип независимости систем.

????????? ??????

LevelPlan

Конкретный результат применения GenerationParams к одному запуску. Если GenerationParams — это правила, то LevelPlan — это уже принятые решения для конкретного уровня.

Включает: расстановку форм (SegmentShape), их порядок, типы, длину, назначенные темы. Генерируется один раз при создании уровня.

SegmentShape

Форма сегмента — компонент геометрии. Определяет габариты и направление куска уровня, но не его содержимое.

Примеры форм: квадрат, прямоугольник, вытянутый прямоугольник, угол Г, вытянутый угол Г, вертикальный подъём (короткий / длинный), падение (короткое / длинное).

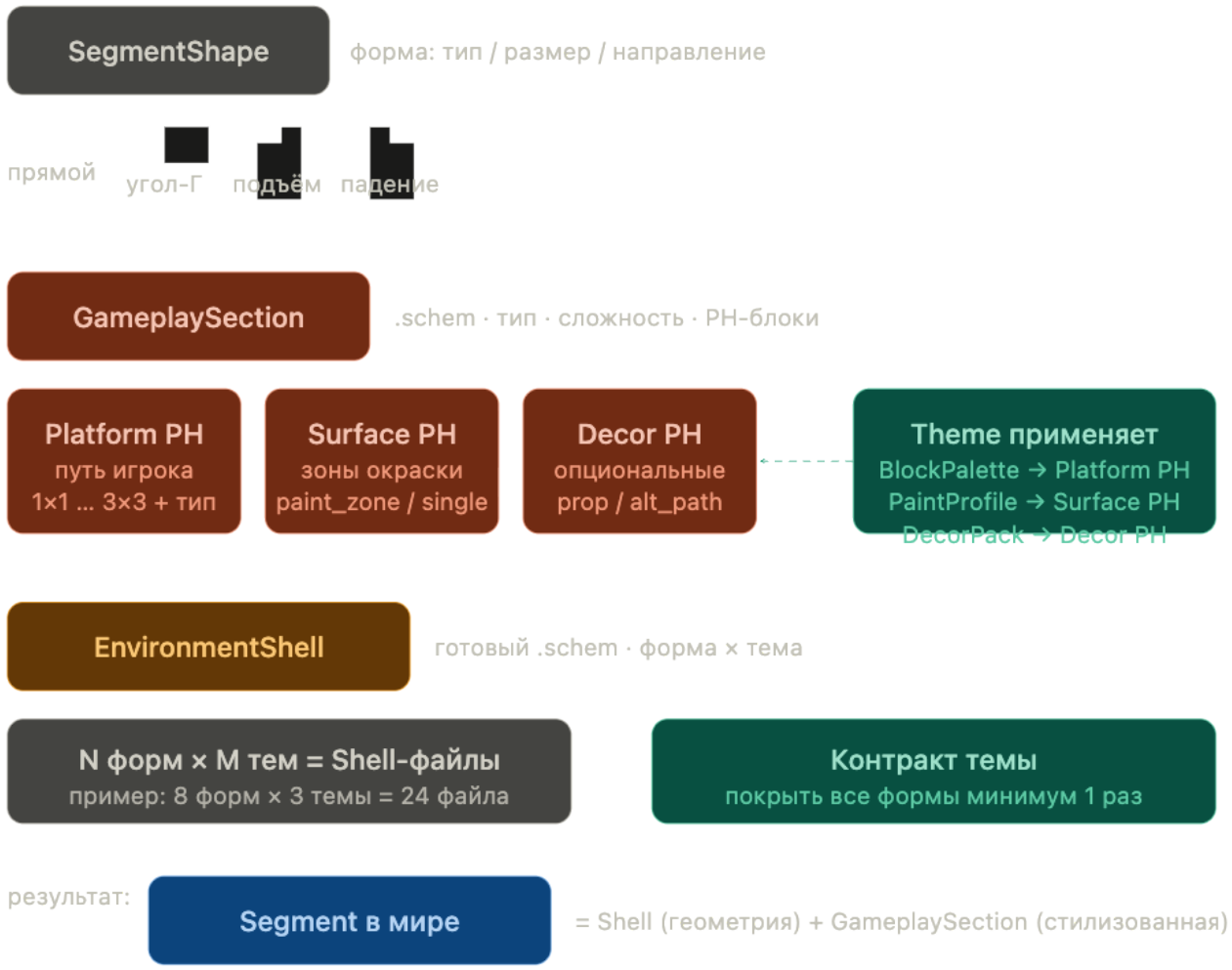
Количество форм намеренно ограничено — это позволяет вручную поддерживать оболочки (EnvironmentShell) под каждую форму без чрезмерной нагрузки на билдеров.

Segment

Физическая единица уровня в мире — **занятая форма**. Состоит из двух частей: геймплейной (GameplaySection) и оболочки (EnvironmentShell).

> **Каноническое имя:** в системе используется термин **Segment**, не «секция». Ранее в обсуждениях встречалось слово «секция» как синоним — оно выводится из употребления во избежание ложного ощущения вложенности. Segment = одна геймплейная часть + одна оболочка по форме, заданной в LevelPlan.

Сегменты соединяются вплотную, образуя один непрерывный маршрут.



?????????? ????????

GameplaySection

Геймплейная часть сегмента — то, по чему игрок проходит. Хранится как `.schem` файл, размеченный блоками-плейсхолдерами (PH).

Каждая GameplaySection имеет:

- Тип (определяет характер геймплея)
- Сложность
- `.schem` с плейсхолдерами

Один Segment содержит ровно одну GameplaySection.

EnvironmentShell

Оболочка сегмента — стены, пол, потолок вокруг геймплейной части. Не содержит геймплейной логики, только визуальное обрамление.

Shell хранится как готовый `.schem` файл, привязанный к **паре** `SegmentShape × Theme`. Процедурно не генерируется — собирается вручную.

> **Формула объёма контента:** поскольку каждая тема обязана покрывать все формы (контракт темы), число Shell-файлов = `количество форм × количество тем`. Например, 8 форм × 3 темы = 24 Shell-файла как минимум. Это не ограничение, а планируемый объём работы: добавление одной новой темы означает создание Shell под все формы сразу, а не одного файла.

Shell делать проще, чем геймплейные части, и их количество меньше — поэтому ручное масштабирование по темам оправдано.

?????????? (PH)

Блоки-маркеры внутри `.schem` GameplaySection. Парсер читает их и применяет правила генерации из темы, заменяя плейсхолдеры на финальные блоки и объекты.

Верхний уровень типов:

Тип PH	Назначение
Platform PH	Геймплейный путь игрока — платформы, по которым он прыгает
Surface PH	Поверхности под стилизацию (окраску)

Тип PH	Назначение
Decor PH	Опциональные декоративные объекты

> **Подтипы — отдельная дискуссия.** Каждый тип PH имеет подтипы (например, у Platform PH: опорные, парящие, наклонные, тонкие заборы и т.д.). Подтипы определяются в конфиге, а не захардкожены. Полный список типов и подтипов фиксируется в отдельном документе — **Placeholder Spec** — до начала разработки первого Theme Pack. Здесь намеренно зафиксирован только верхний уровень.

Принцип PH как контракта: набор типов и подтипов PH — это контракт между билдером и генератором. Он должен быть зафиксирован до создания первой темы и первой GameplaySection. Иначе каждый билдер изобретает свои блоки, и система рассыпается.

???? ? ???????????

Theme

Визуальное наполнение сегмента — определяет, как выглядит уровень. Знает **только свои ассеты** и ничего не знает о структуре уровня или правилах смены тем (это зона GenerationParams).

Контракт темы: каждая тема обязана покрывать все формы (для Shell) и все типы PH минимум по одному варианту. Тема с неполным покрытием не может быть зарегистрирована.

StyleConfig

Коллекция маппингов внутри темы: какой PH-тип → какой ассет или правило генерации. Это «оглавление» стилизации темы.

```
Theme
 └── StyleConfig          — коллекция маппингов PH → ассет/правило
      ├── PaintProfile    — упорядоченный список WE-команд для paint_zone
      ├── BlockPalette    — блоки/ассеты для Platform PH
      └── DecorPack       — ассеты для Decor PH
```

PaintProfile

Упорядоченный список команд WorldEdit для стилизации paint_zone. **Порядок команд важен** — команды выполняются последовательно и зависят от результата предыдущих.

Пример последовательности:

```
//replace <0&1 2          # блок под воздухом (камень-плейсхолдер) → трава
//replace 1 50%3,50%3:1   # оставшийся камень → смесь земли и грубой земли
```

PaintProfile применяется по региону сегмента (а не всего уровня) — это необходимо для поддержки нескольких тем на одном уровне.

BlockPalette

Набор блоков и ассетов темы для генерации Platform PH (опоры, платформы, фундаменты).

DecorPack

Набор декоративных ассетов темы под Decor PH. Хранит ассеты под разные размеры слотов.

Поведение при отсутствии ассета: если в DecorPack нет ассета под нужный размер или подтип — генератор **пропускает** этот плейсхолдер с записью в лог, а не падает. Это требование надёжности: неполный DecorPack не должен ломать генерацию. При этом контракт темы требует покрытия всех типов минимум по разу, так что пропуск — исключение, а не норма.

??????????

Parser

Независимый инструмент обработки `.schem`. Работает офлайн — запускается отдельно от игрового сервера.

Делает две вещи:

- Читает `.schem` и строит конфиг GameplaySection
- Валидирует корректность: все блоки внутри paint_zone входят в разрешённый список; тема покрывает все используемые PH

Parser — отдельная система, не часть генератора рантайма.

LiveValidator

Планируется. Алгоритм работы не определён.

Желаемый инструмент для билдера: проверка `.schem` прямо во время строительства, с визуальным фидбеком в живом мире (подсветка проблемных PH, превью с темой). Это **отдельный инструмент** от Parser — разные триггеры запуска (Parser обрабатывает файл, LiveValidator работает с живым миром). Не следует смешивать их в одной реализации.

???????? ???????

Следующие пункты не зафиксированы и требуют отдельных решений:

- **Подтипы PH** — полный список для всех трёх типов (Placeholder Spec)
- **Формат paint_zone** — набор из 5 блоков или два угловых маркера определяют регион
- **Список разрешённых блоков-маркеров** под каждый тип PH
- **Алгоритм LiveValidator** — как именно проверять при строительстве
- **Кандидаты v2** — подтипы вроде `ceiling_hang` (потолочные платформы): входят в первую версию или откладываются

---

*Связанные документы: **Placeholder Spec** (типы и подтипы PH), **Level Generation Spec** (pipeline генерации), **Theme Pack Spec** (требования к теме).*