НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматизованих систем обробки інформації та управління

**КУРСОВА РОБОТА**

з програмування структурованих даних

на тему: “Розробка багатокористувацької гри”

Студента I курсу ФІОТ групи ІП-71

напряму підготовки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

спеціальності “Інжеренія програмного забезпечення”

Амброса В. В.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ - 2018 рік

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 21 сторінка, 14 рисунків, 0 таблиць, 8 посилань.

Об'єкт дослідження: багатокористувацькі браузерні двовимірні ігри побудовані на концепції архітектури "клієнт-сервер".

Мета роботи: розробка комплексу програм для багатокористувацької гри у мережі Інтернет у власно спроектований 2D шутер “Havio”.

Виконана програмна реалізація серверної та клієнтської частини багатокористувацької гри “Havio. Покращено сумісність з більшістю браузерів, в тому числі й мобільними.

ЗМІСТ

[ВСТУП 4](#_Toc515583490)

[1. ПОСТАНОВКА ЗАДАЧІ 5](#_Toc515583491)

[2. ТЕОРЕТИЧНІ ВІДОМОСТІ 6](#_Toc515583492)

[3. ОПИС АЛГОРИТМІВ 8](#_Toc515583493)

[3.1. Алгоритми виявлення зіткнень. 8](#_Toc515583494)

[3.2. Алгоритм штучного інтелекту. 10](#_Toc515583495)

[3.3 Алгоритм розподілу гравців по кімнатах 12](#_Toc515583496)

[4. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 13](#_Toc515583497)

[4.1 Функціональна структура програмного забезпечення. 13](#_Toc515583498)

[4.2 Опис функцій частин програмного забезпечення. 13](#_Toc515583499)

[5. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15](#_Toc515583500)

[6. ІНСТРУКЦІЯ КОРИСТУВАЧА 16](#_Toc515583501)

[ВИСНОВКИ 17](#_Toc515583502)

[ПЕРЕЛІК ПОСИЛАНЬ 18](#_Toc515583503)

# ВСТУП

Відеогра — гра з використанням послідовності зображень на екрані монітора та вводу користувача (клавіатура, миша, джойстик тощо). Першою офіційно зареєстрованою грою вважається гра з використанням екрану телевізора та растрових зображень — спрайтів (раніше був розроблений “Ракетний симулятор”, але вона не вважається грою, бо не використовує растрового зображення).

Історично відеоігри з'явились раніше, ніж персональний комп'ютер. Це зумовлено тим, що в 1960-х роках комп'ютерні технології не були масовими та не розглядались як серйозний інструмент в соціальній галузі. Ігри були створені на електронній базі та зіграли важливу роль в поширенні комп'ютерної техніки. В 1970-х відеоігри стали однією з найпопулярніших розваг. Ігрові автомати встановлювалися в барах, кафе та інших закладах.

Перша в сучасному розумінні відеогра була створена Баєром Ральфом.[2] Баєр розробив ігрову консоль, яка передавала зображення на екран телевізора.

Браузерні ігри — вид відеоігор, де в якості користувацького інтерфейсу використовується браузер або плагін для браузера (наприклад, Adobe flash).[3] Браузерні ігри поділяються на однокористувацькі, багатокористувацькі та масово-багатокористувацькі. Більшість браузерних ігор орієнтовані на модель Free-to-play, тобто розповсюджуються безкоштовно. В даній курсовій роботі розробляється проект браузерної масово-багатокористувацької гри.

# 1. ПОСТАНОВКА ЗАДАЧІ

Задача полягає у тому, щоб розробити клієнтську та серверну частину гри а також алгоритм штучного інтелекту для ботів, що емулюють поведінку справжнього гравця та керуються сервером.

Основними завданнями до реалізації є:

* динамічність (постійний рух і загроза програти);
* стратегічність (правильно розрахувати доступні “кошти”, для покращення юніта);
* швидкодія і малий інтернет-трафік, що потребує гра;

Правила гри наступні: гравець, керуючи “точкою” (далі — юніт) в центрі екрана за допомогою курсора миші переміщує її по ігровому полі, на якому розташовані перешкоди та різні “бонуси”; випускає “кулі” у напрямку свого руху, пошкоджуючи та знищуючи інших таких же гравців; отримує за це деяку кількість очок, за які потім можна змінювати характеристики свого юніта — “прокачувати” його. Переможцем вважається той, хто набере більшу кількість “вбивств” та меншу кількість “смертей”. Після “смерті” (коли закінчуються одиниці життя), юніт відроджується, але вже з втраченими покращеннями.

Алгоритм для бота являє собою скінченний автомат та має три стани:

* випадкове блукання;
* обминання перешкод;
* переслідування цілі.

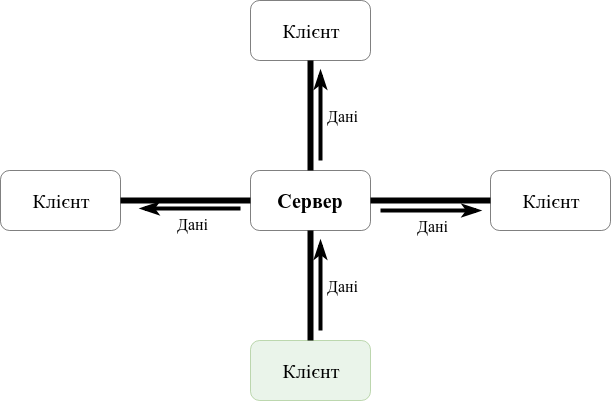
Обов'язково реалізується алгоритм, що розподіляє гравців по різних ігрових кімнатах, щоб попередити їх переповнення та ситуацію, що унеможливлює ігровий процес.

Також основним алгоритмом у грі є виявлення зіткнень гравців з перешкодами “кулями” опонентів.

# 2. ТЕОРЕТИЧНІ ВІДОМОСТІ

Система являє собою сервер та набір клієнтів. Сервер — комп'ютер зі встановленою операційною системою та програмним забезпеченням (див. “Вимоги до програмного забезпечення”). Необхідно, щоб сервер мав статичну IP-адресу в мережі.

Клієнт — користувач (гравець), також комп'ютер з встановленою операційною системою, браузером та доступом в мережу Інтернет. Для того, щоб гра виконувала свою функцію, необхідний зв'язок між клієнтами. Робити пряме підключення від кожного клієнта до інших недоцільно (кількість активних з'єднань: ). Для цього, кожен клієнт приєднується до посередника — сервера, а сервер, в свою чергу, транслює дані від клієнта іншим клієнтам. Тоді кількість з'єднань скорочується до *n*. Схема роботи такої системи зображення на рис. 1.

Рис. 1. Схема роботи системи “клієнт-сервер”

Для забезпечення такого з'єднання використовується протокол websocket. Для відкриття з'єднання по протоколу websocket необхідно відправити на сервер GET-запит із заголовком 101. Сервер сприйме це як запит на відкриття з'єднання WebSocket, і далі, всі дані передаватимуться саме через цей протокол.

Серверне програмне забезпечення реалізується на оточенні Node.js. Такий вибір обґрунтований тим, що Node.js використовує подієву модель (англ. — event-based), яка зручна тим, що всі запити обробляються асинхронно, тобто запит одного користувача не блокує виконання програми.[1]

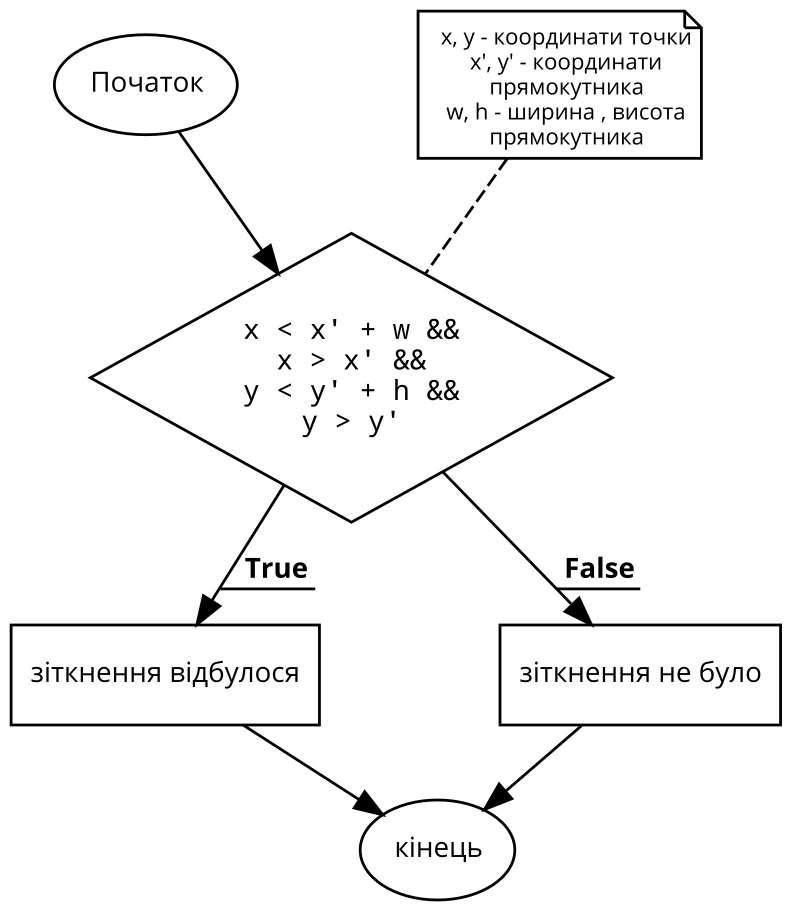
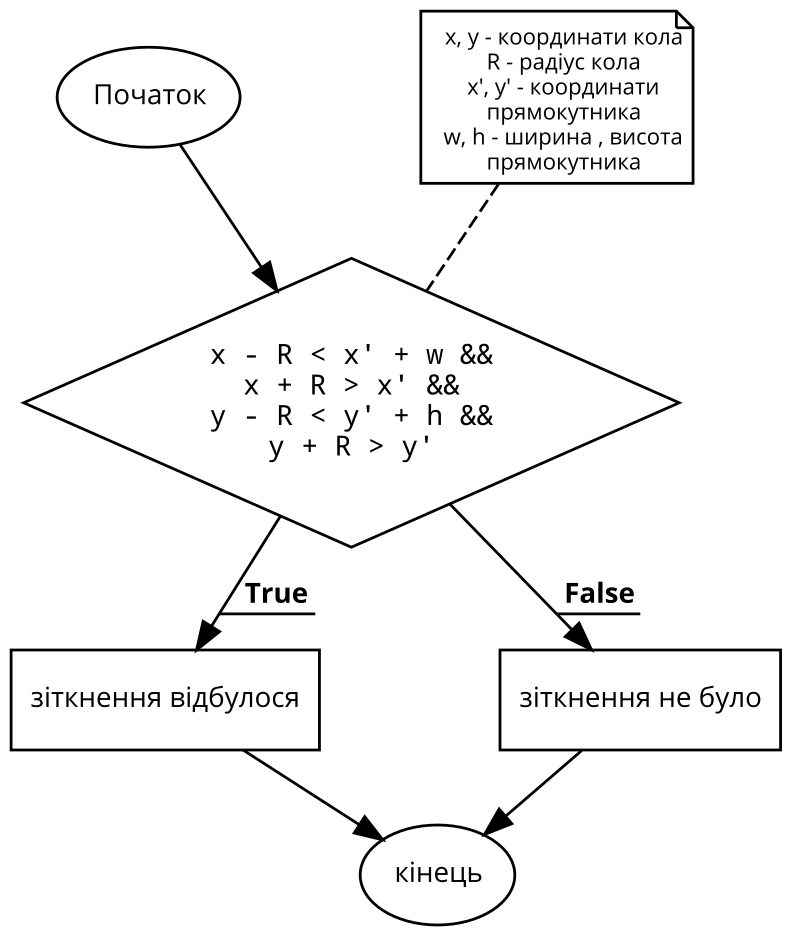
Клієнтська частина являє собою html-сторінку, яка рендериться в браузері користувача та додаткові файли, а саме:

* JavaScript код (.js);
* каскадні таблиці стилів (.css);
* спрайти (.png);
* аудіо (.wav)
* шрифти (.woff2)

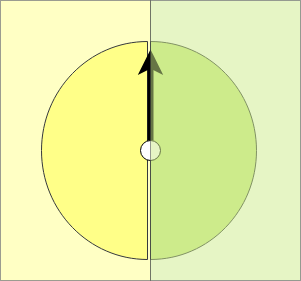
Протокол websocket забезпечує двосторонню передачу даних, тобто і клієнт і сервер можуть обмінюватися даними (на відміну від http, де запит на відправляється лише клієнтом).

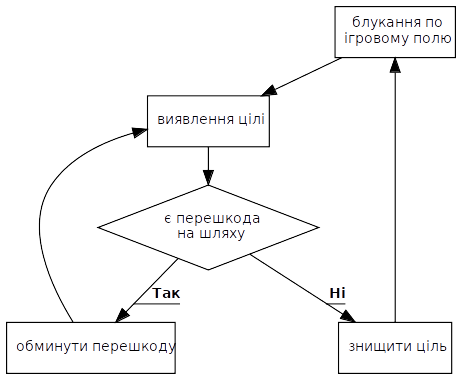
# 3. ОПИС АЛГОРИТМІВ

### 3.1. Алгоритми виявлення зіткнень.

* 1. У грі використовуються два базові алгоритми для виявлення зіткнень: зіткнення точки з прямокутником (точка знаходиться всередині прямокутника) та зіткнення круга з прямокутником. На рис. 2 та рис. 3 представленні блок. схеми цих алгоритмів.
  2. 
  3. Рис. 2. Алгоритм виявлення зіткнення точки з прямокутником
  4. 
  5. Рис. 3. Алгоритм виявлення зіткнення круга з прямокутником

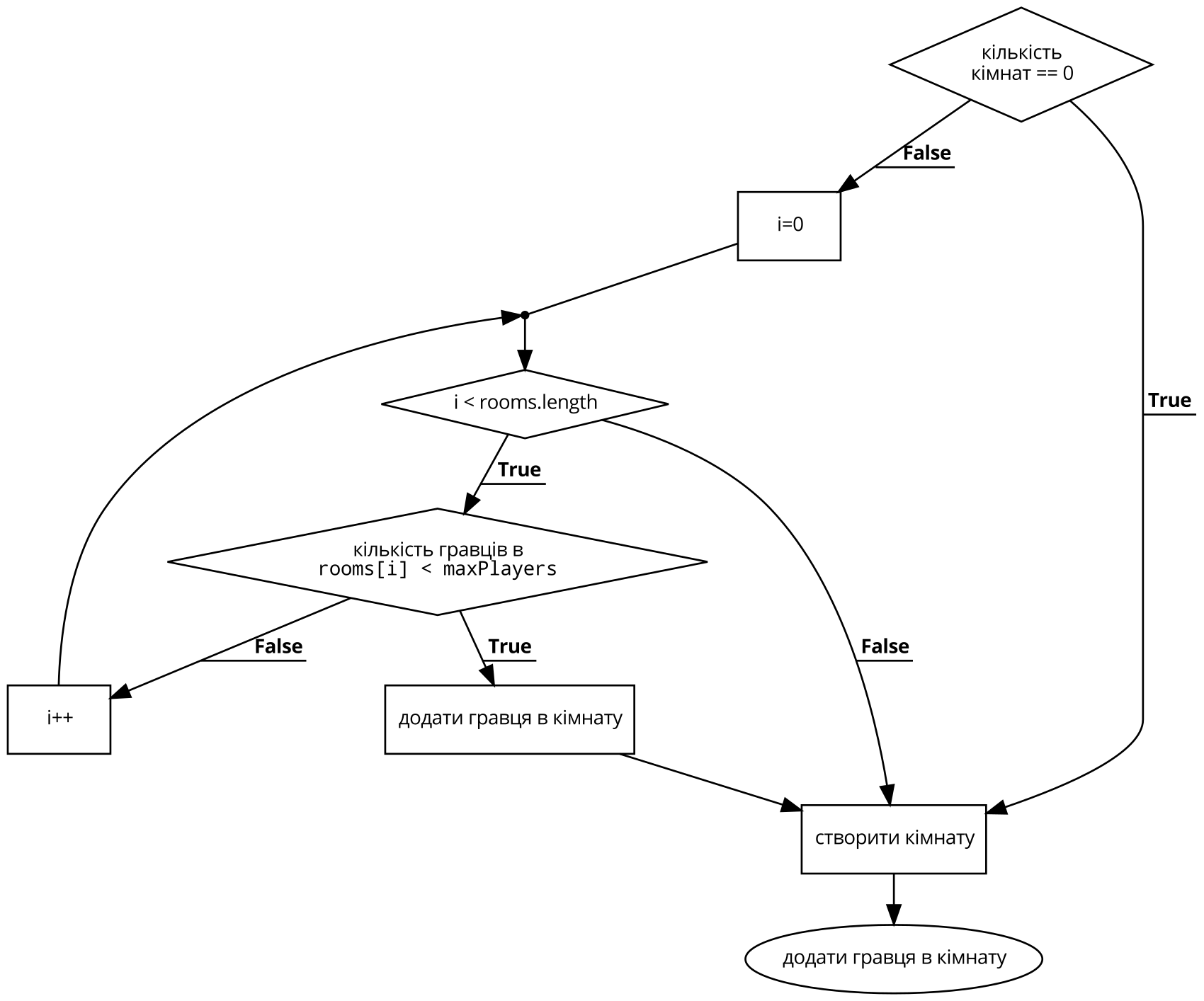
### 3.2. Алгоритм штучного інтелекту.

* 1. Алгоритм поведінки ботів (гравці, що керуються сервером) є скінченним детермінованим автоматом та має три стани: блукання, обминання перешкод, та переслідування цілі. На рис. 4 схематично позначено “очі” бота в режимі обминання перешкод. Якщо перешкода перетинається з лівим півколом (жовтий колір), він починає обертатися вправо, а якщо з правим (зелений) — вліво. Якщо інший гравець або бот наближається на деяку відстань, бот переходить з режиму блукання в режим переслідування цілі. Якщо під час переслідування на шляху трапляється перешкода, то доки вона не буде обминута, бот знаходиться в стані обходу перешкод. Далі бот повертається в режим переслідування. Після знищення цілі бот переходить в режим блукання. На рисунку 5 зображено блок-схему алгоритму поведінки бота.
  2. Рис. 4. Схематичне зображення зон, які “сканує” бот.

Рис. 5. Алгоритм поведінки бота.

### 3.3 Алгоритм розподілу гравців по кімнатах

Для забезпечення рівномірної кількості гравців у кожній “кімнаті”, алгоритм визначає, в яку кімнату реєструвати кожного гравця. Таким чином, в кімнаті не може бути більше, ніж 10 гравців (maxPlayers). На рис. 6 зображено блок-схему алгоритму.

Рис. 6. Алгоритм додавання гравця в кімнату.

# 4. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Функціональна структура програмного забезпечення.

Код програми є моделлю об'єктно-орієнтовного програмування. На рисунку 7 наведено діаграму класів, що використовуються в програмі.

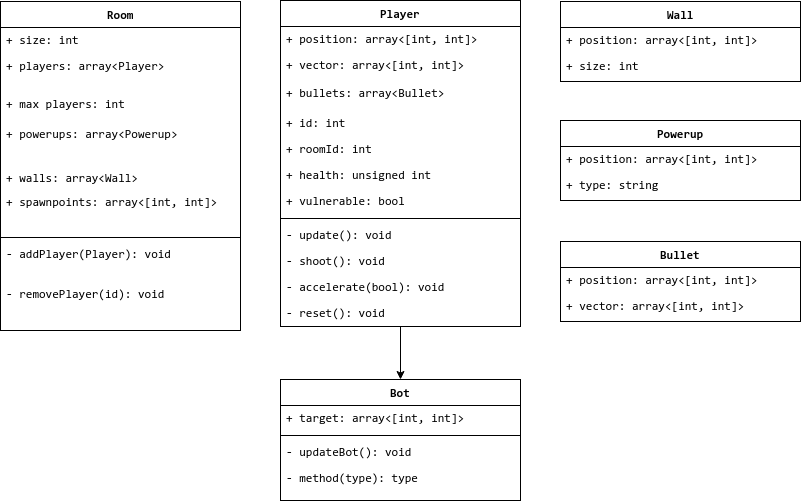


Рис. 7. Діаграма класів.

Код гри містить модуль classes.js, де описуються всі класи, що використовуються у грі.

### 4.2 Опис функцій частин програмного забезпечення.

Обов’язковою частиною програми є функція render(), що вимальовює гравців та інші об'єкти згідно даних, що приходять з сервера. Функція render() викликається, як callback всередині requestAnimationFrame()[4] (вбудована в браузер функція).

"Прокачка" юніта — зміна параметрів гравця, таких, як швидкість, кількість куль, що випускаються за раз, максимальне здоров'я та ін.

Загальна блок-схема роботи програми зображена на рис. 8.

# Рис. 8. Блок-схема роботи програми.

# 5. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програму було протестовано на критерій швидкості завантаження сторінки. Було вибрано декільна типів з'єднань (normal, fast 3G, slow 3G). Скріншоти графіків завантаження (waterfall) наведені на рисунках 9, 10, 11.

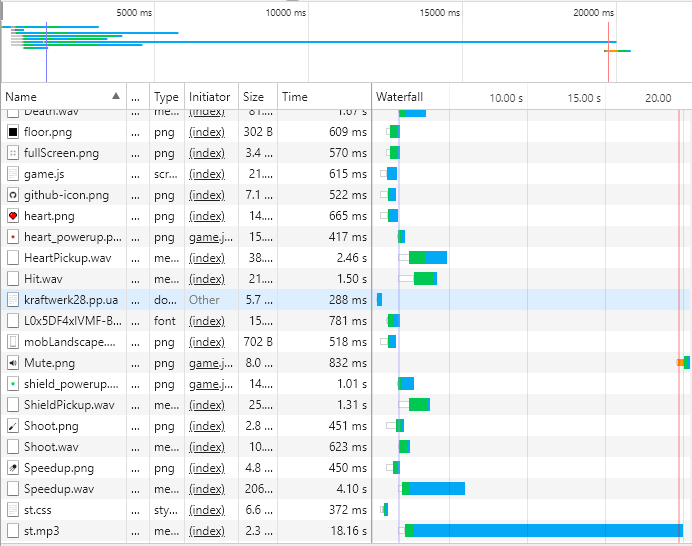


Рис. 9. Час повного завантаження сторінки при відносно швидкому з'єднанні.

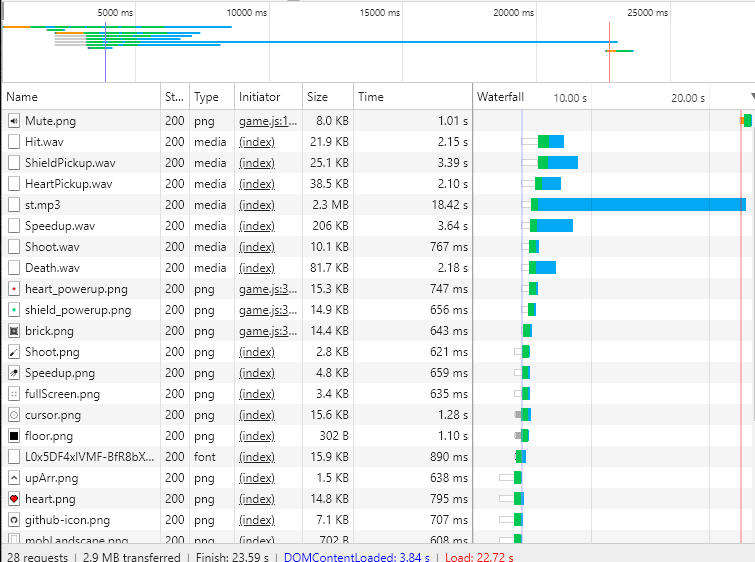


Рис. 10. Час повного завантаження сторінки при з'єднанні 3G.

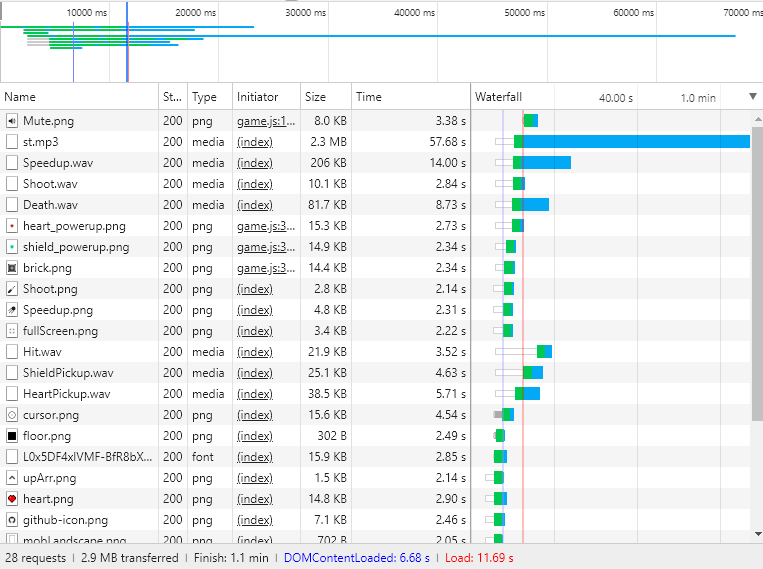


Рис. 11. Час повного завантаження сторінки при повільному з'єднанні.

# 6. ІНСТРУКЦІЯ КОРИСТУВАЧА

Для комфортної гри рекомендується інтернет-з'єднання з швидкістю не меншою за 1 Мб/с.

Для початку гри необхідно відкрити браузер (див. Вимоги до програмне забезпечення) та перейти за посиланням <http://kraftwerk28.pp.ua>.

Далі слід ввести свій ігровий нік та натиснути Play (рис. 12).

Керування здійснюється мишею, прискорення — правою кнопкою миші, а постріл — лівою копкою миші. Для покращення юніта слід вибрати меню покращення (рис. 14) та вибрати необхідний пункт. Положення гравців на мапі зображено в правому нижньому кутку (міні-карта рис. 13).



Рис. 12. Початковий екран гри.

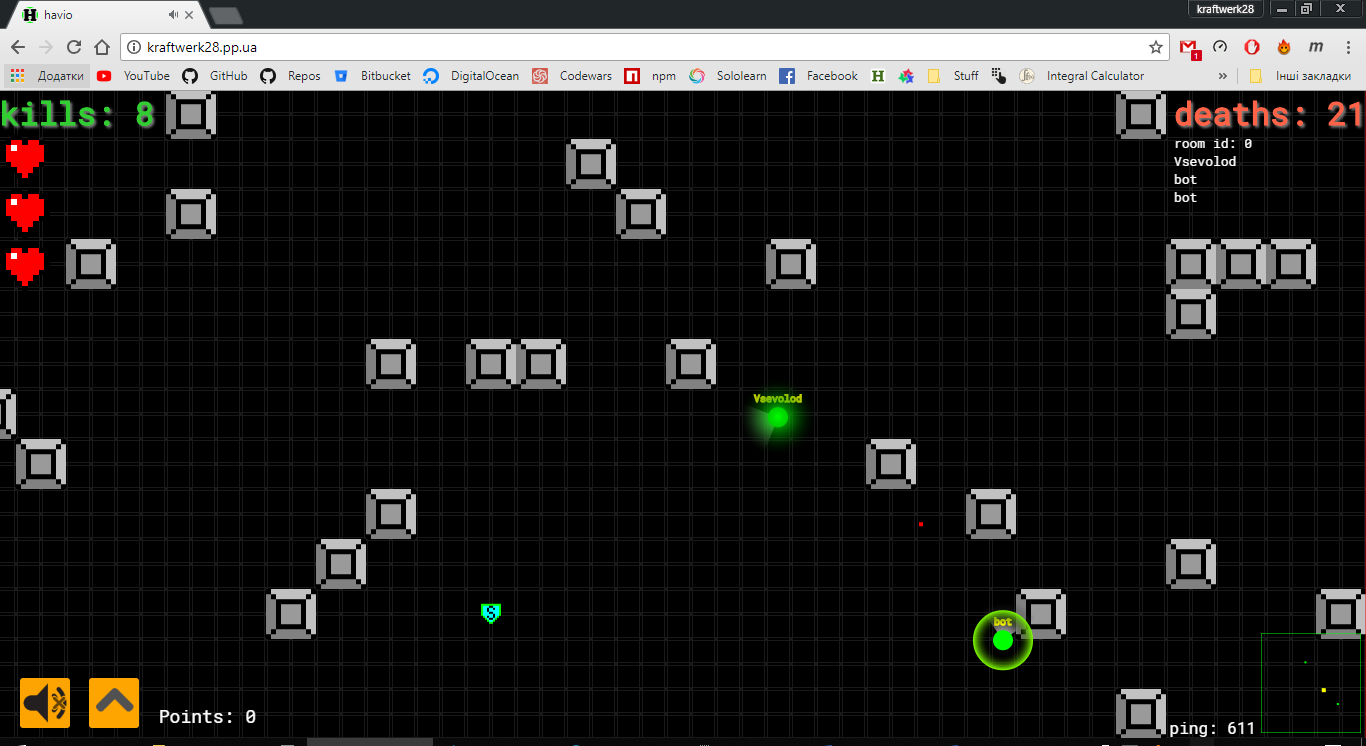


Рис. 13 Користувацький інтерфейс

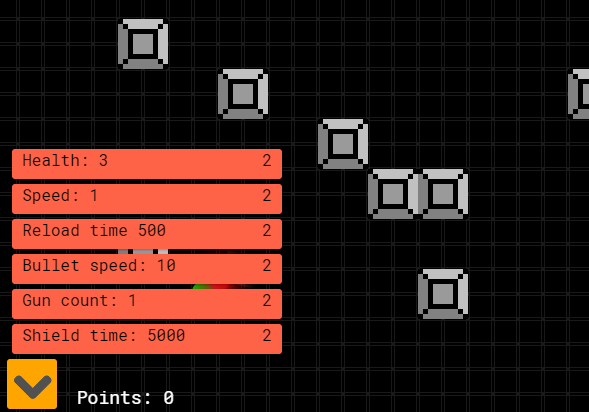


Рис. 14. Меню "Прокачки"

# ВИСНОВКИ

У роботі було використано алгоритми виявлення зіткнень напростіших геометричних фігур (точка, прямокутник, коло) для реалізації взаємодії гравців у багатокористувацькій грі "Havio". За допомогою html-розмітки побудовано "дружній" користувацький інтерфейс.

Арендовано сервер DigitalOcean[7] та домен <http://kraftwerk28.pp.ua>, куди завантажено кінцевий реліз гри.

Весь сирцевий код вільний і розповсюджується за ліцензією МІТ і знаходиться в репозиторії GitHub[6].

Для можливості пошуку гри на пошуковій сторінці Google, сайт проіндексовано в сервісі Google Webmaster[8].

# ПЕРЕЛІК ПОСИЛАНЬ

1. <https://nodejs.org/uk/>
2. <https://en.wikipedia.org/wiki/Video_game>
3. <https://en.wikipedia.org/wiki/Browser_game>
4. <https://developer.mozilla.org/ru/docs/DOM/window.requestAnimationFrame>
5. <https://stackoverflow.com/>
6. <https://github.com/kraftwerk28/hav-io>
7. <https://cloud.digitalocean.com>
8. https://www.google.com/webmasters/