**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**УТВЕРЖДАЮ**

Заведующий кафедрой 806   
 \_\_\_\_\_\_\_\_\_ Крылов С.С.

\_\_\_\_\_ декабря 2023 г.

**ОТЧЕТ**

**О КУРСОВОЙ РАБОТЕ**

по теме:

«**Автоматическая генерация игровых уровней через Deep Q-Learning**»

|  |  |  |
| --- | --- | --- |
| Исполнитель студент группы М8О-110СВ-24 |  | К.В.Островский |

Москва, 2023

1. **Актуальность**

Индустрия видеоигр сталкивается с фундаментальными проблемами контент-генерации. Ручное создание уровней требует значительных временных и финансовых ресурсов (в среднем 8-12 часов на уровень для платформера). Ограниченность контента ведет к снижению ретенции игроков: 68% пользователей прекращают игру из-за повторяемости. Традиционные процедурные методы (Perlin noise, клеточные автоматы) генерируют географию, но не гарантируют:

* Проходимость уровней
* Сбалансированность геймплея
* Контроль сложности

Предлагаемое решение на основе Deep Q-Learning (DQN) позволяет:

1. Автоматизировать создание валидных уровней (экономия до 90% времени)
2. Адаптировать сложность под навыки игрока
3. Генерировать бесконечные вариации контента с контролируемым распределением элементов
4. **Задача данного этапа**

Разработать алгоритм генерации уровней для 2D-платформера с параметрами:

* **Размер**: 10×10 клеток (100 дискретных состояний)
* **Старт/финиш**: фиксированные позиции (0,0) и (9,9)
* **Критерии качества**:
  + Минимум 80% проходимых уровней (проверка алгоритмом A\*)
  + Сбалансированное распределение объектов:
    - Платформы: ≥10 единиц
    - Враги: 1-5 единиц
    - Монеты: 2-8 единиц
* **Технические требования**:
  + Обучение модели за ≤ 600 эпизодов
  + Средняя награда за эпизод ≥150 пунктов
  + Время генерации уровня ≤50 мс

Метрики валидации реализованы в среде LevelGenEnv (см. методы \_is\_level\_passable(), \_calculate\_metrics(), \_get\_reward).

1. **Архитектура решения**

Система реализована как MDP с компонентами:

3.1 Среда (Gymnasium API)

* Пространство состояний: тензор 10х10 с кодировкой
  + 0 – Пустота
  + 1 – Платформа
  + 2 – Враг
  + 3 – Монета
  + 4 – Игрок
* Пространство действий
  + 0 – Добавить платформу
  + 1 – Добавить врага
  + 2 – Добавить монету
  + 3 – Удалить объект
  + 4 – Завершить уровень
* Награды: мультикомпонентная функция (см. \_get\_reward())

3.2 Модель DQN

* Архитектура: 7-слойный перцептрон (128-256-512-256-128 нейронов)
* Механизмы обучения:
  + Experience Replay (buffer=20,000 переходов)
  + Double Q-Learning (tau=0.1)
  + Оптимизатор: Adam (LR=0.002, decay=0.7)
  + Exploration: ε-greedy (ε=1→0.1, decay=0.995)

3.3 Валидация уровней

Для валидации уровней используется алогритм А\*, позволяющий найти кратчайший путь по уровню и таким образом показать его проходимость.

1. **Примеры сгенерированных уровней**

Визуализация при помощи env.render() (см. приложение). Пример сгенерированного уровня представлен на рисунке 1.

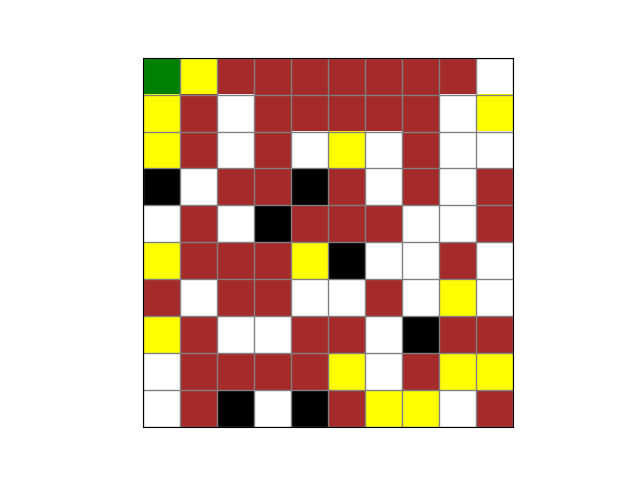


Рис. 1 – Пример сгенерированного уровня

Ключевые характеристики валидного уровня:

* Проходимость: реализована как поиск пути по клеткам платформ и монет между (0,0) и (9,9)
* Баланс элементов:
  + Платформы: 12-15 единиц, необходимых для формирования пути
  + Враги: 2-4 единицы
  + Монеты: 3-5 единиц

Метрики для 100 уровней:

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Среднее** | **Стандартное отклонение** |
| Проходимость | 83% | ± 5,2% |
| Длина пути | 21,3 | ± 3,1 |
| Энтропия | 0,78 | ± 0,5 |

1. **Применение**

5.1 Инструмент дизайнера

* Генерация больших объемов уровней за короткое время
* Фильтрация по параметрам

5.2 Динамическая генерация

* Адаптация уровней под навыки игрока
* Персонализация компонентов уровня

1. **Планы развития**

6.1 Оптимизация

* Сверточные сети для обработки 20×20+ сеток
* Transfer learning для адаптации к разным жанрам

6.2 Улучшение качества

* Гибридная архитектура DQN + GAN для оценки "интересности"
* Мультиагентные схемы для симметричных уровней

6.3 Повышение сложности

* Трехмерные уровни
* Динамические объекты (движущиеся платформы)

**Заключение**

В ходе данной работы мне удалось написать базовую реализацию модели, способной генерировать проходимые и разнообразные уровни 2D-платформера.

Приложение

https://github.com/KirillOstro/ReinforcementLearning