

Нижe — 3 документа в одном сообщении: **PRD для продакта**, **Tech Design Doc для тимлида**, **TЗ+план работ для разработчиков** (с Definition of Done/Ready и задачами, которые можно прямо переносить в Jira/YouTrack). MoSCoW использую как метод приоритизации требований (Must/Should/Could/Won't).^{[1](#)}

1) PRD (для продакта)

1.1. Визия и цель

Мы делаем браузерную аркадную гонку, где “машины” заменены на змей, а цель — быстрее всех пройти трассу и попасть в лидерборд. Критерии успеха в оценке проекта: плавность управления, стабильный FPS (≥ 60), отсутствие критических багов, чистая архитектура и качественная презентация.^{[2](#)}

1.2. Платформа и текстек (продуктовые решения)

- Клиент: Phaser 3 (HTML5 2D framework) для быстрого прототипирования, спрайтов, тайлмапов, UI. [https://phaser.io\[^4\]](https://phaser.io[^4])
- Онлайн матчи (до 4): Colyseus (комнаты, синхронизация состояния, серверная логика матча). [https://docs.colyseus.io\[^5\]](https://docs.colyseus.io[^5])
- Лидерборды: Nakama (leaderboards), делаем authoritative, чтобы клиент не мог отправлять результаты напрямую. [https://heroiclabs.com/docs/nakama/concepts/leaderboards/\[^6\]](https://heroiclabs.com/docs/nakama/concepts/leaderboards/[^6])

1.3. Режимы игры

- Offline: 1 игрок + 3 NPC на трассе, фиксируем время, пишем в лидерборд (если есть сеть) или локально с последующей синхронизацией.
- Online: 1–4 игрока в комнате, сервер авторитарно считает гонку, результаты пишутся в серверный лидерборд.

1.4. Основные фичи (MoSCoW)

MoSCoW — метод приоритизации требований по Must/Should/Could/Won't.^{[3](#)}

Must (MVP обязателен):

- 3 уникальные трассы, старт/финиш, логика выезда за трассу.^{[2](#)}
- Таймер прохождения, таблица змей с временем, интуитивный интерфейс, спидометр.^{[2](#)}
- 4 вида змей (разные характеристики) + экран выбора.
- 4 пикапа (2 бафа + 2 минуса/бомба).
- Offline 1+3 NPC.
- Online до 4 игроков + серверный лидерборд.

Should (после MVP, но желательно до защиты):

- “Линька” (сброс длины ради ускорения) как уникальная механика.^{[2](#)}
- “Терморесепторы” (тепловая карта/идеальная линия) как инновация.^{[2](#)}
- Качественные эффекты/звук на ключевые события.^{[2](#)}

Could (если успеваеете):

- Подбор “набора” косметики (шапки/паттерны), дополнительные режимы (Time Trial).
- Реплеи “призрака” лучшего заезда.

Won't (не делаем в этой итерации):

- Сложная физика хвоста по всей длине, полноценный pathfinding по navmesh, кросс-платформенные аккаунты с соц-логинами.

1.5. UX-скелет (экраны)

- Main Menu: Play Offline, Play Online, Leaderboards, Settings.
- Snake Select: 4 карточки змей (статы + “фишка”).
- Track Select: 3 трассы (превью + лучший результат).
- Race HUD: таймер, спидометр, позиция/круг (если нужен), мини-лидерборд, активный эффект (иконка+таймер).
- Results: времена всех участников, кнопки “Retry/Next Track/Back”.

1.6. Метрики качества (что меряем)

- FPS: 60+ на “тестовой машине” (вы определяете конфиг), без фризов >50 мс.
- Input latency: управление не должно “вязнуть” в offline; в online — минимальная заметная задержка (предикция+коррекция).
- Crash-free: 10 минут непрерывной гонки без падений.
- Честность результатов: authoritative leaderboard, клиент не может подделать время.⁴

2) Tech Design Doc (для тимлида)

2.1. Принципы архитектуры (под code review)

- Разделение: Input → Simulation → Networking → Rendering → UI.
- Симуляция детерминированная по фиксированному тику.
- В онлайн сервер авторитарен: клиент отправляет только input, сервер обновляет state и синхронизирует. Colyseus показывает паттерн “сервер мутирует state, клиент слушает изменения”. [https://docs.colyseus.io/state/\[^7\]](https://docs.colyseus.io/state/[^7])

2.2. Компоненты системы (модули)

Client /client (TS):

- input/: Keyboard/Gamepad, нормализация в InputCommand.
- sim/: оффлайн симуляция (тот же код, что на сервере, если возможно).
- net/: Colyseus client, буфер команд, интерполяция/коррекция.
- render/: Phaser сцены, камеры, слой трассы, слой змей, VFX.
- ui/: HUD, меню, лидерборды.
- assets/: atlas, tilemaps, audio.

Match Server /match-server:

- rooms/RaceRoom: lifecycle комнаты, matchmaking.
- state/RaceStateSchema: snakes/pickups/racePhase/timers.
- sim/: authoritative симуляция, обработка коллизий и эффектов.
- anti-cheat/: серверная валидация input (скорость поворота/частота ability).
- results/: фиксация времени и отправка в Nakama.

Backend /backend:

- Nakama (docker), runtime hooks/RPC для сабмита результатов.
- Конфиг leaderboards по трассам.

Shared /shared:

- types/: SnakeStats, EffectType, TrackData, InputCommand, Snapshot.
- config/: snakes.json, perks.json, tracks manifest.

2.3. Модель данных (ключевые структуры)

Snake:

- id, userId|npcId, snakeTypeId, pos, vel, heading, speed.
- effects[]: ActiveEffect { type, endsAtTick, magnitude, charges? }
- race: { lap, checkpointIndex, finished, finishTimeMs }

Pickups:

- id, type, pos, active, respawnAtTick.

Track:

- tilemapKey, layers: road/offroad/walls, objects: start/finish/checkpoints/spawners.
- surfaceAt(x,y) возвращает road/offroad/outside.

2.4. Физика “ползучести” (простая и стабильная)

- Движение: speed интегрируется с accel/drag, поворот ограничен turnRate.
- Offroad: множитель к maxSpeed и accel, плюс VFX “пыль/грязь”.
- Столкновение со стеной: отражение/откат + штраф скорости, щит может поглотить 1 удар.

2.5. 4 типа змей (баланс как данные)

snakes.json:

- speedster: maxSpeed высокий, turnRate ниже.
- handler: turnRate высокий, maxSpeed ниже.
- bully: mass высокий, хуже поворот/разгон.
- trickster: минимальный offroadPenalty.

Требование: никакой “if snake==X then ...” в логике; только чтение статов из конфига.

2.6. 4 пикапа (система эффектов)

perks.json:

- BOOST: +скорость временно.
- SHIELD: 1 заряд, игнорирует удар/бомбу.
- OIL: снижает управляемость/увеличивает скольжение (дебаф).
- BOMB: AoE замедление/отброс.

Онлайн: подбор/взрыв/эффект решает сервер.

2.7. Онлайн (Colyseus): state sync и тики

Colyseus state model: клиент подписывается на изменения состояния комнаты и реагирует на патчи.

<https://docs.colyseus.io/state/>^[7] Рекомендуемый подход:

- Сервер тик 20–30 Hz (экономия), клиент рендер 60 Hz, интерполяция других змей.
- Клиент предсказывает свою змею (опционально), но сервер authoritative.

2.8. Лидерборд (Nakama): authoritative

Nakama: authoritative leaderboard означает, что **клиенты не могут сабмитить**; “all submissions must be made via the server runtime”, флаг authoritative задаётся при создании и потом не меняется. <https://heroiclabs.com/docs/nakama/concepts/leaderboards/>^[6] Политика:

- В оффлайне клиент отправляет “replay hash + время” на сервер, сервер проверяет минимально (или принимает с пометкой “unverified” — если надо).
- В онлайн match-server сабмитит финальное время напрямую через runtime.

2.9. Производительность (60 FPS) и стабильность

- Object pooling для хвоста/частиц/пикапов.
- Atlas для спрайтов.
- Ограничить количество сегментов хвоста в рендере (LOD: далеко — меньше сегментов).
- Перф-HUD (dev): FPS, frame time, количество активных объектов.

3) ТЗ + план выполнения для разработчиков (с DoR/DoD и задачами)

3.1. Definition of Ready (DoR)

Задача “готова к взятию”, если:

- Есть макет/референс поведения.
- Есть точные входные/выходные данные (тип/формат).
- Известны критерии приёмки (tests/скрин/видео).
- Нет блокировок по ассетам (есть заглушки).

3.2. Definition of Done (DoD)

Definition of Done — это список проверяемых критериев, при которых работа считается завершённой и готовой к релизу/интеграции. <https://lucid.co/blog/definition-of-done-agile>^[2] Для проекта DoD на фичу:

- Код в нужном модуле, без нарушения слоёв (sim vs render vs ui).
- Тестовый прогон: 10 минут гонки без краша.
- Нет регрессий FPS (dev сцена).
- Логи/метрики (если нужно).
- Короткое видео/гиф “как работает”.

3.3. ТЗ по подсистемам (конкретные требования)

A. Трассы (3 шт.)

- Формат: Tiled JSON, обязательные слои Road/Offroad/Walls/Objects.
- Objects: start, finish, checkpoint_1..n, pickup_spawn_1..m.
- Offroad: speed cap + accel penalty, VFX.
- Outside: reset на последний checkpoint + time penalty.

B. Гонка

- Фазы: Lobby → Countdown(3...2...1) → Running → Finished.
- Таймер: ms, отображение до тысячных.
- Финиш: фиксируем время каждого участника; сортировка результатов.

C. Змеи (4 типа)

- Экран выбора.
- Статы из snakes.json.
- Визуально: отличающиеся цвета/головы (заглушки на MVP).
- “Линька” (если включили): ability, cooldown.

D. Пикапы (4 типа)

- Спавн по точкам, respawn.
- Сервер authoritative (в online).
- UI: иконка+таймер.

E. NPC (3)

- Waypoint follow.
- 3 профиля поведения (аккуратный/норм/агрессивный).

F. Online (до 4)

- Создание/вход в комнату.
- Синхронизация позиций/эффектов/пикапов.
- Обработка отключений (leave → закончить гонку или заменить на NPC).

G. Лидерборды (Nakama)

- По каждой трассе: track_{id}_time.
- Сортировка ASC, operator best.
- Authoritative: клиент не сабмитит напрямую.⁴

3.4. План работ (таск-лист по эпикам)

Эпик E0: Инфраструктура

- E0.1 Monorepo, сборка client, линтер/форматтер.
- E0.2 Colyseus server шаблон комнаты на 4.
- E0.3 Nakama docker + создание leaderboards.

Эпик E1: Оффлайн вертикальный срез (MVP core)

- E1.1 Симуляция змеи (плавное управление, инерция).
- E1.2 Загрузка трассы #1 из Tiled, start/finish, checkpoints.
- E1.3 Offroad/Outside логика.
- E1.4 HUD: timer, speedometer, standings.
- E1.5 NPC 3 шт.

Эпик E2: Контент и разнообразие

- E2.1 4 типа змей (конфиг + UI выбора).
- E2.2 4 пикапа (BOOST/SHIELD/OIL/BOMB).
- E2.3 VFX/SFX hooks (события).

Эпик E3: Онлайн

- E3.1 Server tick loop + authoritative state.
- E3.2 Client net layer (join/leave, interpolation).
- E3.3 Server финализация результата.
- E3.4 Сабмит результата в Nakama authoritative leaderboard.[4](#)

Эпик E4: 3 трассы + полировка + презентация

- E4.1 Трасса #2 и #3 (уникальные), баланс.
- E4.2 Перф-оптимизация до стабильных 60 FPS.[2](#)
- E4.3 Демозапись геймплея + ответы комиссии.[2](#)

3.5. Критерии приёмки (готовые “чек-листы”)

Для релиз-кандидата:

- 3 трассы проходимы, start/finish корректны.
- Offroad всегда замедляет, outside всегда ресетит.
- В offline 1+3 NPC заезд без багов.
- В online 2–4 игрока доезжают и видят одинаковые результаты.
- Лидерборд обновляется только сервером (authoritative).[4](#)
- FPS стабильно ≥ 60 на тестовом ПК.[2](#)

**

1. <https://www.wrike.com/project-management-guide/faq/what-is-definition-of-done-agile/> ↩
2. <https://activecollab.com/blog/project-management/moscow-method> ↩
3. <https://airfocus.com/glossary/what-is-moscow-prioritization/> ↩
4. <https://heroiclabs.com/docs/nakama/concepts/leaderboards/best-practices/> ↩
5. <https://www.atlassian.com/agile/project-management/definition-of-done> ↩
6. image.jpg↩↩↩↩↩↩↩↩
7. <https://lucid.co/blog/definition-of-done-agile> ↩
8. <https://community.atlassian.com/forums/App-Central-articles/Understanding-the-MoSCoW-prioritization-How-to-implement-it-into/ba-p/2463999> ↩
9. <https://heroiclabs.com/docs/nakama/concepts/leaderboards/> ↩ ↩ ↩ ↩
10. <https://www.productplan.com/glossary/moscow-prioritization/> ↩