План демонстрации программного продукта

1. Введение (30 секунд)

Цель демонстрации:
«Сейчас я покажу, как работает программный прототип, реализующий три алгоритма процедурной генерации уровней: BSP, Drunkard Walk и клеточные автоматы. Вы увидите, как архитектура системы обеспечивает гибкость, а алгоритмы генерируют уровни под разные игровые сценарии»

2. Обзор архитектуры (1 минута)

Диаграмма классов:
«Система построена на модульной архитектуре.

1. **LevelGenerator** — центральный класс, который управляет алгоритмами через интерфейс ILevelGeneratorAlgorithm.
2. **LevelData** — универсальная структура для хранения данных уровня (стены, полы).
3. **Аналитический модуль** — независимо собирает метрики (время генерации, связанность).
4. **Визуализатор** — отрисовывает уровень в Unity, используя Tilemap System».

Sequence-диаграмма:
«Пользователь через интерфейс задаёт параметры → LevelGenerator запускает выбранный алгоритм → Результаты передаются в LevelData → Данные визуализируются и анализируются».

3. Демонстрация интерфейса (2 минуты)

Настройка параметров:
«Вот панель управления. Для каждого алгоритма можно задать уникальные параметры:

1. **BSP**: минимальный размер комнаты, число итераций, толщина стен.
2. **Drunkard Walk**: длина пути (walkLength).
3. **Клеточные автоматы**: вероятность стен, число итераций, правила рождения/смерти.

Например, увеличив walkLength для Drunkard Walk, мы получим более плотные уровни».

Запуск генерации:
«Нажимаем кнопку “Сгенерировать” → Алгоритм выполняется → Уровень

отображается в окне Unity. Время генерации выводится в реальном времени». Одновременно с этим в консоль выводится статистика генерации, по заданным нами метрикам.

Сохранение/загрузка:
«Все параметры можно сохранить в JSON-файл и загрузить для повторного использования».

4. Показ работы алгоритмов (3 минуты)

BSP (Binary Space Partitioning):
«Зададим параметры: minRoomSize = 5, maxIterations = 5.

Алгоритм создаёт иерархию комнат, соединяя их коридорами.

Результат: структурно упорядоченный уровень (Рис. 3.1).

Особенность: гарантированная связность, но геометрическая жёсткость».

Drunkard Walk:
«Параметр: walkLength = 3000.

Агент “протаптывает” путь, создавая органичные лабиринты.

Результат: уровень с ветвистыми путями (Рис. 4.2.1).

Риск: фрагментация при малом walkLength».

Клеточные автоматы:
«Параметры: initialWallChance = 0.45, simulationSteps = 5.

После 5 итераций формируются пещероподобные структуры (Рис. 4.3.1).

Можно менять правила: например, повысив birthLimit, получим более “агрессивные” стены».

5. Сравнительный анализ (2 минуты)

Метрики на экране:
«После генерации модуль анализа выводит:

Время выполнения (BSP: 45 мс, Drunkard Walk: 12 мс, CA: 80 мс).

Связанность (BSP: 100%, Drunkard Walk: 92%, CA: 88%).

Доля тупиков (BSP: 5%, Drunkard Walk: 18%, CA: 25%)».

Тепловая карта (условно):
«На примере Drunkard Walk: красным выделены тупики, зелёным — основные пути. Видно, как алгоритм создаёт “ветви”».

График зависимости времени от размера уровня:
«BSP масштабируется хуже из-за рекурсии, Drunkard Walk — линейно, CA — экспоненциально».

6. Интерактивная демонстрация (1 минута)

Смена параметров в рамках одного алгоритма, лучше всего продемонстрировать на CA меняя правила игры, тк они непосредственно будут менять структуру уровня.

7. Итоги (30 секунд)

Сильные стороны каждого алгоритма:
«BSP — контроль и структурность, Drunkard Walk — скорость и органичность, CA — естественность ландшафтов».

Рекомендации для разработчиков:
«Комбинируйте BSP и Drunkard Walk: первым создавайте комнаты, вторым — коридоры. Для пещер используйте CA с birthLimit = 4».

Практическая ценность:
«Прототип готов к интеграции в игровые движки. Его можно расширить, добавив новые алгоритмы через интерфейс ILevelGeneratorAlgorithm».