Міністерство освіти і науки України
Одеський національний технологічний університет
Вінницький національний технічний університет
Інститут комп`ютерної інженерії, автоматизації, робототехніки та програмування ім.П.Н.Платонова

МАТЕРІАЛИ

V ВСЕУКРАЇНСЬКОЇ НАУКОВО – ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ, АСПІРАНТІВ ТА СТУДЕНТІВ

«КОМП'ЮТЕРНІ ІГРИ І МУЛЬТИМЕДІА ЯК ІННОВАЦІЙНИЙ ПІДХІД ДО КОМУНІКАЦІЇ - 2025»

25-26 вересня 2025 р.
ОДЕСА

УДК 004.415

ПРОЦЕДУРНА ГЕНЕРАЦІЯ РІВНІВ ІЗ LLM-ПРОМПТАМИ: ШВИДКИЙ ПРОТОТИП У WEBGL/TYPESCRIPT

ПАНАСЮК Б.Ю. (boris.panasyuk@gmail.com)
БАБЮК Н.П. (babiuk@vntu.edu.ua)
Вінницький національний технічний університет

Представлено підхід до швидкого прототипування ігрових рівнів шляхом процедурної генерації з використанням великих мовних моделей (LLM). Розглянуто можливості LLM для створення параметризованих рівнів на основі текстових промптів та окреслено методи забезпечення детермінізму й контролю якості згенерованого контенту. Проаналізовано архітектурні обмеження середовища WebGL/TypeScript при інтеграції LLM, способи ефективного промптування моделей, валідацію якості рівнів та межі генеративних можливостей. У висновках сформульовано переваги запропонованого підходу та пов'язані ризики його застосування.

Постановка проблеми

Процедурна генерація ігрових рівнів є одним із перспективних напрямів в ігровій індустрії, що дозволяє автоматично створювати контент за допомогою алгоритмів. Традиційно такі алгоритми або суворо детерміністичні (наприклад, генерація на основі заданих правил), або навчаються на конкретних наборах даних гри. Великі мовні моделі (LLM) останніх поколінь відкрили нові можливості для процедурного генерування ігрових рівнів [1]. Однак інтеграція LLM у ігровий рушій чи середовище типу WebGL потребує врахування обмежень обчислювальних ресурсів, швидкості відгуку та якості генерованого контенту [2]. Постає проблема розробки методики використання LLM для генерації рівнів, яка б забезпечила керованість процесом та прийнятну якість результатів.

Мета дослідження

Метою дослідження є розробка методів та створення прототипу системи процедурної генерації рівнів, яка використовує LLM-промпти для створення ігрового контенту у середовищі WebGL/TypeScript. Це передбачає дослідження способів формування ефективних промптів, механізмів керування випадковістю та валідації згенерованих рівнів.

Приклад гіпотетичного прототипу

Розглянемо гіпотетичний сценарій використання розробленого прототипу. Наприклад, дизайнер гри бажає швидко отримати варіанти рівня для платформера у стилі *"містичний ліс"*. Він задає такі вимоги: розмір рівня – приблизно 50x10 умовних одиниць (горизонтальний *side-scroller* рівень), тема – нічний ліс із руїнами, рівень складності – середній, присутні 5 ворогів (літаючі та наземні), одна схована область з бонусом, фінальний бос наприкінці рівня. Дизайнер формулює промпт, що описує ці вимоги, але для моделі (яка краще працює з англійською) промпт автоматично перекладається англійською. Додатково, промпт містить інструкцію щодо формату: модель має повернути об'єкт JSON з полями: "terrain" (список сегментів платформи з їх координатами), "enemies" (список ворогів із типом та координатами), "secretArea" (координати прихованої зони), "boss" (позиція боса), тощо. Після відправки промпта LLM генерує відповідь. Приклад фрагменту згенерованого JSON (спрощено):

```
{
  "terrain": [ {"x":0,"y":0,"length":10}, {"x":12,"y":-1,"length":5}, … ],
  "enemies": [
     {"type":"bat","x":8,"y":5},
     {"type":"goblin","x":15,"y":0}, …
  ],
  "secretArea": { "entrance":{"x":22,"y":0}, "treasure":{"x":25,"y":3} },
  "boss": { "type":"treeSpirit", "x":48,"y":0 }
}
```

Отримані дані імпортуються у движок WebGL: створюється віртуальна сцена, де сегменти платформи (terrain) відображаються як земна поверхня й підвищення, вороги (enemies) – як відповідні спрайти з поведінковими скриптами, бос – як окремий великий персонаж із особливою областю. При завантаженні рівня рушій перевіряє, що всі координати коректні (не виходять за межі сцени), що секретна область досяжна (в даному випадку перевіряється наявність входу entrance і що він прилягає до якогось сегмента terrain), а також що вороги розподілені рівномірно, не скупчені надмірно в одній точці. Деякі аспекти, які модель не завжди враховує, автоматично поправляються рушієм. Наприклад, якщо модель розмістила ворога типу “гоблін” (y=0) на сегменті землі, але не врахувала, що над ним має бути достатньо простору для графіки персонажа, рушій може зсунути такого ворога трохи вище y=1 або на інший сегмент. Після цього рівень доступний для тестування – дизайнер може одразу грати й оцінити його. Припустимо, тестування показало, що рівень занадто легкий (вороги розташовані далеко один від одного). Дизайнер може відредагувати промпт (збільшити кількість ворогів чи змінити їх розподіл) або ж попросити модель: “Зроби рівень складнішим: додай більше ворогів у середній частині рівня та декілька шипів як пастки”. Модель, отримавши такий уточнений запит, згенерує новий JSON із додатковими ворогами та пастками (наприклад, "traps": [{"type":"spikes","x":30,"y":0}, …]). Таким чином, в інтерактивному циклі за лічені хвилини створюється і поліпшується ігровий рівень, що відповідає задуму дизайнера. Даний приклад ілюструє перевагу підходу: використовуючи LLM, можна отримати різноманітні варіанти рівнів, просто змінюючи описові параметри, без необхідності вручну моделювати геометрію чи розставляти кожен об’єкт.

Варто зазначити, що у даному прототипі ми свідомо спростили низку аспектів. Зокрема, модель не генерує безпосередньо код на TypeScript чи GLSL шейдери – вона лише описує рівень на високому рівні. Існують роботи, де LLM використовується для генерації ігрової логіки або коду (наприклад, генерації скриптів на TypeScript для керування об’єктами), проте це породжує додаткові складнощі перевірки коректності коду та безпеки. Натомість наш акцент – на генерації контенту (даних рівня), а не поведінкового коду. Іншим спрощенням є відсутність складної геометрії: рівень представлено дискретно (сітка або сегменти), що зручно для текстового опису. У разі 3D-рівнів із довільною геометрією промпт мав би бути набагато складнішим, або вимагався б інший підхід (наприклад, генерація на основі повідомлень-функцій типу “додати об’єкт на сцену” з послідовним викликом таких функцій моделлю). Натомість формат, обраний у прототипі, добре підходить для платформера чи лабіринту, де важливі лише розташування об’єктів на сітці.

Методи та реалізація

Архітектура рішення: Прототип складається з фронтенд-додатка на TypeScript, що працює в браузері (Canvas/WebGL), та хмарного API доступу до LLM (наприклад, OpenAI GPT-4). TypeScript використано для швидкої розробки і типобезпеки – вихідний код компілюється в JavaScript і виконується у браузері [3], що означає ті самі обмеження середовища, що й для звичайного JS. WebGL використовується для рендерингу 3D/2D-графіки у браузері; це JavaScript API для апаратно прискореної графіки, що дозволяє безпосередньо використовувати GPU для відтворення сцени на canvas. Обмеження WebGL включають відсутність сучасних можливостей, таких як багатопотоковість і обчислювальні шейдери (порівняно з новішим WebGPU) . Проте WebGL сумісний з усіма сучасними браузерами і має розвинену екосистему, що важливо для доступності прототипу. Використання LLM в такому клієнтському середовищі можливе лише через мережеві запити до API; безпосереднє розгортання моделі у браузері наразі обмежене через вимоги до ресурсів. Тому прототип надсилає промпти до віддаленого сервера LLM та отримує згенерований контент у вигляді структурованих даних (JSON).

Очікувані результати

Очікується, що прототип підтвердить життєздатність запропонованого підходу: LLM зможе генерувати різноманітні рівні на основі текстових описів майже миттєво. Передбачається отримати приклади рівнів, які відповідають заданим умовам і залишаються відтворюваними при тих самих промптах, демонструючи керованість процесу генерації. Аналіз результатів дозволить сформулювати рекомендації щодо ефективного промптування та методів контролю якості контенту.

Висновки

У роботі сформовано прототип системи процедурної генерації рівнів із використанням LLM-промптів у середовищі WebGL/TypeScript. Проведений аналіз дозволяє зробити такі висновки:

Переваги підходу:

1. Швидке прототипування та творчість. LLM здатна генерувати різноманітні дизайни рівнів за мінімального втручання людини. Зміна текстового опису призводить до якісно нових варіантів рівня, що прискорює творчий процес. Це особливо корисно на ранніх етапах розробки, коли потрібен брейнштормінг ідей рівнів.

2. Параметризація і гнучкість. На відміну від жорстких алгоритмів PCG, модель реагує на високорівневі параметри (тема, наратив, стиль). Можна легко накладати додаткові вимоги, навіть розмиті (“зроби рівень похмурішим”), і модель врахує їх у міру свого розуміння. Це відкриває новий рівень інструментарію для дизайнерів – описовий, семантичний.

3. Інтеграція різних аспектів контенту. LLM може одночасно генерувати ігровий світ та наративні елементи. Наприклад, у перспективі модель здатна створити опис рівня і зв’язану з ним історію або діалоги. Таке об’єднання традиційно потребувало окремого дизайну для рівня та сюжету, а генеративна модель може забезпечити цілісність “історії і простору”.

4. Масштабованість без додаткового навчання. Застосування готових LLM дозволяє генерувати контент для різних жанрів і стилів без тренування спеціалізованих моделей під кожну гру. Моделі володіють знаннями про безліч ігрових жанрів і можуть працювати в режимі zero-shot, створюючи осмислені рівні навіть для нових умов. Це скорочує витрати на розробку генераторів контенту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ed-douibi H., Cánovas J. L., Bordeleau F., Cabot J. WAPIml: Towards a Modeling Infrastructure for Web APIs. Proc. of the 22nd ACM/IEEE Int. Conf. on Model Driven Engineering Languages and Systems (MODELS 2019 Companion). 2019. P. 748–752.
2. Node.js. Офіційний веб-сайт URL: https://nodejs.org (дата звернення: 23.06.2025).
3. Using TypeScript with WebGL to render graphics on the web. URL: https://blog.logrocket.com/using-typescript-webgl-render-web-graphics/ (дата звернення: 23.06.2025).

УДК 004.92:004.932:658.5

МУЛЬТИМЕДІА ТА ШТУЧНИЙ ІНТЕЛЕКТ У ГЕЙМІФІКАЦІЇ ОСВІТНІХ І ВИРОБНИЧИХ ІНФОРМАЦІЙНИХ СИСТЕМ

ПАНОВИК У.П. (uliana.p.panovyk@lpnu.ua), ПАНОВИК Р.Р. (roman.r.panovyk@lpnu.ua)
ГІДЕЙ Р.В. (roman.v.hidei@lpnu.ua)
Національний університет «Львівська політехніка»

Розглянуто застосування мультимедіа та штучного інтелекту в гейміфікації освітніх (АСУ НМК/LMS) і виробничих (SCADA/HMI) систем. Запропоновано модель «користувач – мультимедіа – AI – інформаційна система» та алгоритм адаптації, що поєднує регулювання складності, процедурну генерацію і персоналізовані рекомендації. В освіті це підвищує мотивацію і знижує когнітивне навантаження, у виробництві – скорочує час реакції та помилки. Використання уніфікованих метрик підтверджує ефективність такого підходу.

Постановка проблеми. Мультимедіа стало важливим інструментом комунікації, навчання та управління процесами. В освіті воно забезпечує інтерактивність і мотивацію, особливо в поєднанні з гейміфікацією, а штучний інтелект дає змогу персоналізувати траєкторії та автоматично адаптувати завдання. У виробництві мультимедіа виконує роль інтерфейсу між оператором і технологічним середовищем, а алгоритми AI аналізують дані й подають результати у