Міністерство освіти і науки України

Чернівецький національний університету імені Юрія Федьковича»

Відокремлений структурний підрозділ «Фаховий коледж Чернівецького національного університету імені Юрія Федьковича»

Природниче відділення

Циклова комісія комп'ютерної інженерії

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»

зі спеціальності *123 Комп'ютерна інженерія*

підготовки за освітньо-професійною програмою «*Комп'ютерна інженерія*»

на тему: «**Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ»**

**Виконав:**

студент 4-го курсу *Деркач Максим Андрійович*

(прізвище, ім'я та по батькові) (підпис)

**Керівник:**

Мельничук А.Ю. _____

(науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

**Рецензент:**

к. ф.-м. н., Тащук О.Ю. _____

(науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

**До захисту допущено:**

**Протокол засідання циклової комісії № _**

від „___"_____ 2024 р.

Голова циклової комісії _____ Олександр ТАЩУК

Чернівці, 2024

Завдання та календарний план роботи

Міністерство освіти і науки України
Чернівецький національний університету імені Юрія Федьковича»
Відокремлений структурний підрозділ «Фаховий коледж
Чернівецького національного університету імені Юрія Федьковича»

Затверджую
Голова циклової комісії,
Олександр ТАЩУК.

_____

«____» _____2023 р.

**Завдання**

| | |
|---|---|
| На кваліфікаційну роботу | Деркач Максим Андрійович |
| Тема кваліфікаційної роботи | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ |
| Постановка завдання в короткій формі | Відтворення та доповнення класичної аркадної гри «Арканоїд» використоуючи мову програмування javascript |
| Вихідні дані | https://en.wikipedia.org/wiki/Arkanoid<br><br>https://en.wikipedia.org/wiki/JavaScript |

**Календарний план роботи**

| Дата отримання версії звіту керівником | Етап виконання роботи | Виконання (зазначає керівник) |
|---|---|---|
| 10.10.23 | Отримання завдання до кваліфікаційної роботи. | |
| 12.10.23 | Ознайомлення з оригінальною версією гри, нотації її основних механік та об’єктів | |
| 30.10.23 | Огляд існуючих аналогів. Оформлення розділу 1. | |
| 15.11.23 | Вибір мови програмування для створення кодової частини гри. Оформлення розділу 2. | |
| 15.12.23 | Створення текстур основних об’єктів та пошук і завантаження більшості аудіофайлів | |
| 07.02.24 | Написання основного коду програми. Оформлення розділу 3. | |

| | | |
|---|---|---|
| 31.04.24 | Фінальні доповнення та зміни, тестування програми. Оформлення розділу 4. | |
| 20.05.24 | Оформлення та редагування кваліфікаційної роботи | |
| *за графіком* | Подання роботи на перевірку Інтернет-сервісом Unicheck. | |
| *за графіком* | Подання роботи на рецензію | |
| *за графіком* | Представлення роботи на засіданні Циклової комісії | |
| *за графіком* | Захист кваліфікаційної роботи | |

Дата видачі завдання _____.

Студент Максим ДЕРКАЧ

Керівник (П.І.Б) Христина МЕЛЬНИЧУК

Андрій МЕЛЬНИЧУК

# АНОТАЦІЯ

В роботі «Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ» розглянуто розширене застосування мови javascript для динамічних проектів, проаналізовано основи та стандартні методи зазвичай використовувані для реалізації простих ігрових проектів, аналізовано умови та вимоги щодо програмування кодової частини проекту, досліджено різні способи компонування та покращення існуючих методів відповідальних за виконання ігрових подій та здійснено численні випробування для виправлення недоліків та покращення проекту.

Створення відео-ігор, крім можливості опублікування якісного продукту й отримання прибутку, зазвичай включає набуток досвіду в багатьох корисних сферах роботи з цифровими проектами.

Метою роботи є реалізація javascript коду та додатків до нього, формуючих разом функціональну версію класичної гри “Арканоїд”, з реалізацією статичних, динамічних об’єктів та взаємодії між ними, системою життів, бонусів, балів та інших компонентів гри.

Робота складається з 4 розділів, в яких зроблено аналіз вимог та умов проекту, реалізовано програмний код гри з динамічними об’єктами та їх взаємодіями, системами бонусів, життів, балів, перемоги та поразки, здійснені численні випробовування проекту.

Кваліфікаційна робота містить 68 сторінок, 9 рисунків та 18 посилань на літературні джерела.

**Ключові слова**: *Арканоїд, JavaScript, гра, клас, функція, метод, змінна, значення, об’єкт, текстура, аудіо-файл, платформа, кулька, цеглина, бонус, бали, рекорд, життя, рівень.*

**ЗМІСТ**

## СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

| | | |
|---|---|---|
| JS | – | JavaScript |
| HTML | – | HyperText Markup Language |
| CSS | – | Cascade Style Sheets |
| VSC | – | Visual Studio Code |
| Баг | – | помилка, вада або дефект в комп'ютерній програмі або системі, що викликає в ній неправильний або неочікуваний. |
| Геймплей | – | ігровий процес, особливості взаємодії людини з відеогрою. |

# ВСТУП

Дана робота є актуальною, оскільки конструювання відеогри стандартними способами програмування є надійним способом здобути та закріпити навички роботи з обраною мовою програмування, зіткнутись з складними та комплексними проблемами та навчитись їх розв’язувати.

Метою роботи є відтворення класичної гри “Арканоїд” та її доповнення/модифікування на власний розсуд. В грі має бути платформа, керована гравцем комп’ютерною мишею, кулька розташована на платформі та декілька рядів цеглинок різних кольорів у верхній частині екрану. При запуску гри, куля почне свій рух вверх, і при зіткненні з цеглинами, стінами ігрового поля чи платформою змінить напрямок свого руху. Кожна цеглина має свою міцність та відповідний колір, і при контакті з шаром буде втрачати міцність, - доки не розіб’ється. Також, при контакті з цеглиною з невеликою вірогідністю може з’явитись один з багатьох бонусів, які будуть підсилювати гравця, якщо їх зловити. Суть Арканоїду полягає в тому, щоб розбити всі цеглини на ігровому полі й не дати шару впасти вниз, відбиваючи його платформою.

Задачі для досягнення поставленої мети:

1. Ознайомитись з оригінальною грою “Арканоїд”, його механіками, контентом та виглядом;

2. Дослідити способи створення схожих проектів використовуючи javascript, загальноприйняті способи реалізації основних ігрових елементів;

3. Створити текстури основних об’єктів для майбутнього використання;

4. Знайти та завантажити, або створити необхідні звукові ефекти;

5. Створити HTML файл, слугуючий сторінкою на якій буде знаходитись гра;

6. Створити CSS файл для стильового оформлення веб-сторінки;

7. Створити JS файл який буде містити в собі код гри та написати даний код, паралельно тестуючи та покращуючи його.

# 1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

## 1.1. Призначення та область застосування пристрою

Дана робота призначена для того, щоб покращити та закріпити наявні навички програмування на мові javascript. Розробка ігор без використання зовнішніх програм-конструкторів - це чудовий спосіб отримання досвіду роботи з мовами програмування. В багатьох випадках реалізація задуманих механік у відеогрі потребує важких роздумів та багатьох спроб використань різних методів й підходів. Результатом цього є покращення загального розуміння синтаксису коду і набуття багатьох способів вирішення проблем, з якими програміст зіткнеться в майбутньому.

Тривала розробка якісної гри потребує вивчення роботи з бібліотеками, плагінами, даними та їх типами, файловою системою, взаємодією об’єктів, зв’язками компонентів проекту між собою та іншим.

Додатково, невід’ємними частинами конструювання гри є графічний, аудіо та гейм-дизайн.

Графічний дизайн є незамінним, оскільки відповідає за вибір доречних і якісних зображень, а також їх редагування чи створення.

Знаходження й редагування аудіо-файлів теж може бути корисним для деяких функцій програм.

Гейм дизайн потрібний для чіткого планування всього проекту, його сумісність з майбутніми оновленнями і передбачення всіх можливих взаємодій між певними елементами чи дій виконаних користувачем для створення якісних умов використання, відсутності багів, помилок та незапланованих явищ.

Основне можливе застосування проекту - його максимальне покращення, завершення і публікація. Для цього потрібно переробити код проекту для сумісності з мобільними системами та інтерфейсом,

використовуючи існуючий проект як основу. Потім, добавити достатньо нових функцій для унікальності гри та розширити геймплей з розрахуванням на повторне і тривале використання (наприклад безкінечною генерацією рівнів з елементами випадковості). Обов'язковим є видалення/переробка всіх елементів гри, які будуть конфліктувати з іншими вже існуючими схожими проектами, або які заборонені у використанні в публічних цілях (наприклад певні зображення й аудіофайли завантажені з інтернет джерел). При успішному виконанні всіх вищезазначених пунктів, можлива публікація гри в крамниці застосунків, такій як "Playmarket", та її монетизація з допомогою вбудованої реклами.

### 1.2. Огляд існуючих аналогів

Очевидним аналогом який потрібно оглянути є оригінальна версія гри, «Arkanoid» (див. рис. 1.1). Попри те, що гра була створена ще в 1986 році для аркадних ігрових автоматів, основна концепція та механіка гри досі широко використовується в сучасних мобільних іграх та міні-іграх.


Рисунок 1.1. Офіційний аркадний флаєр гри

Початок гри представляє гравцю невелику сюжетну лінію, в якій космічний корабель «Арканоїд» був знищений, і менший корабель «Ваус» відокремився від попереднього в надії врятуватись, але потрапив у просторову пастку встановлену кимось невідомим (див. рис. 1.2). Після представлення, запускається перший рівень (див. рис. 1.3).


Рисунок 1.2. Стартовий екран гри

Гравець контролює невелику платформу, розташовану в нижній частині екрану. Платформу можна рухати вліво чи вправо, до контакту з стінами рівня. На платформі розташована кулька, яку гравець може довільно запустити вверх. У верхній частині екрану розташована велика кількість цеглинок різних кольорів, удар кульки по цеглинці призводить до того що остання ламається. В деякі цеглинки потрібно поцілити більше одного разу. Кулька відбивається від платформи, всіх цеглинок та поверхонь ігрового поля, крім нижньої поверхні. Вилітання кульки за нижню частину ігрового поля призводить до втрати життя гравця, а отже його задача - відбивати кульку платформою доки вона не розіб’є всі існуючі цеглинки й не дати їй впасти вниз.

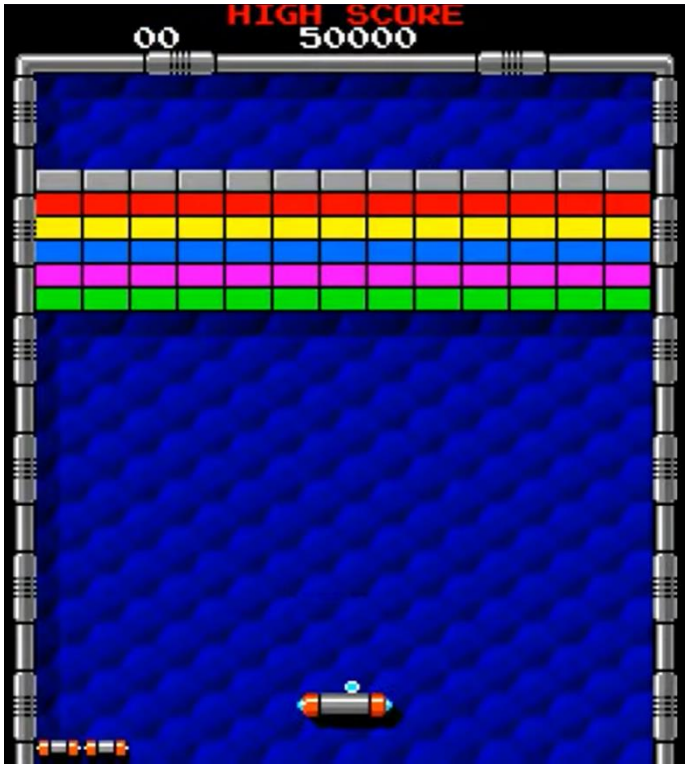



Рисунок 1.3. Зображення ігрового поля на початку гри

Завантаживши комп’ютерний порт гри [1], пройшовши її до кінця було досліджено її додаткові механіки:

- Гравець починає гру з 3 життями. Кількість життів можна побачити по шкалі в лівій нижній частині ігрового поля.
- Чим ближче зіткнення кульки та платформи до центру платформи, тим менше буде горизонтальне відхилення кулі.

У відхилення є мінімальне значення, встановлене щоб виключити можливість запускати кульку по ідеальній вертикалі. Виходячи з цієї інформації, не завжди вигідно відбивати шар краєм платформи, оскільки його буде важче відбити, але цю техніку можна використовувати щоб скерувати його в потрібне місце, як нішу в ряду цеглинок.

- Швидкість кульки залежить від кількості цеглинок, які залишились на ігровому полі. Чим менше цеглинок - тим більша швидкість кулі.
- При розбиванні цеглинки існує невелика вірогідність, що з неї випаде

випадковий бонус, який почне повільно падати вниз.

- Бонусів є всього 7:

1) Добавляє бонусне життя, максимум до шести.
2) Розширює платформу в два рази до завершення рівня або підбирання іншого бонусу.
3) Платформа отримує можливість стріляти лазерами. При натисканні на пробіл два паралельні лазери вилітають з платформи розбиваючи будь-яку цеглинку при контакті.
4) Гравець отримає можливість пройти в прохід справа і автоматично перейти на наступний рівень.
5) Сповільнює кульку приблизно на 30-50% до завершення рівня чи підбирання іншого бонусу.
6) При кожному зіткненні кульки з платформою, вона відбивається не одразу, а при натисканні на пробіл кнопку миші або автоматично після 2-3 секунд. Це дозволяє розмістити платформу на вигідну позицію перед випуском кулі.
7) Кулька розділяється на три ідентичні кульки, кожна з двох нахилена на 45° вліво і вправо. Кожна кулька функціонує ідентично, відбиваючись від поверхонь та розбиваючи цеглинки. В разі вильоту кулі за ігрове поле, життя гравця зникне лише якщо це була остання кулька на ігровому полі, що означає що наслідків за два пропуски після використання бонусу не буде.

У події випадіння бонусу є певний час перезарядки, очевидно з випадків коли шар залітає у вигідну позицію і ламає величезну кількість цеглинок за короткий період часу (див. рис. 1.4).

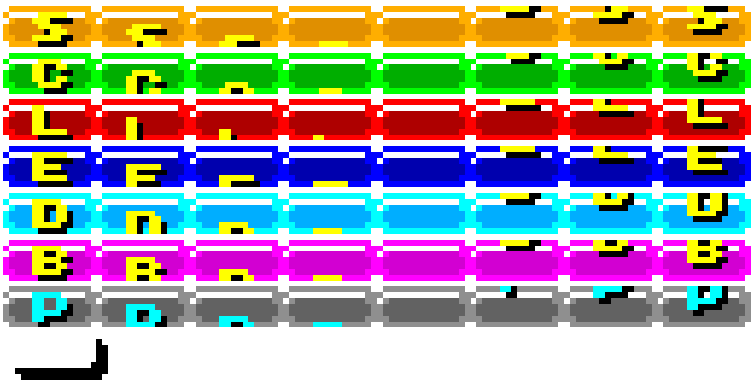


Рисунок 1.4. Офіційні текстури всіх бонусів, упорядковані в спрайт-лист для анімації падіння

- Час від часу, з верхньої частини ігрового поля вилітають ворожі об’єкти (див. рис. 1.5). Кожен з них повільно рухається в певному напрямку. Це може бути або напрямок запрограмований з самого початку рівня, де ворог ніби рухається по невидимому, заданому творцем рівня шляху, або ж рух імітуючий фізику ковзання по цеглинках, відбивання від стін та падіння вниз, у разі відсутності цеглинок під собою. Призначення таких об’єктів одне - зіткнутись з кулькою і, можливо, нашкодити таким чином гравцю.

Рисунок 1.5. Офіційні текстури всіх ворогів, упорядковані в

спрайт-лист для анімацій руху та знищення

- Всього в грі 32 рівня та фінальна битва з «босом». Кожен рівень має одне з 4 фонових зображень (див. рис. 1.6), один з 4 видів ворогів (якщо присутні в рівні) та унікальне розміщення різної кількості та видів цеглинок.

Зазвичай цеглинки розміщені таким чином, щоб дати гравцю можливість скерувати кульку в певну область, де кулька застрягне на деякий час відбиваючись від цеглинок у всіх напрямках. В таких випадках велика частина рівня буде пройдена завдяки явищу рикошету кульки без втручання гравця, поки куля не знайде або не проб’є вихід з структури цеглин.

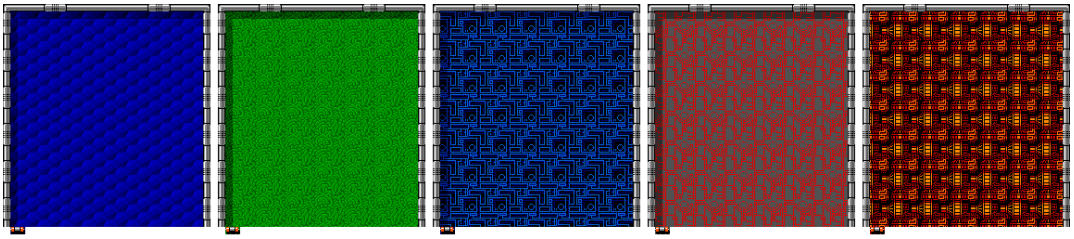


Рисунок 1.6. Текстури всіх фонових зображень

- Є 8 видів звичайних цеглинок (див. рис. 1.7), які ламаються з першого удару: біла, голуба, синя, зелена, жовта, помаранчева, червона та пурпурна. Срібна цеглина ламається з 3 ударів. Золоту цеглину зламати неможливо.

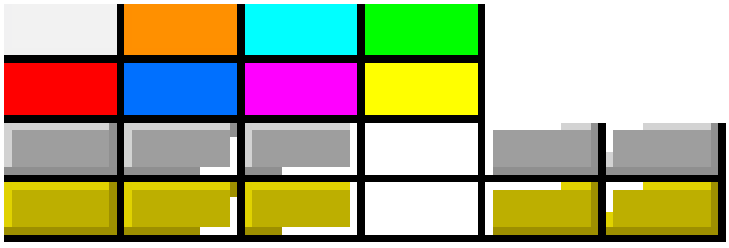


Рисунок 1.7. Офіційні текстури всіх цеглинок в грі

- Фінальний рівень має унікальне фонове зображення і не має жодної цеглинки - замість них у верхній центральній частині екрану розташований фінальний «бос» гри у вигляді голограми космічного шолому (див. рис. 1.8).

Кожні 2 секунди «бос» стріляє чергою з п’яти снарядів, контакт з якими призводить до втрати життя. В більшості випадків, снаряд цілиться в місцеположення платформи на момент вистрілу, але вистріл може відбутись і лівіше/правіше очікуваної локації. Для перемоги над «босом», потрібно попасти кулькою в його верхню частину лиця сім разів. Зіткнення з босом супроводжується невеликою анімацією отримання шкоди. Космічний шолом поводиться як твердий об’єкт, а отже кулька буде відбиватись від нього як від цеглинок чи стін ігрового поля.

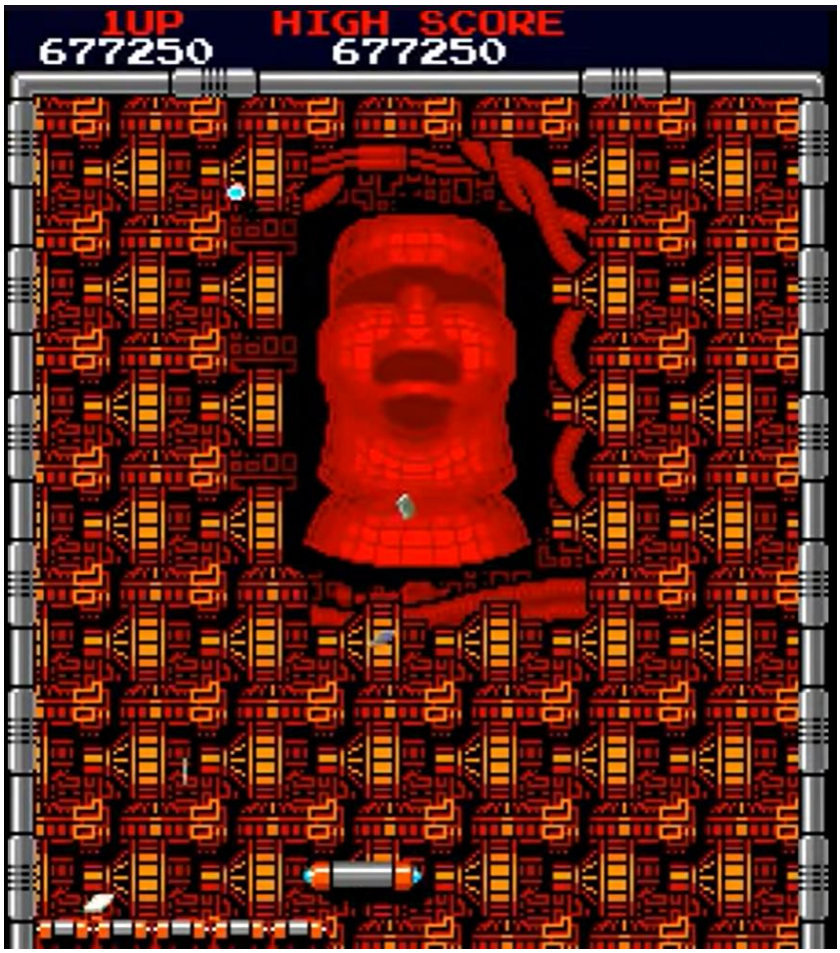



Рисунок 1.8. Зображення фінального рівня та битви з босом

- Після перемоги над босом, оповідач сповіщає що космічний корабель гравця вирвався з пастки та гра завершується.
- В грі існує система балів та їх рекордної кількості - їх можна побачити у верхній частині екрану над ігровим полем. Бали нараховуються за розбивання цеглинок та ворожих об’єктів. Зазвичай кожен розбитий об’єкт надає 100 балів, але їхня кількість може зростати залежно від кольору цеглини. Після закінчення гри, гравцю надається можливість зберегти свій рекорд та вписати своє ім’я (див. рис. 1.9).

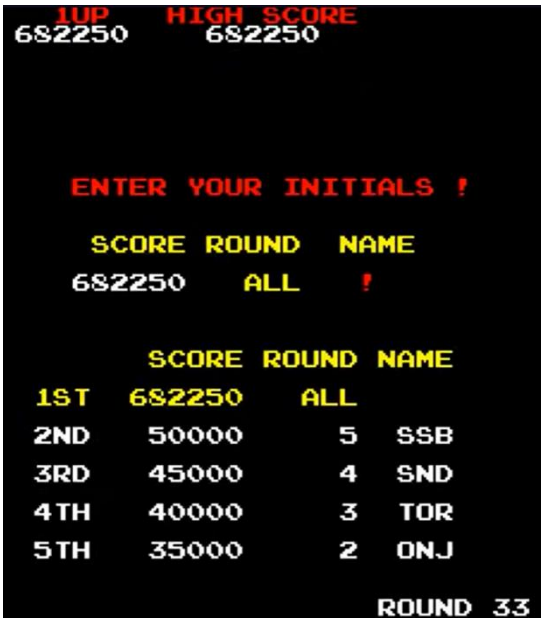



Рисунок 1.9. Фінішне меню гри, з очками гравця
та рекордами всіх попередніх гравців на пристрої

Проаналізувавши гру, я повністю ознайомився з її механізмами та отримав уявлення її рекреації. Щодо недоліків, я вважаю що бонусів можна добавити більше, а рух платформи, що йде за курсором миші має бути швидшим. Також, смужка життів в оригінальній грі візуально зображає на одне життя менше, ніж справжній запас гравця.

## 1.3. Постановка завдання

У висновку, згідно попереднього аналізу доцільності та плану моєї роботи, моєю метою є приблизне відтворення гри Арканоїд, з дозволеними довільними змінами.

У грі має бути:

- Платформа, керована гравцем
- Кулька, з фізикою руху по екрану та відбивання від поверхонь
- Цеглини, з різними кольорами та значеннями міцності
- Бонус розширення платформи
- Бонус підсилення кульки, дозволяючий їй розбивати будь-які цеглинки з одного удару
- Бонус прискорення кульки
- Бонус вибуху, заряджаючий кулю змушуючи її створити вибух при наступному контакті з цеглиною
- Бонус лазерів, дозволяючий платформі стріляти лазерами
- Бонус аннігіляції, дозволяючий випустити з платформи руйнівний промінь знищуючий все на своєму шляху
- Бонусне життя
- Бомба, зменшуюча життя гравця на 1 при контакті з платформою
- Функція паузи гри
- Консоль розробника, дозволяюча створити бонуси та змінити швидкість кульки клавішами на клавіатурі для тестування гри
- Система очків та рекорду
- Система життів та програшу і рестарту гри

Основні пункти розробки:

1. Ознайомлення з оригінальною грою “Арканоїд”
2. Дослідження створення ігор з використанням javascript

3. Створення текстур та аудіофайлів
4. Створення та наповнення HTML й CSS файлів
5. Створення JS файлу і написання коду програми
6. Тестування роботи та завершення готового проекту

# 2. ОПИС І ОБГРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ

## 2.1. Вибір засобів та мови програмування

Для даного проекту, було вирішено використовувати мову програмування javascript. Ця мова є одним з найкращих виборів для створення веб-ігор, так як виконується безпосередньо в браузері клієнта. JS документи зазвичай пов’язуються з HTML сторінкою веб-сайту, слугуючи своєрідним кодовим ядром, відповідальним за всі функції, поведінки та дані. Сама ж HTML сторінка слугує основним полотном, де розташований ввесь вміст проекту. В більшості випадків до таких документів прив’язують CSS файл, відповідальний за стильове оформлення певних елементів веб-сторінки.

Причини вибору мови JavaScript:

- Легка у вивченні та використанні;
- Серед безлічі різноманітних мов програмування, javascript є далеко не найважчою і відносно легко та інтуїтивно сприймається. В деяких випадках, синтаксис не такий строгий як в інших мовах [17];
- Відома та широко застосовувана;
- 99% всіх вебсторінок використовують javascript для реалізації певних поведінок на сайті;
- Інструкції, відео та посібники по javascript наявні у великих кількостях і легко знаходяться. В зв’язку з її популярністю багато досвідчених користувачів можуть поділитись способами вирішення поширених проблем чи реалізації певних методів;
- Підтримувана веб-браузерами;
- Javascript підтримує всі сучасні існуючі браузери і виводить на кожному з них ідентичний результат;
- Відкритий доступ.

В більшості випадках, користувач зможе побачити js код будь-якої не сильно захищеної сторінки. Це дозволяє вивчити та розібрати функціонал

існуючих проектів. З іншої сторони, цю причину можна вважати негативною рисою.

Основним недоліком цієї мови є її чутливість - зазвичай навіть найменша помилка синтаксису коду чи проблема обробки даних призводить до повної відмови рендеру веб-сторінки. [5]

Як середовище програмування, було обрано Visual Studio Code. Це найпопулярніший безкоштовний редактор вихідного коду створений компанією Microsoft.

Переваги VSC:

- Стандартний та надійний
- Visual Studio Code є одним з найбільш використовуваним інструментом в програмуванні, згідно щорічним опитуванням. Він був випробуваний безліччю програмістами, надійний у використанні та легкий у завантаженні
- Підтримує більшість сучасних мов програмування, звісно ж включаючи javascript
- Простий та зручний використанні, має високий функціонал

Це середовище програмування має дуже багато різних функцій, інструментів і плагінів, розібратись у використанні яких займе багато часу. Але, комфортно користуватись ним можуть програмісти будь якого рівня, і у випадку відсутності необхідності у використанні цих інструментів та функцій, VSC можна використовувати як звичайний редактор коду, з колірним розрізненням блоків та частин коду, підсвіченням помилок, консоллю, дебагером й автодоповненням.

Основний недолік Visual Studio Code:

Як було зазначено раніше, дане середовище можна не тільки використовувати як звичайний редактор, а і сильно збільшити ефективність та можливості, використовуючи просунуті функції. Але, для початківця, конфігурація та розбір таких доповнень може бути складним, тривалим і

потребувати певного часу, щоб успішно налаштувати все потрібне до роботи.

Для створення більшості текстур для гри, було використано мобільний додаток «IbisPaint X». Ця програма є загальноприйнятим основним професійним редактором зображень, використовувана як для малювання, так і для покращення зображення.

Переваги «IbisPaint X»:

- IbisPaint X є найкращим фоторедактором в своїй сфері не маючи конкурентів
- Безкоштовна й багатофункціональна
- Не дивлячись на відсутність ціни, містить в собі всі необхідні основні інструменти і більше для малювання та редагування

Для комбінації декількох аудіо-файлів в один, був використаний мобільний відеоредактор «KineMaster». Використавши чорне зображення як основу, було створено відео з потрібним аудіо. За допомогою онлайн-конвертора «Convertio» [14] виконано перетворення відео-файлу формату mp4 в аудіо-файл формату mp3. Для редагування швидкості, тону та інколи довжини звуку використано онлайн аудіоредактор «MP3Cut» [15].

Для маніпуляцій потрібних GIF-зображень, завантажених з зовнішніх джерел, і їх автоматичного розділення на кадри, було використано онлайн-редактор «EzGif» [16]. Це дозволило перетворити GIF-файл у покадровий лист зображень, для майбутнього використання в анімації певних об’єктів.

Для загального розуміння створення ігор використовуючи javascript, я використав YouTube плейлисти відеоуроків декількох каналів, присвячених створенням простих відеоігор [6]. Мета всіх відео плейлистів - створення простої гри типу платформер, з головним персонажем, керованим гравцем,

рівнем з перешкодами, ворогами, тощо

В результаті ознайомлення з відеоуроками, я збагатив знання мови javascript, навчився правильно застосовувати класи об’єктів, вивчив ефективний спосіб завантажень всіх зображень в гру до її початку, навчився створювати анімовані об’єкти використовуючи спрайт-листи, та інше.

Принцип написання коду:

Перед початком написання коду, було створено список обов’язкових та необов’язкових речей, які потрібно реалізувати в даному проекті. Створивши основні об’єкти та їхні механіки, була збережена копія коду як запасна. Продовжуючи працювати з кодом були додані нові функції вже існуючим об’єктам, потім додані додаткові елементи гри. При виникненні помилок під час тестування коду, я користувався дебаг-консоллю.

При зустрічі з помилками, які важко було вирішити самостійно без додаткової інформації, були використані сайти “StackOverflow” [2], “W3Schools” [3] і “Mozilla Developer Network” [4] для ознайомлення з рішеннями схожих проблем, виконаними досвідченими програмістами.

Більшість зображень були завантажені з архіву текстур [8], а деякі анімовані gif-зображення - з сайту “GIFER” [9]. Зображення, які складаються в основному з тексту були створені за допомогою сайту “TextStudio” [12].

Декілька оригінальних звуків були отримані з відео звуків арканоїду [10], але для більшості я звертався до вихідних кодів чи архівів популярних піксельних ігор, як Minecraft, Terraria, Calamity Mode чи Undertale [11].

Піксельний шрифт, використаний у проекті заватажений з бібліотеки шрифтів [13].

## 2.2. Опис функціонування системи

Всі файли й компоненти програми повинні бути зібрані в одній папці. Створимо папку “ARKANOID” та наступні елементи:

- ARKANOID.html: веб-сторінка нашої гри, полотно на якому будуть відбуватись всі ігрові події та розташовані об’єкти
- style.css: файл таблиць стилів, призначений для легкого стильового оформлення веб-сторінки
- script.js: JavaScript файл, який буде містити ввесь код гри
- images: папка для зберігання всіх зображень використовуваних грою
- sounds: папка яка містить всі аудіофайли для програми
- fonts: папка з tff документом восьми-бітного шрифта, потрібного для текстових надписів у грі

Наповнення перших двох файлів буде мінімальним, так як ввесь основний контент буде міститись в третьому. HTML файл буде містити всі теги потрібні для створення веб-сторінки, її назву, браузерну іконку, ширину, висоту, та зв’язки з іншими двома файлами. CSS файл оформить ігрове поле, встановить фоновий колір решти сторінки на чорний та підключить до сторінки завантажений шрифт.

Починаючи планування js коду, спочатку потрібно буде встановити зв’язок з css файлом та оголосити всі змінні, які знадобляться в майбутньому на різних стадіях оформлення коду.

Далі оформимо класи всіх об’єктів. Деякі елементи, як цеглинки, будуть наявні в грі у великій кількості, тому для кожного необхідно створити відповідний клас з властивостями елементу для легкого створення об’єктів. У всіх елементів будуть властивості висоти, ширини, положення по координаті x та положення по координаті y. Інші властивості будуть залежити від об’єктів, наприклад: в платформи буде багато властивостей та функцій відповідальних за її стан і вигляд, куля буде мати швидкості горизонтального та вертикального польоту, тип цеглини, тощо.

Наступний крок - створити клас самої гри. В ньому будуть знаходитись всі функції відповідальні за роботу гри. Оголосимо змінні, які будуть потрібні

для майбутньої роботи, які будуть незмінними на протязі ігрового процесу до закриття і повторного відкривання сторінки. Значенням як бали, рекордні бали, життя та поточний рівень даємо можливість зберігання в пам’яті проекту, щоб вони не втрачались навіть при закритті гри чи перезавантаження сторінки.

Створимо функцію GameStart, відповідальну за оголошення решти значень, класів, списків та створення початкових елементів гри як платформа та кулька. Функція розрахована на повторне використання під час ігрового процесу.

Наступна функція - BuildBricks, яка буде створювати структуру з цеглинок для розбивання. Форма та зміст цеглинок буде залежити від рівня.

Функція SetBG буде встановлювати одне з п’яти фонових зображень відповідно до номеру рівня.

GamePause буде відповідальна за можливість зупинити гру і висвітлення інтерфейсу паузи.

AlignBall - маленька функція, яка буде вирівнювати кульку по центру платформи.

BrickCollision буде виконуватись при зіткненні з цеглиною будь якого об’єкту, який має з нею контактувати. Завдяки цій функції цеглини будуть втрачати міцність при зіткненні з кулею чи лазерами. Також тут буде прописана вірогідність випаду бонусу та його типу.

InputDeviceControl - дуже важлива функція потрібна для зчитування вхідної інформації від користувача. Вона буде реєструвати рух комп’ютерної миші та натискання певних клавіш на клавіатурі.

Функція LoadImages буде містити назву та шлях кожної текстури ігрових елементів та завантажить всі зображення після їх готовності.

Playsound буде відповідальна за програвання звукових файлів.

Остання функція яку потрібно буде створити - Loop. ЇЇ особливість полягає в тому, що вона працює без зупинки. Тому, в ній буде розташований

ввесь код відповідальний за все, що відбувається у грі і потребує постійного оновлення. В ній потрібно створити наступне:

- Механіка руху кульки, додаючи до її координат значення її швидкостей в горизонтальному та вертикальному напрямках.
- Фізика відбивання кулі від рамок ігрового поля, платформи та цеглин.
- Ініціалізація всіх бонусів, з такими значеннями як виконувана подія, умова підбирання, зображення та звук. Їхній вертикальний рух вниз, механіка підбирання та автоматичного видалення з часом.
- Підтримка та функціонал додаткових об’єктів як лазер, вибух, знищувальний промінь та інші елементи утворені в процесі гри.
- Умова поразки, зниження кількості життів гравця та повний рестарт при відсутності життів.
- Умова виграшу (відсутність цеглин в рівні) та перехід на наступний.
- Постійне присвоєння зображень всім існуючим об’єктам.
- Підрахунок балів та рекордних балів.

Реалізувавши всі вищенаведені пункти, проект повинен функціонувати виконуючи план роботи.

# 3. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

## 3.1. Опис функціонування проекту

В наступному описі представлений не ввесь код, а лише його фрагменти відповідальні за основні функції. Деякі частини не вказані або скорочені.

Почнемо проект створивши js файл і підключивши css для створення ігрового полотна, а також створимо деякі змінні для роботи в майбутньому. (див. Додаток А)

```
var canvas = document.getElementById('game');
var ctx = canvas.getContext('2d');
canvas.style.cursor = 'none';
var canvasWidth = 760;
var canvasHeight = 970;
var scaleX = (window.innerWidth) / canvasWidth;
var scaleY = (window.innerHeight) / canvasHeight;
var scale = Math.min(scaleX, scaleY);
canvas.style.transformOrigin = '0 0';
canvas.style.transform = 'scale(' + scale + ')';
canvas.width = canvasWidth;
canvas.height = canvasHeight;
let gameframe = 0;
var Pause = false;
var DevConsoleSpeed = 0;
var KeyUnpressed = {};
```

Створимо класи всіх об’єктів з їх відповідними значеннями.

```
class PLATFORM {…}
class BALL {…}
class BRICK {…}
class EXPLOSION {…}
class BOMBEXPLOSION {…}
class LASER {…}
class ANNIHILATORRAY {…}
class BONUS {
…
  class ENLARGEBONUS extends BONUS {}
  class POWERBALLBONUS extends BONUS {}
  class SPEEDBONUS extends BONUS {}
  class BOOMBONUS extends BONUS {}
  class LASERBONUS extends BONUS {}
  class ANNIHILATEBONUS extends BONUS {}
  class BONUSBALL extends BONUS {}
  class LIFE extends BONUS {}
  class BOMB extends BONUS {}
```

Для прикладу, зобразимо клас кульки:

```
class BALL {
  constructor(x, y) {
    this.x = x; this.y = y;
    this.speedX = 0; this.speedY = 0;
    this.BaseSpeed = 10;this.size = 18;
    this.AccelerationValue = 1;
    this.SlowValue = 1;
    this.frameX = 0;
    this.CanCollide = true;
    this.isPowerBall = false;
    this.isAccelerated = false;
    this.isSlow = false;
    this.isCharged = false;
    …
    this.AccelerationValue = 1.5;
    this.trail = []}
ChangeDirection(Coordinate) {
this.CanCollide = false;

this['speed' + Coordinate] = -this['speed' + Coordinate];
   setTimeout(() => {this.CanCollide = true}, 30)}
PowerBall() {…}
Acceleration() {
clearTimeout(this.accelerationTime);
if (this.isAccelerated) this.AccelerationValue = 1;
this.isAccelerated = true;
this.accelerationTime = setTimeout(() =>{
this.AccelerationValue = 1;
this.isAccelerated = false;
this.trail = []}, 10000)}
  SlowDown() {…}
  Start(){this.speedX = 0; this.speedY = -this.BaseSpeed}}
```

В цьому класі ми визначаємо об’єкт “Куля” з значеннями координат, розміру, швидкостей вертикального та горизонтального напрямків, та інші, наприклад AccelerationValue. При підбиранні бонуса прискорення, воно зміниться на 1.5. Оскільки під час розрахунку швидкості в майбутній частині коду AccelerationValue входить у формулу, швидкість кулі підвищиться на 50%.

Інший приклад: CanCollide, визначає чи може шар контактувати з цеглинами. В подальшому буде відключатись на малий проміжок часу після зіткнення з цеглиною. FrameX - важлива змінна, присутня у всіх об’єктів маючих анімацію. Анімовані елементи мають список текстур (спрайт-лист) замість звичайної текстури, і при початку анімації frameX росте, що враховується функцією малювання зображення елемента, і тому область зображення на спрайт-листі зміщується вниз в залежності від значення frameX. Тепер розберемо одну функцію притаманну лише об’єкту кулі, наприклад PowerBall(). При запуску функції куля перетвориться в суперкулю і збільшиться в розмірі. Запуститься таймер на 20 секунд, після закінчення якого куля повернеться в нормальний стан.

```
class GAME {
  constructor () {
    this.images = [];
    this.score = localStorage.getItem('score');
    if (this.score == null) {this.score == 0}
    else this.score = parseInt(this.score);
    …
    this.NextAttempt = false;
    this.InputDeviceControl();
    this.GameStart(); }
  GameStart() {
      this.LoadImages(this.images, () => {
          this.BricksCount = 0;
          this.NextLevelTimer = 0;
          this.textFade = 1;
          this.balls = [];
          this.enlargebonuses = [];
…
this.platform = new PLATFORM(0, 0);
this.ball = new BALL(0, 0);
this.BuildBricks();
this.Loop();
this.PlaySound('sounds/GameStartSound.mp3');
});}
```

В класі GAME при створенні об’єкту гри, дані що не знаходяться в GameStart завантажаться лише один раз до перезапуску сторінки. В свою чергу GameStart влаштований так, що при повторному виконанні всі його значення скидаються за замовчуванням. Додатково, значення очків, рекордних очків, життів та номеру рівня зберігаються в пам’яті проекту, що дозволяє їх зберегти і використати навіть якщо сторінка була перезавантажена чи закрита.

```
this.bricks = [];
      switch (this.level) {
        case 1:
          for (let i = 0; i < 4; i++) {
            let y = i * 2 * 25 + i + 130;
            for (let j = 0; j < 13; j++) {
               let x = j * 54 + 30;
            this.bricks.push(new BRICK(x, y, i, this.w, this.h))}}
          break;}
      this.BricksCount = this.bricks.length}
```

BuildBricks будує структуру з цеглинок різних типів, створюючи їхні рядки та стовпці використовуючи математичні формули. Структура залежить від поточного рівня, в зразку показано лише один випадок.

```
Loop() {
      gameframe++;
      if (gameframe >= 10000) gameframe = 500;
      if (Pause == true) return;
      ctx.clearRect(0, 0, canvas.width, canvas.height);
      if (!this.Start && !this.NextAttempt) {
         const startScreen = this.images['start'];
         ctx.drawImage(startScreen, ((canvas.width - startScreen.width) / 2), ((canvas.height - startScreen.height) / 2 +
150));}
```

Функція Loop буде містити в собі код основних функцій гри. В фрагменті вище ми постійно збільшуємо gameframe - значення яке нам потрібне для всіх циклів анімації зображень. Далі код відповідає за правильну роботу паузи і виведення стартового тексту до початку гри

```
      if (this.ball.x + this.ball.size >= this.platform.x && this.ball.x <= this.platform.x + this.platform.w &&
         this.ball.y + this.ball.size > this.platform.y && this.ball.y < this.platform.y + this.platform.h &&
         this.platform.CanCollide && this.Start) {
               this.ball.speedX = this.ball.BaseSpeed * (this.ball.x - (this.platform.x + this.platform.w / 2)) * 2 /
this.platform.w;
          if (this.ball.speedX < 3 && this.ball.speedX > 0) this.ball.speedX = 3;
          else if (this.ball.speedX > -3 && this.ball.speedX < 0) this.ball.speedX = -3;
          this.ball.ChangeDirection('Y');
          this.platform.CollisionDisable();
          this.PlaySound('sounds/BallHitPlatform.mp3');
          if (this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3')}
```

Як зразок функції зіткнення кулі з об’єктом, обрана платформа. При контакті, швидкість кулі міняється на протилежну та програється відповідний звук. Також, реалізоване горизонтальне відхилення кулі при зіткненні: чим далі від центру кулі воно відбулось, тим сильнішим буде горизонтальне відхилення шара. Встановлене мінімальне значення для відхилення, щоб не

можливо було тримати кулю в ідеально вертикальному положенні відбиваючи її центром платформи. Схожими методами реалізовані контакти з цеглинами та рамками ігрового поля, але з своїми відмінностями та нюансами.

```
this.calculatedSpeedX = ((this.ball.speedX) + this.ball.speedX * DevConsoleSpeed) * this.ball.SlowValue * this.ProgressSpeed;
this.calculatedSpeedY = ((this.ball.speedY) + this.ball.speedY * DevConsoleSpeed) * this.ball.SlowValue * this.ProgressSpeed * this.ball.AccelerationValue;
this.ball.x += this.calculatedSpeedX;
this.ball.y += this.calculatedSpeedY;
```

Тут відбувається розрахунок вертикальної та горизонтальної швидкостей кулі - на значення швидкості впливає багато факторів, і після обробки додається до координат кулі, що в циклічному повторі призводить до її руху.

```
this.explosions.forEach((explosion, explosionIdx) => {
    ctx.drawImage(this.images['explosion'], explosion.frameX * explosion.w, 0, explosion.w, explosion.h, explosion.x, explosion.y, explosion.w, explosion.h);
    if (gameframe % 8 == 0) {
        explosion.frameX++;
        if (explosion.frameX >= 7) delete this.explosions[explosionIdx];}});
```

Вище наведений приклад поведінки елементів, мета яких - виконати свою анімацію та зникнути після її завершення. В деяких подібних об’єктів мета може бути розширена, наприклад політ та контакт з цеглиною для лазера.

```
const bonuses = [
    { array: this.enlargebonuses, … },
    { array: this.powerballbonuses, … },
    { array: this.speedbonuses, … },
    { array: this.boombonuses, … }, {
     array: this.laserbonuses,
     image: this.images['laserbonus'],
     onPickUp: () => {
        this.LaserCount = 20;
        if (this.platform.isLarge) {
            clearTimeout(this.platform.enlargeTime);
            this.platform.x += 55;
            this.platform.w = 110;
            this.platform.isLarge = false}
        if (this.platform.isAnnihilator) {
            this.platform.isAnnihilator = false;
            this.platform.frameX = 0}},
    condition: () => !this.platform.AnnihilatorTime,
    sound: 'sounds/LaserBonusPickUp.mp3'},
    { array: this.annihilatorbonuses, … },
    { array: this.lives, … },
    { array: this.bombs, … }];
```

В цьому коді створено список ігрових бонусів, в кожного з яких є назва, текстура, подія яка відбудеться якщо його підібрати, умова підбирання та унікальний звук. Для прикладу наведено бонус лазерної платформи: при контакті наша кількість вистрілів встановлюється до 20. Пізніше в коді, зміна буде помічена що результує в накладанні текстури лазерної платформи і

функції стрільби при натисканні пробілу на клавіатура. Всі попередні бонусні ефекти платформи анулюються, і її позиція вирівнюється.

```
bonuses.forEach(({ array, image, onPickUp, condition, sound }) =>
        {array.forEach((bonus, idx) => {
ctx.drawImage(image, bonus.frameX * bonus.w, 0, bonus.w, bonus.h, bonus.x, bonus.y, bonus.w, bonus.h);
            if (gameframe % 6 == 0) {
               bonus.frameX++; if (bonus.frameX >= 14) bonus.frameX = 0}
               bonus.Fall();
            if
               (bonus.y + bonus.h >= this.platform.y && bonus.y <= this.platform.y + this.platform.h &&
                bonus.x + bonus.w >= this.platform.x && bonus.x <= this.platform.x + this.platform.w && condition())
               {onPickUp(bonus);
               delete array[idx];
               this.PlaySound(sound)}
            if (bonus.y >= 1000)
              {delete array[idx]}})});
```

Ця частина коду відповідальна за виконання всіх подій та умов бонусів в списку наведеному вище, а також за падіння бонусів і їх видалення при вильоту за нижню частину екрану.

```
if (this.ball.y >= 1030) {
   this.ball.y = 1000;
   this.ball.speedY = 0;
   this.ball.speedX = 0;
   this.NextAttempt = true;
   this.platform.canMove = false;
   this.ball.trail = [];
   this.PlaySound('sounds/LifeLost.mp3');
   setTimeout(() => {{clearTimeout(this.powerballTime);
   clearTimeout(this.platform.enlargeTime);
   clearTimeout(this.accelerationTime);
   clearTimeout(this.slowDownTime);
   this.ball.AccelerationValue = 1;
   this.ball.isCharged = false;
   this.ball.isAccelerated = false;
   if (this.ball.isPowerBall) {
      this.ball.x += 5;
      this.ball.y += 5;
      this.ball.isPowerBall = false;
      this.ball.size = 18}
      if (this.platform.isLarge) {
      this.platform.x += 55;
      this.platform.w = 110;
      this.platform.isLarge = false}
            if (this.platform.isAnnihilator) {
               this.platform.isAnnihilator = false;
               this.platform.AnnihilatorTime = 0;
               this.platform.frameX = 0;}
            this.LaserCount = 0;
                this.lifecount -= 1;
                  localStorage.setItem('lifecount',
this.lifecount);
                this.AlignBall();
                this.ball.frameX = 1;
                                  if   (this.lifecount)
this.PlaySound('sounds/BallAppear.mp3');
                                  setTimeout(()   =>
{{ this.platform.canMove = true;
            this.Start = false}}, 1200)}}, 1500)}
```

Якщо куля вилітає нижче видимої ігрової зони - всі бонусні ефекти скидаються та кількість життів знижується на 1. За наявності життів куля з’являється на платформі і по закінченню анімації її появи гра продовжується.

```
if (this.lifecount <= 0) {setTimeout(() =>{{
this.GameStart()}}, 3000);
this.PlaySound('sounds/GameOver.mp3');
ctx.clearRect(0, 0, canvas.width, canvas.height);
this.score = 0;
                        const GameOverScreen =
this.images['gameover'];
                    ctx.drawImage(GameOver
Screen,
          ((canvas.width                    -
GameOverScreen.width) / 2),
          ((canvas.height                   -
GameOverScreen.height) / 1.3));
                    return}
```

```
localStorage.setItem('score', this.score);
this.lifecount = 3;
localStorage.setItem('lifecount', this.lifecount);
this.level = 1;
localStorage.setItem('level', this.level);
this.Start = true;
```

У випадку закінчення життів, зображення програшу висвічується на екрані і через короткий час гра починається заново, з єдиним збереженим прогресом - рекордними очками.

```
ctx.font = "30px EightBit";
ctx.fillStyle = 'white';
ctx.textAlign = 'left';
ctx.fillText(this.score, canvas.width / 2 - 300, canvas.height - 910);
ctx.fillStyle = 'red';
ctx.fillText("Рекорд: ", canvas.width / 2 - 60, canvas.height - 910);
ctx.fillStyle = 'white';
ctx.fillText(this.bestscore, canvas.width / 2 + 80, canvas.height - 910);
if (gameframe <= 200) {
   ctx.font = "90px EightBit";
   ctx.fillStyle = 'rgba(255, 255, 255, ' + this.textFade + ')';
   ctx.textAlign = 'center';
   ctx.fillText('Рівень ' + this.level, canvas.width / 2, canvas.height / 2)}
if (gameframe <= 200 && gameframe > 120) this.textFade -= 0.015;
if (this.score > this.bestscore) {
   this.bestscore = this.score;
   localStorage.setItem('bestscore', this.bestscore)}
```

Значення балів та рекодних балів циклічно виводяться над ігровим полем. У випадку якщо бали вищі ніж рекордні, рекорд оновлюється. Також в цій частині коду створений надпис, який виводить номер рівня на його початку.

```
const platform = (ctx, this.images[this.platform.isAnnihilator ? 'annihilateplatform' : this.platform.isLarge ?
'largeplatform' : this.LaserCount ? 'laserplatform' : 'platform']);
ctx.drawImage(platform, this.platform.frameX * this.platform.w, 0, this.platform.w, this.platform.h + 20,
this.platform.x, (this.platform.isAnnihilator ? this.platform.y-20 : this.platform.y), this.platform.w, this.platform.h +
20);
if (this.Start && gameframe % this.platform.staggerFrame == 0) {
   this.platform.frameX++;
   if (!this.platform.isAnnihilator && this.platform.frameX >= 2) this.platform.frameX = 0;
   else if (this.platform.isAnnihilator && this.platform.frameX >= 6) this.platform.frameX = 0}
this.ball.trail.forEach((position, index) => {
const ballImage = this.images[this.ball.isCharged ? (this.ball.isPowerBall ? 'chargedpowerball' : 'chargedball') :
(this.ball.isPowerBall ? 'powerball' : 'ball')];
const opacity = (index + 1) / this.ball.trail.length; ctx.globalAlpha = opacity;
ctx.drawImage(ballImage, this.ball.frameX * (this.ball.isPowerBall ? 30 : 24), 0, this.ball.size + this.ballSizeIncr,
this.ball.size + this.ballSizeIncr, position.x - this.ballSizePos, position.y - this.ballSizePos, this.ball.size +
this.ballSizeIncr2, this.ball.size + this.ballSizeIncr2)});
ctx.globalAlpha = 1;
const ball = (ctx, this.images[this.ball.isCharged ? (this.ball.isPowerBall ? 'chargedpowerball' : 'chargedball') :
(this.ball.isPowerBall ? 'powerball' : 'ball')]);
this.ballSizeIncr = this.ball.isPowerBall ? -6 : 6;
```

```
this.ballSizeIncr2 = this.ball.isPowerBall ? 0 : 6;
this.ballSizePos = this.ball.isPowerBall ? 0 : 3;
ctx.drawImage(ball, this.ball.frameX * (this.ball.isPowerBall ? 30 : 24), 0, this.ball.size + this.ballSizeIncr,
this.ball.size + this.ballSizeIncr, this.ball.x - this.ballSizePos, this.ball.y - this.ballSizePos, this.ball.size +
this.ballSizeIncr2, this.ball.size + this.ballSizeIncr2);
ctx.drawImage(this.images['lifebar'], 0, 0, this.lifecount * 35, 13, 49, canvas.height - 25, this.lifecount * 35, 13);
if (this.ball.frameX >= 1) {
   if (gameframe % 4 == 0) {
      this.ball.frameX++;
      if (this.ball.frameX >= 13) this.ball.frameX = 0}}
```

Код вище малює зображення майже кожного об’єкту, використовуючи gameframe та frameX для анімацій. Текстури елементів інколи відрізняються в залежності від значень його певних змінних - наприклад куля може мати звичайну текстуру, текстуру суперкулі, зарядженої кулі і комбінації попередніх двох. Також в неї є анімація появи після програшу і повторної появи на платформі, і ефект шлейфу під час прискорення.

```
if (this.bricks.filter(brick => brick.type !== 4).length === 0 && !this.NextLevelTimer) {
   this.PlaySound('sounds/LevelComplete.mp3');
   this.NextAttempt = true;
   this.NextLevelTimer = 1;
   this.lvlcmpFrameY = 0
   this.opacity = 0;}
   if (this.NextLevelTimer >= 1 && this.NextLevelTimer < 200) {
      this.ball.speedY = 0; this.ball.speedX = 0;
      this.NextLevelTimer++
      if (this.NextLevelTimer < 150) {
         const levelCompletedScreen = this.images['levelcompleted'];
         ctx.drawImage(levelCompletedScreen, 0, this.lvlcmpFrameY * 41,
         levelCompletedScreen.width, 41, (canvas.width - levelCompletedScreen.width) / 2,
         (canvas.height - 50) / 2, levelCompletedScreen.width, 50);
         if (this.lvlcmpFrameY < 6) {
            if (gameframe % 7 == 0) {
               this.lvlcmpFrameY++}}}}
            if (this.NextLevelTimer > 50 && this.NextLevelTimer < 150) {
               ctx.fillStyle = `rgba(0, 0, 0, ${this.opacity})`;
               ctx.fillRect(0, 0, canvas.width, canvas.height);
               this.opacity += 0.02}
             if (this.NextLevelTimer == 150) {
               if (this.level < 9) {
                  setTimeout(() =>{{this.GameStart()}},1);
                    this.level += 1;
                    localStorage.setItem('level', this.level);
                    ctx.clearRect(0, 0, canvas.width, canvas.height);
                    return;}
                else {
                  setTimeout(() => {{this.GameStart();}},4000);
                    ctx.clearRect(0, 0, canvas.width, canvas.height);
                    this.score = 0;
                    localStorage.setItem('score', this.score);
                    this.lifecount = 3;
                    localStorage.setItem('lifecount', this.lifecount);
                    this.level = 1;
```

```
localStorage.setItem('level', this.level);
this.PlaySound('sounds/WinSound.mp3');
ctx.drawImage(this.images['winscreen'],
((canvas.width - this.images['winscreen'].width) / 2),
((canvas.height - this.images['winscreen'].height) / 1.3));
return;}}
```

Даний код постійно перевіряє, чи залишились цеглини в рівні, і якщо не залишили - зупиняє кульку і висвічує “Рівень пройдено” зображення. Після цього екран темніє, і через короткий час гра повністю очищується і перезапускається, цього разу будуючи новий рівень. Рівні будуть збільшуватись до 9, після якого гра буде повністю пройдена, і після вітального зображення гравця ввесь прогрес скинеться до початкових значень крім рекордних балів.

```
window.requestAnimationFrame(() => {this.Loop();});
```

Завдяки цьому рядку, функція Loop циклічно повторюється

```
AlignBall()
    {
      this.ball.x = this.platform.x + this.platform.w / 2 - this.ball.size / 2;
      this.ball.y = this.platform.y - this.ball.size;
    }
```

Ця функція вирівнює кульку по центру платформи.

```
BrickCollision(idxBrick, source) {
   const brick = this.bricks[idxBrick];
   if (source === 'ball' && this.ball.isCharged) {
      this.ball.isCharged = false;
      const explosion = new EXPLOSION((this.bricks[idxBrick].x -98), (this.bricks[idxBrick].y -113));
      this.explosions.push(explosion);
      this.PlaySound('sounds/Explosion.mp3');
      this.bricks.forEach((brick, idxBrick) => {
   if (explosion.x + explosion.w > brick.x && explosion.x < brick.x + brick.w &&
      explosion.y + explosion.h >= brick.y && explosion.y <= brick.y + brick.h) {
      setTimeout(() => {this.BrickCollision(idxBrick, 'explosion')})}})}
   if (brick.type != 4 &&
   ((source === 'ball' && !this.ball.isPowerBall) || source === 'laser' || source === 'explosion')) {
      this.bricks[idxBrick].type--;
      this.score++;
       localStorage.setItem('score', this.score)}
   else if (brick.type != 4 && ((source === 'ball' && this.ball.isPowerBall) || source === 'annihilatorray')) {
      this.score = this.score + this.bricks[idxBrick].type + 1;
      localStorage.setItem('score', this.score);
      this.bricks[idxBrick].type = -1;
   if (source === 'ball' && this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3')}
         var BricksLeft = Math.round((this.bricks.filter(brick => brick.type !== 4).length * 100) / (this.BricksCount -
this.bricks.filter(brick => brick.type === 4).length));
       if (BricksLeft >= 17 && BricksLeft <= 34) this.ProgressSpeed = 1.6;
```

```
      else if (BricksLeft >= 35 && BricksLeft <= 50) this.ProgressSpeed = 1.48;
      else if (BricksLeft >= 51 && BricksLeft <= 67) this.ProgressSpeed = 1.36;
      else if (BricksLeft >= 68 && BricksLeft <= 83) this.ProgressSpeed = 1.24;
      else if (BricksLeft >= 84 && BricksLeft <= 100) this.ProgressSpeed = 1.12;
      if (brick.type != 4 && (Math.floor(Math.random() * 100) < 30 && !this.BonusCooldown)) {
         this.BonusCooldown = true;
         setTimeout(() => {this.BonusCooldown = false}, 4000);
          const BonusType = Math.floor(Math.random() * 9);
          const BonusClass = this.bonusClasses[BonusType];
          const bonusArray = this.bonusArrays[BonusType];
          const bonus = new BonusClass((brick.x + brick.w / 2), brick.y + brick.h);
          bonus.x -= bonus.w / 2;
          bonusArray.push(bonus)}}
```

Функція BrickCollision викликається кожен раз коли кулька, лазер чи промінь контактують з цеглиною. Спочатку перевіряється власник удару: якщо власником була звичайна куля чи лазер - в цеглини знижується міцність на 1, якщо ж власником була суперкуля чи промінь - цеглина розбивається моментально. Якщо власник-куля була зарядженою, на місці зіткнення створюється вибух. Після нанесення шкоди цеглинам, провіряється їхня кількість - швидкість кулі росте пропорційно до того як мало цеглинок залишилось, маючи максимум 50% прискорення. Останнє що відбувається - з невеликою вірогідністю з цеглини випадає випадковий бонус. Ця подія має перезарядку в 4 секунди.

```
InputDeviceControl() {
   document.addEventListener('mousemove', (mouse) => {
   if (this.platform.canMove) {
      if (this.platform.isLarge) {
         this.platform.x = Math.max(30, Math.min(mouse.screenX - canvas.width / 2, 512))}
      else if (!this.platform.AnnihilatorTime) {
         this.platform.x = Math.max(30, Math.min(mouse.screenX - canvas.width / 2 + this.platform.w / 2, 622))}
   if (!this.Start) this.AlignBall()});
      document.addEventListener('keydown', (Gamekeys) => {
         if (Gamekeys.key === ' ' && KeyUnpressed) {
            if (!this.Start) {
               this.ball.Start();
               this.Start = true}
            else if (this.Start && this.LaserCount) {
               KeyUnpressed = false;
               this.lasers.push(new LASER((this.platform.x + this.platform.w / 2 - 35), this.platform.y));
               this.lasers.push(new LASER((this.platform.x + this.platform.w / 2 + 35), this.platform.y));
               this.PlaySound('sounds/LaserSound.mp3');
               this.LaserCount--}
            else if (this.Start && this.platform.isAnnihilator && !this.platform.AnnihilatorTime) {
               this.platform.AnnihilatorTime++
               this.PlaySound('sounds/Annihilator.mp3');
               this.saveSpeedY = this.ball.speedY
               this.saveSpeedX = this.ball.speedX}}
         else if (Gamekeys.key === 'Escape' && KeyUnpressed) {
```

```
            KeyUnpressed = false;
            this.GamePause();
            if (Pause) this.PlaySound('sounds/PauseSound.mp3');
            else this.PlaySound('sounds/UnpauseSound.mp3')}});
        document.addEventListener('keydown', (DeveloperConsole) => {
          const keyIndex = parseInt(DeveloperConsole.key, 10) - 1;
          if (keyIndex >= 0 && keyIndex < this.bonusClasses.length && KeyUnpressed) {
            KeyUnpressed = false;
            const BonusClass = this.bonusClasses[keyIndex];
            const bonusArray = this.bonusArrays[keyIndex];
            const bonus = new BonusClass((this.platform.x + this.platform.w / 2), 100);
            bonus.x -= bonus.w / 2;
            bonusArray.push(bonus)}
          else if (DeveloperConsole.key === '+' || DeveloperConsole.key === '=') {
              DevConsoleSpeed = DevConsoleSpeed + 0.1;
              DevConsoleSpeed = Math.round(DevConsoleSpeed * 10) / 10}
          else if (DeveloperConsole.key === '-') {
              DevConsoleSpeed = Math.max(DevConsoleSpeed - 0.1, -1);
              DevConsoleSpeed = Math.round(DevConsoleSpeed * 10) / 10}});
        const UnpressKeys = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', ' ', 'Escape'];
        const Gamekeys = [' ', 'Escape'];
        document.addEventListener('keyup', (DeveloperConsole) => {
          if (UnpressKeys.includes(DeveloperConsole.key) ||
UnpressKeys.includes(Gamekeys.key)){KeyUnpressed = true}})}
```

InputDeviceControl призначена для зчитування інформації вводу користувача. Розділяється на два детектори - комп’ютерної миші та клавіш клавіатури. Перший потрібен для руху платформи, другий - для початку гри, стрільби з бонусних платформ та паузи. В коді також реалізована консоль розробника - допоміжний інструмент для тестування та швидкого проходження гри, дозволяючий створити будь який бонус натиснувши на відповідну клавішу цифри чи збільшити/зменшити швидкість кульки, клавішами “+” і “-”.

```
LoadImages(images, finishloading) {
   const imageNames = [ 'start', 'gameover', 'winscreen', 'pausescreen', 'levelcompleted', 'ball', 'powerball',
'chargedball', 'chargedpowerball', 'platform', 'largeplatform', 'laserplatform', 'annihilateplatform', 'lifebar',
'enlargebonus', 'powerballbonus', 'speedbonus', 'boombonus', 'laserbonus', 'annihilatebonus', 'bonusball', 'life',
'enemy', 'explosion', 'laser', 'annihilatorray', 'bomb', 'bombexplosion', 'bricks'];
   images = imageNames.map(name => ({ name, path: `images/${name}.png` }));
   const promises = [];
   Array.isArray(images) && images.forEach((image) => {
      promises.push(new Promise((load) => {
         var img = new Image();
         img.src = image.path;
         img.onload = () => { this.images[image.name] = img; load()}}))});
      Promise.all(promises).then(function() { finishloading()})}
```

Всі назви зображень та шляхи до текстур об’єктів знаходяться в цій функції. Замість того щоб присвоїти картинки одразу, LoadImages спочатку

завантажує всі зображення в окремий список, і тільки після повного завантаження і готовності, текстури малюються до потрібних елементів.

```
PlaySound(SoundLocation) {let audio = new Audio(SoundLocation); audio.play()}}
```

Ця функція існує для легкого доступу до аудіофайлів та програвання звуку.

Завершується код наступним рядком: `new GAME();`

Цей рядок створює об’єкт класа GAME, тобто починає саму гру і виводить її на полотно.

З додатку А2 видно, що .game оформлює ігрове полотно та його положення, міняє певні значення під впливом js коду. Body змінює фоновий колір всієї веб-сторінки на чорний. @font-face підключає файл з 8-бітним шрифтом з папки fonts.

Зміст HTML файла наведений в додатку А1. В цьому документі використовується стандартний набір тегів для створення веб-сторінки, задані її титул, ширина та висота, створений зв’язок з js та css файлами та встановлене зображення для іконки.

# 4. ДЕМОНСТРАЦІЯ ТА ВПРОВАДЖЕННЯ ПРОЕКТУ

Для демонстрації проекту необхідно зібрати всі його файли в архів та перенести на пристрій, який буде використовуватись для демонстрації. Розпакувавши проект на пристрої, необхідно запустити файл ARKANOID.html. Правильний стан роботи проекту включає:

- Ігрове поле має бути вирівняне по центру екрана і повністю видно по горизонталі та вертикалі. В іншому випадку, скоріше всього проблема в екрані пристрою та його налаштуванні
- Гра має запуститись з усіма зображеннями. В разі провалу, потрібно перевірити чи немає проблем з папкою images і її розташуванням, і чи немає факторів які можуть впливати на завантаження зображень.
- Всі функції гри мають працювати - завантаження зображень, прийому даних вводу, програвання аудіофайлів, циклу, і решти. Причиною неочікуваної поведінки чи багу може бути пристрій та його особливості роботи з веб-сторінками.

У разі запуску гри з усіма зображеннями, створення всіх об’єктів, правильного руху та фізики кульки, реагування на комп’ютерну мишу та клавіші, розбивання цеглин, функціонал бонусів, роботи шкали життів та умов програшу/виграшу, виконання мети можна вважати успішним.

Інструкція по завантаженні гри: Отримання архіву -> Розпакування -> Запуск ARKANOID.html

Результат роботи відео-гри наведений в Додатку В.

# ВИСНОВКИ

В результаті виконання роботи було сконструйовано версію класичної аркадної гри «Арканоїд» з довільними змінами.

Провівши аналіз мов програмування, їх особливостей та призначень, була обрана мова JavaScript завдяки її ефективності й зручності у використанні в розробці веб-ігор.

Під час створення гри «Арканоїд» було розширено та закріплено існуючі знання та навички програмування, ефективно використано класи та підкласи об’єктів, створено анімації для зображень, використано як основу для кожного коду функцію для підготовки зображень до їх використання, засвоєно роботу зі змінними, їх значеннями й доступністю та багаторазово компоновано великі подібні між собою частини коду в їхні менші та ефективніші версії вищого рівня знання мови програмування.

Готовий проект має платформу керовану гравцем, кулю з фізикою зіткнення з поверхнями, 9 бонусів, 9 рівнів з унікальним формуванням цеглин та систему життів, балів й рекордів. Гра не має на даний момент занотованих багів чи некоректної поведінки, тож згідно умов результат повністю задовільний і готовий до використання.

Порівнявши цей проект з іншими подібними, можна зробити висновок, що наступні кроки в його покращенні - реалізація меню та налаштувань, більше рівнів і їхня безкінечна випадкова генерація, оптимізація гри, більше унікальних елементів, покращення графіки й аудіо дизайну і головне - його адаптація для ширшого спектру пристроїв, в тому числі й для мобільних.

Розроблену гру рекомендовано використовувати в розважальних цілях, або ж як основу для модифікацій перелічених вище і її можливого опублікування.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Онлайн версія гри “Арканоїд”. URL: https://www.retrogames.cz/play_039-NES.php (дата звернення: 12.12.23)

2. Сайт для розробників StackOverflow. URL: https://stackoverflow.com (дата звернення: 23.04.24)

3. Навчальний сайт W3Schools. URL: https://www.w3schools.com/js/ (дата звернення: 12.04.24)

4. Ресурс для розробників Mozilla Developer Network (MDN). URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 23.04.24)

5. Навчальний ютуб-канал Programming with Mosh - JavaScript Tutorials. URL: https://youtube.com/playlist?list=PLTjRvDozrdlxEIuOBZkMAK5uiqp8rHUax&si=Zt4rcYcM9xz7Y-XE (дата звернення: 17.11.23)

6. Навчальний ютуб-канал freeCodeCamp.org - JavaScript Game Development Course for Beginners. URL: https://youtu.be/GFO_txvwK_c?si=3OnPd96HvTb8nE_m (дата звернення: 03.02.24)

7. Кадри оригінального ігрового процесу “Арканоїд” - Arcade Longplay Arkanoid. URL: https://youtu.be/Th-Z6QQ5AOQ?si=fyQKGbAr7L85q7Xt (дата звернення: 11.03.24)

8. Онлайн бібліотека текстур The Spriters Resource – Arkanoid. URL: https://www.spriters-resource.com/arcade/arkanoid/ (дата звернення: 15.12.23)

9. Онлайн бібліотека GIF зображень GIFER. URL: https://gifer.com/en/ (дата звернення: 01.02.24)

10. Колекція звукових ефектів гри “Арканоїд” - Arkanoid Arcade Sound Effects. URL: https://youtu.be/peiQJ0NfQf8?si=w5Xhcfej5sB72rwU (дата звернення: 15.12.23)

11. Онлайн бібліотека звукових файлів TheSoundsResource – Undertale. URL: https://www.sounds-resource.com/pc_computer/undertale/sound/6275/ (дата звернення: 17.12.23)
12. Текстовий генератор Text Studio. URL: https://www.textstudio.com (дата звернення: 18.12.23)
13. Онлайн бібліотека шрифтів Fontspace - Diary of an 8-bit mage. URL: https://www.fontspace.com/diary-of-an-8-bit-mage-font-f28455 (дата звернення: 07.04.24)
14. Конвертор файлів Convertio. URL: https://convertio.co (дата звернення: 15.12.23)
15. Аудіоредактор MP3Cut. URL: https://mp3cut.net (дата звернення: 26.03.24)
16. GIF редактор Ezgif. URL: https://ezgif.com (дата звернення: 01.02.24)
17. Марейн Хавербеке. Красномовний джаваскрипт. No Starch Press - Четверте видання, 2024.
18. Грем Стюарт. Представляємо джаваскрипт розробку ігор: побудуй 2D гру з нуля. Apress - 2017.

# ДОДАТКИ

## Додаток А

## Повний код програми

### А1. HTML код вебсторінки

```
<!doctype html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>ARKANOID</title>
	<link rel="stylesheet" href="style.css">
	<link rel="icon" href="images/icon.png">
</head>
<body>
<canvas class="game" id="game" width="760" height="970"></canvas>
<script async src="script.js"></script>
</body>
</html>
```

# А2. CSS код

```
.game {
  display: block;
  margin: 0 auto;
  position: absolute;
  width: auto;
  height: auto;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;}

body {
  background: #000000;
  font-family: 'Roboto', sans-serif; }

@font-face {
  font-family: "EightBit";
  src: url('fonts/EightBit.ttf') format('truetype');
}
```

# А3. JS код програми

```
var canvas = document.getElementById('game');
var ctx = canvas.getContext('2d');
canvas.style.cursor = 'none';
var canvasWidth = 760;
var canvasHeight = 970;
var scaleX = (window.innerWidth) / canvasWidth;
var scaleY = (window.innerHeight) / canvasHeight;
var scale = Math.min(scaleX, scaleY);
canvas.style.transformOrigin = '0 0';
canvas.style.transform = 'scale(' + scale + ')';
canvas.width = canvasWidth; //  * scale
canvas.height = canvasHeight; // * scale
let gameframe = 0;
var Pause = false;
var DevConsoleSpeed = 0;
var KeyUnpressed = {};
class PLATFORM
{
 constructor(x, y)
 {
  this.x = x;
  this.y = y;
  this.w = 110;
  this.h = 30;
  this.frameX = 0;
  this.staggerFrame = 30;
  this.CanCollide = true;
  this.canMove = true;
  this.isLarge = false;
  this.isAnnihilator = false;
  this.LaserCount = 0;
  this.AnnihilatorTime = 0;
 }
 CollisionDisable()
 {
  this.CanCollide = false;
  setTimeout(() => {
   this.CanCollide = true;
  }, 200);
 }
 LargePlatform()
 {
  clearTimeout(this.enlargeTime);
  if (!this.isLarge) this.x -= 55;
  this.w = 220;
  this.isLarge = true;
  this.enlargeTime = setTimeout(() =>
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* 1 | |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |



*Циклова комісія Комп'ютерної інженерії*

```
      {
        if (this.isLarge) this.x += 55;
        this.w = 110;
        this.isLarge = false;
      }, 20000
    );
  }
}
class BALL
{
  constructor(x, y)
  {
    this.x = x;
    this.y = y;
    this.speedX = 0;
    this.speedY = 0;
    this.BaseSpeed = 10;
    this.size = 18;
    this.AccelerationValue = 1;
    this.SlowValue = 1;
    this.frameX = 0;
    this.CanCollide = true;
    this.isPowerBall = false;
    this.isAccelerated = false;
    this.isSlow = false;
    this.isCharged = false;
    this.trail = [];
  }
  ChangeDirection(Coordinate)
  {
    this.CanCollide = false;
    this['speed' + Coordinate] = -this['speed' + Coordinate];
    setTimeout(() =>
      {
      this.CanCollide = true;
      }, 30
    );
  }
  PowerBall()
  {
    clearTimeout(this.powerballTime);
    this.isPowerBall = true;
    this.size = 30;
    this.x -= 6;
    this.y -= 6;
    this.powerballTime = setTimeout(() =>
      {
      this.isPowerBall = false;
      this.size = 18;
      this.x += 6;
      this.y += 6;
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* 1 | |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |



*Циклова комісія Комп'ютерної інженерії*

```
      }, 20000
    );
  }
  Acceleration()
  {
    clearTimeout(this.accelerationTime);
    if (this.isAccelerated) this.AccelerationValue = 1;
    this.isAccelerated = true;
    this.AccelerationValue = 1.5;
    this.accelerationTime = setTimeout(() =>
      {
      this.AccelerationValue = 1;
      this.isAccelerated = false;
      this.trail = [];
      }, 10000
    );
  }
  SlowDown()
  {
    clearTimeout(this.slowDownTime);
    if (this.isSlow) this.SlowValue = 1;
    this.isSlow = true;
    this.SlowValue = 0.5;
    this.slowDownTime = setTimeout(() =>
      {
      this.SlowValue = 1;
      this.isSlow = false;
      }, 100
    );
  }
  Start()
  {
    this.speedX = 0;
    this.speedY = -this.BaseSpeed;
  }
}
class BRICK
{
  constructor(x, y, type)
  {
    this.x = x;
    this.y = y;
    this.w = 54;
    this.h = 25;
    this.type = type;
  }
}
class EXPLOSION
{
  constructor(x, y)
  {
```

| Змн. | Арк. | № докум. | Підпис | Дат | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
|---|---|---|---|---|---|---|---|---|
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів 45 | 1 |
| Т. Контр. | | | | | | Циклова комісія Комп'ютерної інженерії | | |
| Реценз. | | | | | | | | |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
    this.x = x;
    this.y = y;
    this.w = 250;
    this.h = 250;
    this.frameX = 0;
   }
}
class BOMBEXPLOSION
{
  constructor(x, y)
  {
    this.x = x;
    this.y = y;
    this.w = 100;
    this.h = 100;
    this.frameX = 0;
  }
}
class LASER
{
  constructor(x, y)
  {
    this.x = x;
    this.y = y;
    this.w = 8;
    this.h = 12;
  }
  Fly()
  {
    this.y -= 15;
  }
}
class ANNIHILATORRAY
{
  constructor(x, y)
  {
    this.x = x;
    this.y = y;
    this.w = 145;
    this.h = 890;
    this.frameX = 0;
  }
}
class BONUS {
  constructor(x, y, w = 50, h = 23) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.frameX = 0;
  }
```

```
  Fall() {this.y += Math.floor(Math.random() * 3 + 2)}
}
class ENLARGEBONUS extends BONUS {}
class POWERBALLBONUS extends BONUS {}
class SPEEDBONUS extends BONUS {
  Fall() {this.y += Math.floor(Math.random() * 4 + 4)}
}
class BOOMBONUS extends BONUS {}
class LASERBONUS extends BONUS {}
class ANNIHILATEBONUS extends BONUS {}
class BONUSBALL extends BONUS {}
class LIFE extends BONUS {}
class BOMB extends BONUS {
  constructor(x, y) {super(x, y, 40, 48)}
  Fall() {this.y += Math.floor(Math.random() * 6 + 3)}
}
class ENEMY
{
  constructor(x, y)
  {
    this.x = x;
    this.y = y;
    this.w = 50;
    this.h = 23;
  }
  Texture(ctx, image)
  {
    ctx.drawImage(image, 0, 0, this.w, this.h, this.x, this.y, this.w, this.h);
  }
}
class GAME
{
  constructor ()
  {
    this.images = [];
    this.score = localStorage.getItem('score');
    if (this.score == null) {this.score == 0}
    else this.score = parseInt(this.score);
    this.bestscore = localStorage.getItem('bestscore');
    if (this.bestscore == null) {this.bestscore == 0}
    else this.bestscore = parseInt(this.bestscore);
    this.lifecount = localStorage.getItem('lifecount');
    if (this.lifecount == null) {this.lifecount == 3}
    else this.lifecount = parseInt(this.lifecount);
    this.level = localStorage.getItem('level');
    if (this.level == null) {this.level == 1}
    else this.level = parseInt(this.level);
    //this.level = 1;
    //localStorage.setItem('level', this.level);
    this.NextAttempt = false;
    this.InputDeviceControl();
```

| Змн. | Арк. | № докум. | Підпис | Дат | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
|---|---|---|---|---|---|---|---|---|
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів 1 | |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | Циклова комісія Комп'ютерної інженерії | | |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
  this.GameStart();
 }
 GameStart()
    {
     this.LoadImages(this.images, () =>
      {
       gameframe = 0;
       this.ProgressSpeed = 1;
       this.BricksCount = 0;
       this.NextLevelTimer = 0;
       this.textFade = 1;
       this.balls = [];
       this.enlargebonuses = [];
       this.powerballbonuses = [];
       this.speedbonuses = [];
       this.boombonuses = [];
       this.laserbonuses = [];
       this.annihilatebonuses = [];
       this.bonusballs = [];
       this.lives = [];
       this.enemies = [];
       this.explosions = [];
       this.bombexplosions = [];
       this.lasers = [];
       this.bombs = [];
            this.bonusClasses = [ENLARGEBONUS, POWERBALLBONUS, SPEEDBONUS, BOOMBONUS, LASERBONUS, ANNIHILATEBONUS, BONUSBALL, LIFE, BOMB, ENEMY];
          this.bonusArrays = [this.enlargebonuses, this.powerballbonuses, this.speedbonuses, this.boombonuses, this.laserbonuses, this.annihilatebonuses, this.bonusballs, this.lives, this.bombs, this.enemies];
       this.LaserCount = 0;
       this.platform = new PLATFORM(0, 0);
       this.platform.x = canvas.width / 2 - this.platform.w / 2;
       this.platform.y = canvas.height - 70;
       this.ball = new BALL(0, 0);
       this.Start = false;
       this.Pause = false;
       this.BuildBricks();
       this.SetBG();
       this.Loop();
       this.AlignBall();
       this.PlaySound('sounds/GameStartSound.mp3');
      }
     );
    }
    BuildBricks() {
     this.bricks = [];
     switch (this.level)
     {
      case 1:
       for (let i = 0; i < 4; i++) {
        let y = i * 2 * 25 + i + 130;
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
| Змн. | Арк. | № докум. | Підпис | Дат | | | | |
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів | 1 |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | 48 | | |
| Н. Контр. | | | | | | Циклова комісія Комп'ютерної інженерії | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
            for (let j = 0; j < 13; j++)
              {
                let x = j * 54 + 30;
                this.bricks.push(new BRICK(x, y, i, this.w, this.h));
              }
            }
            break;
          case 2:
            for (let i = 0; i < 12; i++)
              {
                for (let j = 0; j <= i; j++)
                  {
                    let x = j * 54 + 30;
                    this.bricks.push(new BRICK(x, 400, 4, this.w, this.h));
                  }
              }
              this.bricks.push(new BRICK(12 * 54 + 30, 400, 1, this.w, this.h));
              for (let i = 0; i < 12; i++)
                {
                  for (let j = 0; j <= i; j++)
                    {
                      let x = j * 54 + 30;
                      let y = (12 - i + j - 1) * 25 + 100;
                      let type = (j % 2 === 0) ? 1 : 2;
                      this.bricks.push(new BRICK(x, y, type, this.w, this.h));
                    }
                }
          case 3:
            break;
        }
        this.BricksCount = this.bricks.length;
      }
      SetBG()
      {
        if (this.level == 1 || this.level == 5) {game.style.backgroundImage = "url('images/bg1.png')";}
        else if (this.level == 2 || this.level == 6) {game.style.backgroundImage = "url('images/bg2.png')";}
        else if (this.level == 3 || this.level == 7) {game.style.backgroundImage = "url('images/bg3.png')";}
        else if (this.level == 4 || this.level == 8) {game.style.backgroundImage = "url('images/bg4.png')";}
        else if (this.level == 9) {game.style.backgroundImage = "url('images/bg5.png')";}
      }
      GamePause()
      {
        Pause =! Pause;
        if (Pause)
          {
            const pauseScreen = this.images['pausescreen'];
            ctx.drawImage(pauseScreen, ((canvas.width - pauseScreen.width) / 2), ((canvas.height - pauseScreen.height)
/ 2 + 10));
          }
        else
        {
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* 1 | |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |



*Циклова комісія Комп'ютерної інженерії*

```
            this.Loop();
          }
        }
        Loop()
        {
          gameframe++;
          if (gameframe >= 10000) gameframe = 500;
          if (Pause == true) return;
          ctx.clearRect(0, 0, canvas.width, canvas.height);
          if (!this.Start && !this.NextAttempt)
            {
              const startScreen = this.images['start'];
              ctx.drawImage(startScreen, ((canvas.width - startScreen.width) / 2), ((canvas.height - startScreen.height) / 2 +
150));
            }
          if (this.ball.x + this.ball.size > canvas.width - 30 || this.ball.x < 30)
            {
              this.ball.ChangeDirection('X');
              this.PlaySound('sounds/BallHitBrickHorizontally.mp3');
              if (this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3');
            }
          if (this.ball.x < 20)
            {
              this.ball.x = 30;
              Math.abs(this.calculatedSpeedX);
            }
          if (this.ball.x + this.ball.size > canvas.width - 20)
            {
              this.ball.x = canvas.width - this.ball.size - 30;
              -Math.abs(this.calculatedSpeedX);
            }
          if (this.ball.y < 100)
            {
              this.ball.ChangeDirection('Y');
              this.PlaySound('sounds/BallHitBrickVertically.mp3');
              if (this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3');
            }
          if (this.ball.y < 75)
            {
              this.ball.y = 100;
              Math.abs(this.calculatedSpeedY);
            }
          if (this.ball.x + this.ball.size >= this.platform.x && this.ball.x <= this.platform.x + this.platform.w &&
              this.ball.y + this.ball.size > this.platform.y && this.ball.y < this.platform.y + this.platform.h &&
              this.platform.CanCollide && this.Start)
              {
                    this.ball.speedX = this.ball.BaseSpeed * (this.ball.x - (this.platform.x + this.platform.w / 2)) * 2 /
this.platform.w;
                if (this.ball.speedX < 3 && this.ball.speedX > 0) this.ball.speedX = 3;
                else if (this.ball.speedX > -3 && this.ball.speedX < 0) this.ball.speedX = -3;
                this.ball.ChangeDirection('Y');
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* 1 | |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |



*Циклова комісія Комп'ютерної інженерії*

```
            this.platform.CollisionDisable();
            this.PlaySound('sounds/BallHitPlatform.mp3');
            if (this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3');
           }
         this.calculatedSpeedX = ((this.ball.speedX) + this.ball.speedX * DevConsoleSpeed) * this.ball.SlowValue *
this.ProgressSpeed;
         this.calculatedSpeedY = ((this.ball.speedY) + this.ball.speedY * DevConsoleSpeed) * this.ball.SlowValue *
this.ProgressSpeed * this.ball.AccelerationValue;
        this.ball.x += this.calculatedSpeedX;
        this.ball.y += this.calculatedSpeedY;
        if (this.ball.isAccelerated) this.ball.trail.push({ x: this.ball.x, y: this.ball.y });
        if (this.ball.trail.length > 10) {this.ball.trail.shift();}
        this.bricks.forEach((brick, idxBrick) =>
         {
          if (brick.type >= 0)
           {
            if (this.ball.x + this.ball.size > brick.x + 1 && this.ball.x < brick.x + brick.w - 1 &&
             this.ball.y + this.ball.size > brick.y && this.ball.y < brick.y + brick.h && this.ball.CanCollide)
             {
              this.ball.ChangeDirection('Y');
              this.BrickCollision(idxBrick, 'ball');
              this.PlaySound('sounds/BallHitBrickHorizontaly.mp3');
              if (this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3');
              this.ball.SlowDown();
             }
            if (this.ball.x + this.ball.size >= brick.x - 1 && this.ball.x <= brick.x + brick.w + 1 &&
             this.ball.y + this.ball.size >= brick.y && this.ball.y <= brick.y + brick.h && this.ball.CanCollide)
             {
              this.ball.ChangeDirection('X');
              this.BrickCollision(idxBrick, 'ball');
              this.PlaySound('sounds/BallHitBrickVertically.mp3');
              if (this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3');
              this.ball.SlowDown();
             }
           }
          ctx.drawImage(this.images['bricks'], 0, brick.type * brick.h, brick.w, brick.h, brick.x, brick.y, brick.w, brick.h);
           if (brick.type < 0)
            {
             this.bricks.splice(idxBrick, 1);
             this.PlaySound('sounds/BrickBreak.mp3');
            }
         }
        );
        this.explosions.forEach((explosion, explosionIdx) =>
         {
              ctx.drawImage(this.images['explosion'], explosion.frameX * explosion.w, 0, explosion.w, explosion.h,
explosion.x, explosion.y, explosion.w, explosion.h);
          if (gameframe % 8 == 0)
           {
            explosion.frameX++;
            if (explosion.frameX >= 7) delete this.explosions[explosionIdx];
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* | 1 |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | *Циклова комісія Комп'ютерної інженерії* | | 51 |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |

```
        }
      }
    );
    this.bombexplosions.forEach((bombexplosion, bombexplosionIdx) =>
      {
            ctx.drawImage(this.images['bombexplosion'], bombexplosion.frameX * bombexplosion.w, 0,
bombexplosion.w, bombexplosion.h, bombexplosion.x, bombexplosion.y, bombexplosion.w, bombexplosion.h);
        if (gameframe % 8 == 0)
          {
            bombexplosion.frameX++;
            if (bombexplosion.frameX >= 9) delete this.bombexplosions[bombexplosionIdx];
          }
      }
    );
    const bonuses = [
      {
        array: this.enlargebonuses,
        image: this.images['enlargebonus'],
        onPickUp: () =>
          {
            this.platform.LargePlatform();
            this.LaserCount = 0;
            if (this.platform.isAnnihilator)
              {
                this.platform.isAnnihilator = false;
                this.platform.frameX = 0
              }
          },
        condition: () => !this.platform.AnnihilatorTime,
        sound: 'sounds/EnlargeBonusPickUp.mp3'
      },
      {
        array: this.powerballbonuses,
        image: this.images['powerballbonus'],
        onPickUp: () => this.ball.PowerBall(),
        condition: () => true, sound: 'sounds/PowerBallBonusPickUp.mp3'
      },
      {
        array: this.speedbonuses,
        image: this.images['speedbonus'],
        onPickUp: () => this.ball.Acceleration(),
        condition: () => true, sound: 'sounds/SpeedBonusPickUp.mp3'
      },
      {
        array: this.boombonuses,
        image: this.images['boombonus'],
        onPickUp: () => this.ball.isCharged = true,
        condition: () => true, sound: 'sounds/BoomBonusPickUp.mp3'
      },
      {
```

```
  array: this.laserbonuses,
  image: this.images['laserbonus'],
  onPickUp: () =>
   {
    this.LaserCount = 20;
    if (this.platform.isLarge)
     {
      clearTimeout(this.platform.enlargeTime);
      this.platform.x += 55;
      this.platform.w = 110;
      this.platform.isLarge = false
     }
    if (this.platform.isAnnihilator)
     {
      this.platform.isAnnihilator = false;
      this.platform.frameX = 0
     }
   },
     condition: () => !this.platform.AnnihilatorTime,
     sound: 'sounds/LaserBonusPickUp.mp3'
},
{
  array: this.annihilatebonuses,
  image: this.images['annihilatebonus'],
  onPickUp: () =>
   {
    this.platform.isAnnihilator = true;
    this.LaserCount = 0;
    if (this.platform.isLarge)
     {
      clearTimeout(this.platform.enlargeTime);
      this.platform.x += 55;
      this.platform.w = 110;
      this.platform.isLarge = false
     }
   },
  condition: () => true,
  sound: 'sounds/AnnihilatorBonusPickUp.mp3'
},
{
  array: this.bonusballs,
  image: this.images['bonusball'],
  onPickUp: () => true,
  condition: () => true,
  sound: 'sounds/placeholder'
},
{
  array: this.lives,
  image: this.images['life'],
  onPickUp: () => {this.lifecount += 1, localStorage.setItem('lifecount', this.lifecount)},
  condition: () => this.lifecount < 6,
```

| Змн. | Арк. | № докум. | Підпис | Дат | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
|---|---|---|---|---|---|---|---|---|
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів 1 | |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | Циклова комісія Комп'ютерної інженерії | | |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
    sound: 'sounds/LifeUp.mp3'
   },
   {
    array: this.bombs,
    image: this.images['bomb'],
    onPickUp: (bonus) =>
     {
      this.lifecount -= 1;
      localStorage.setItem('lifecount', this.lifecount);
      const bombexplosion = new BOMBEXPLOSION(bonus.x - bonus.w / 2 - 10, bonus.y - bonus.h / 2 - 10);
      this.bombexplosions.push(bombexplosion)
     },
     condition: () => true,
     sound: 'sounds/BombExplosion.mp3'
   }
  ];
  bonuses.forEach(({ array, image, onPickUp, condition, sound }) =>
   {
    array.forEach((bonus, idx) =>
     {
      ctx.drawImage(image, bonus.frameX * bonus.w, 0, bonus.w, bonus.h, bonus.x, bonus.y, bonus.w, bonus.h);
      if (gameframe % 6 == 0)
       {
        bonus.frameX++;
        if (bonus.frameX >= 14) bonus.frameX = 0;
       }
      bonus.Fall();
      if
      (
       bonus.y + bonus.h >= this.platform.y &&
       bonus.y <= this.platform.y + this.platform.h &&
       bonus.x + bonus.w >= this.platform.x &&
       bonus.x <= this.platform.x + this.platform.w &&
       condition()
      )
      {
       onPickUp(bonus);
       delete array[idx];
       this.PlaySound(sound);
      }
      if (bonus.y >= 1000)
       {
        delete array[idx];
       }
     }
    );
   }
  );
  this.lasers.forEach((laser, laserIdx) =>
   {
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
| Змн. | Арк. | № докум. | Підпис | Дат | | | | |
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів | 1 |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | Циклова комісія Комп'ютерної інженерії | | 54 |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
    ctx.drawImage(this.images['laser'], 0, 0, laser.w, laser.h, laser.x, laser.y, laser.w, laser.h);
    laser.Fly();
    let laserCollided = false;
    this.bricks.forEach((brick, idxBrick) =>
     {
      if (laserCollided) return;
      if (laser.x + laser.w > brick.x && laser.x < brick.x + brick.w &&
       laser.y + laser.h >= brick.y && laser.y <= brick.y + brick.h)
       {
        this.BrickCollision(idxBrick, 'laser');
        this.PlaySound('sounds/LaserImpact.mp3');
        delete this.lasers[laserIdx];
        laserCollided = true;
       }
     }
    );
   if (laser.y <= 90) delete this.lasers[laserIdx];
   }
);
if (this.ball.y >= 1030)
 {
   this.ball.y = 1000;
   this.ball.speedY = 0;
   this.ball.speedX = 0;
   this.NextAttempt = true;
   this.platform.canMove = false;
   this.ball.trail = [];
   this.PlaySound('sounds/LifeLost.mp3');
   setTimeout(() =>
    {
     {
      clearTimeout(this.powerballTime);
   clearTimeout(this.platform.enlargeTime);
   clearTimeout(this.accelerationTime);
   clearTimeout(this.slowDownTime);
   this.ball.AccelerationValue = 1;
   this.ball.isCharged = false;
   this.ball.isAccelerated = false;
   if (this.ball.isPowerBall)
    {
     this.ball.x += 5;
     this.ball.y += 5;
     this.ball.isPowerBall = false;
     this.ball.size = 18;
    }
   if (this.platform.isLarge)
    {
     this.platform.x += 55;
     this.platform.w = 110;
     this.platform.isLarge = false;
    }
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* | 55 1 |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |

*Циклова комісія Комп'ютерної інженерії*

```
            if (this.platform.isAnnihilator)
              {
                this.platform.isAnnihilator = false;
                this.platform.AnnihilatorTime = 0;
                this.platform.frameX = 0;
              }
            this.LaserCount = 0;
                 this.lifecount -= 1;
                 localStorage.setItem('lifecount', this.lifecount);
                 this.AlignBall();
                 this.ball.frameX = 1;
                 if (this.lifecount) this.PlaySound('sounds/BallAppear.mp3');
                 setTimeout(() =>
                   {
                     {
                       this.platform.canMove = true;
                       this.Start = false;
                     }
                   }, 1200
                 )
               }
             }, 1500
           );
         }
         if (this.lifecount <= 0)
           {
             setTimeout(() =>
               {
                 {
                   this.GameStart();
                 }
               }, 3000);
               this.PlaySound('sounds/GameOver.mp3');
               ctx.clearRect(0, 0, canvas.width, canvas.height);
               this.score = 0;
               localStorage.setItem('score', this.score);
               this.lifecount = 3;
               localStorage.setItem('lifecount', this.lifecount);
               this.level = 1;
               localStorage.setItem('level', this.level);
               this.Start = true;
               const GameOverScreen = this.images['gameover'];
                    ctx.drawImage(GameOverScreen, ((canvas.width - GameOverScreen.width) / 2), ((canvas.height -
GameOverScreen.height) / 1.3));
               return;
           }
           ctx.font = "30px EightBit";
           ctx.fillStyle = 'white';
           ctx.textAlign = 'left';
           ctx.fillText(this.score, canvas.width / 2 - 300, canvas.height - 910);
           ctx.fillStyle = 'red';
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* | 56 1 |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |

*Циклова комісія Комп'ютерної інженерії*

```
ctx.fillText("Рекорд: ", canvas.width / 2 - 60, canvas.height - 910);
ctx.fillStyle = 'white';
ctx.fillText(this.bestscore, canvas.width / 2 + 80, canvas.height - 910);
if (gameframe <= 200) {
  ctx.font = "90px EightBit";
  ctx.fillStyle = 'rgba(255, 255, 255, ' + this.textFade + ')';
  ctx.textAlign = 'center';
  ctx.fillText('Рівень ' + this.level, canvas.width / 2, canvas.height / 2);
}
if (gameframe <= 200 && gameframe > 120) this.textFade -= 0.015;
//ctx.fillStyle = '#aaddff';
//ctx.textAlign = 'right';
//ctx.fillText(DevConsoleSpeed * 10 + 10, canvas.width - 30, canvas.height - 10);
if (this.score > this.bestscore)
  {
    this.bestscore = this.score;
    localStorage.setItem('bestscore', this.bestscore);
  }
if (this.platform.AnnihilatorTime >= 1)
  {
    if (this.platform.AnnihilatorTime < 140)
      {
        if (this.platform.AnnihilatorTime < 110)
          {
            this.platform.x += (Math.random() - 0.5) * 5
            this.platform.y += (Math.random() - 0.5) * 2
          }
        if (this.platform.AnnihilatorTime == 111)
          {
            this.platform.y = canvas.height - 70;
            this.annihilatorray = new ANNIHILATORRAY(this.platform.x - 17, 100);
            this.bricks.forEach((brick, idxBrick) =>
              {
               if (this.annihilatorray.x + this.annihilatorray.w > brick.x && this.annihilatorray.x < brick.x + brick.w)
                  {
                    this.BrickCollision(idxBrick, 'annihilatorray');
                  }
              }
            );
          }
        if (this.platform.AnnihilatorTime >= 111)
          {
                ctx.drawImage(this.images['annihilatorray'], this.annihilatorray.frameX * this.annihilatorray.w, 0,
this.annihilatorray.w, this.annihilatorray.h, this.annihilatorray.x, this.annihilatorray.y, this.annihilatorray.w,
this.annihilatorray.h);
            if (gameframe % 2 == 0)
              {
                this.annihilatorray.frameX++;
                if (this.annihilatorray.frameX >= 15) this.annihilatorray.frameX = 0;
              }
```

| Змн. | Арк. | № докум. | Підпис | Дат | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
|---|---|---|---|---|---|---|---|---|
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів | 57 1 |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | Циклова комісія Комп'ютерної інженерії | | |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
                }
                this.platform.AnnihilatorTime++;
                this.ball.speedY -= this.ball.speedY / 10;
                this.ball.speedX -= this.ball.speedX / 10;
              }
             else {
              this.platform.frameX = 0;
              this.platform.AnnihilatorTime = 0;
              this.platform.isAnnihilator = false;
              if (this.saveSpeedY <=0) this.ball.speedY = -this.ball.BaseSpeed;
              else this.ball.speedY = this.ball.BaseSpeed;
              this.ball.speedX = this.saveSpeedX;
             }
            }
            const platform = (ctx, this.images[this.platform.isAnnihilator ? 'annihilateplatform' : this.platform.isLarge ? 'largeplatform' : this.LaserCount ? 'laserplatform' : 'platform']);
             ctx.drawImage(platform, this.platform.frameX * this.platform.w, 0, this.platform.w, this.platform.h + 20, this.platform.x, (this.platform.isAnnihilator ? this.platform.y-20 : this.platform.y), this.platform.w, this.platform.h + 20);
            if (this.Start && gameframe % this.platform.staggerFrame == 0)
             {
              this.platform.frameX++;
              if (!this.platform.isAnnihilator && this.platform.frameX >= 2) this.platform.frameX = 0;
              else if (this.platform.isAnnihilator && this.platform.frameX >= 6) this.platform.frameX = 0;
             }
             this.ball.trail.forEach((position, index) => {
                  const ballImage = this.images[this.ball.isCharged ? (this.ball.isPowerBall ? 'chargedpowerball' : 'chargedball') : (this.ball.isPowerBall ? 'powerball' : 'ball')];
              const opacity = (index + 1) / this.ball.trail.length;
              ctx.globalAlpha = opacity;
                   ctx.drawImage(ballImage, this.ball.frameX * (this.ball.isPowerBall ? 30 : 24), 0, this.ball.size + this.ballSizeIncr, this.ball.size + this.ballSizeIncr, position.x - this.ballSizePos, position.y - this.ballSizePos, this.ball.size + this.ballSizeIncr2, this.ball.size + this.ballSizeIncr2);
             });
             ctx.globalAlpha = 1;
           const ball = (ctx, this.images[this.ball.isCharged ? (this.ball.isPowerBall ? 'chargedpowerball' : 'chargedball') : (this.ball.isPowerBall ? 'powerball' : 'ball')]);
            this.ballSizeIncr = this.ball.isPowerBall ? -6 : 6;
            this.ballSizeIncr2 = this.ball.isPowerBall ? 0 : 6;
            this.ballSizePos = this.ball.isPowerBall ? 0 : 3;
             ctx.drawImage(ball, this.ball.frameX * (this.ball.isPowerBall ? 30 : 24), 0, this.ball.size + this.ballSizeIncr, this.ball.size + this.ballSizeIncr, this.ball.x - this.ballSizePos, this.ball.y - this.ballSizePos, this.ball.size + this.ballSizeIncr2, this.ball.size + this.ballSizeIncr2);
            ctx.drawImage(this.images['lifebar'], 0, 0, this.lifecount * 35, 13, 49, canvas.height - 25, this.lifecount * 35, 13);
            if (this.ball.frameX >= 1)
             {
              if (gameframe % 4 == 0)
               {
                this.ball.frameX++;
                if (this.ball.frameX >= 13) this.ball.frameX = 0;
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* 1 | |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | *Циклова комісія Комп'ютерної інженерії* | 58 | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |

```
            }
          }
        if (this.bricks.filter(brick => brick.type !== 4).length === 0 && !this.NextLevelTimer) {
          this.PlaySound('sounds/LevelComplete.mp3');
          this.NextAttempt = true;
          this.NextLevelTimer = 1;
          this.lvlcmpFrameY = 0
          this.opacity = 0;
        }
        if (this.NextLevelTimer >= 1 && this.NextLevelTimer < 200)
          {
            this.ball.speedY = 0;
            this.ball.speedX = 0;
            this.NextLevelTimer++
            if (this.NextLevelTimer < 150)
              {
                const levelCompletedScreen = this.images['levelcompleted'];
                  ctx.drawImage(levelCompletedScreen, 0, this.lvlcmpFrameY * 41, levelCompletedScreen.width, 41,
(canvas.width - levelCompletedScreen.width) / 2, (canvas.height - 50) / 2, levelCompletedScreen.width, 50);
                if (this.lvlcmpFrameY < 6)
                  {
                    if (gameframe % 7 == 0)
                      {
                        this.lvlcmpFrameY++;
                      }
                  }
              }
          }
          if (this.NextLevelTimer > 50 && this.NextLevelTimer < 150)
            {
              ctx.fillStyle = `rgba(0, 0, 0, ${this.opacity})`;
              ctx.fillRect(0, 0, canvas.width, canvas.height);
              this.opacity += 0.02;
            }
            if (this.NextLevelTimer == 150)
              {
                if (this.level < 9)
                  {
                    setTimeout(() =>
                      {
                        {
                          this.GameStart();
                        }
                      },1);
                      this.level += 1;
                      localStorage.setItem('level', this.level);
                      ctx.clearRect(0, 0, canvas.width, canvas.height);
                      return;
                  }
                else
                 {
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* | 1 |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |



*Циклова комісія Комп'ютерної інженерії*

```
                    setTimeout(() =>
                      {
                        {
                          this.GameStart();
                        }
                      },4000);
                      ctx.clearRect(0, 0, canvas.width, canvas.height);
                      this.score = 0;
                      localStorage.setItem('score', this.score);
                      this.lifecount = 3;
                      localStorage.setItem('lifecount', this.lifecount);
                      this.level = 1;
                      localStorage.setItem('level', this.level);
                      this.PlaySound('sounds/WinSound.mp3');
                            ctx.drawImage(this.images['winscreen'], ((canvas.width - this.images['winscreen'].width) / 2),
((canvas.height - this.images['winscreen'].height) / 1.3));
                      return;
                    }
                  }
                  window.requestAnimationFrame(() => {this.Loop();});
              }
              AlignBall()
              {
                this.ball.x = this.platform.x + this.platform.w / 2 - this.ball.size / 2;
                this.ball.y = this.platform.y - this.ball.size;
              }
              BrickCollision(idxBrick, source)
              {
                const brick = this.bricks[idxBrick];
                if (source === 'ball' && this.ball.isCharged)
                  {
                    this.ball.isCharged = false;
                    const explosion = new EXPLOSION((this.bricks[idxBrick].x -98), (this.bricks[idxBrick].y -113));
                    this.explosions.push(explosion);
                    this.PlaySound('sounds/Explosion.mp3');
                    this.bricks.forEach((brick, idxBrick) =>
                      {
                        if (
                          explosion.x + explosion.w > brick.x &&
                          explosion.x < brick.x + brick.w &&
                          explosion.y + explosion.h >= brick.y &&
                          explosion.y <= brick.y + brick.h
                        )
                        {
                          setTimeout(() => {
                            this.BrickCollision(idxBrick, 'explosion');
                          });
                        }
                      }
                    );
                  }
```

| Змн. | Арк. | № докум. | Підпис | Дат | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
|---|---|---|---|---|---|---|---|---|
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів 1 | |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | Циклова комісія Комп'ютерної інженерії | | 60 |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |

```
            if (brick.type != 4 && ((source === 'ball' && !this.ball.isPowerBall) || source === 'laser' || source === 'explosion'))
            {
              this.bricks[idxBrick].type--;
              this.score++;
              localStorage.setItem('score', this.score);
            }
          else if (brick.type != 4 && ((source === 'ball' && this.ball.isPowerBall) || source === 'annihilatorray'))
            {
              this.score = this.score + this.bricks[idxBrick].type + 1;
              localStorage.setItem('score', this.score);
              this.bricks[idxBrick].type = -1;
              if (source === 'ball' && this.ball.isPowerBall) this.PlaySound('sounds/PowerBallHit.mp3');
            }
            var BricksLeft = Math.round((this.bricks.filter(brick => brick.type !== 4).length * 100) / (this.BricksCount - this.bricks.filter(brick => brick.type === 4).length));
          if (BricksLeft >= 17 && BricksLeft <= 34) this.ProgressSpeed = 1.6;
          else if (BricksLeft >= 35 && BricksLeft <= 50) this.ProgressSpeed = 1.48;
          else if (BricksLeft >= 51 && BricksLeft <= 67) this.ProgressSpeed = 1.36;
          else if (BricksLeft >= 68 && BricksLeft <= 83) this.ProgressSpeed = 1.24;
          else if (BricksLeft >= 84 && BricksLeft <= 100) this.ProgressSpeed = 1.12;
          if (brick.type != 4 && (Math.floor(Math.random() * 100) < 30 && !this.BonusCooldown))
            {
              this.BonusCooldown = true;
              setTimeout(() =>
                {
                  this.BonusCooldown = false;
                }, 4000);
                const BonusType = Math.floor(Math.random() * 9);
                const BonusClass = this.bonusClasses[BonusType];
                const bonusArray = this.bonusArrays[BonusType];
                const bonus = new BonusClass((brick.x + brick.w / 2), brick.y + brick.h);
                bonus.x -= bonus.w / 2;
                bonusArray.push(bonus);
            }
          }
        InputDeviceControl()
        {
          document.addEventListener('mousemove', (mouse) =>
            {
              if (this.platform.canMove)
                {
                  if (this.platform.isLarge)
                    {
                      this.platform.x = Math.max(30, Math.min(mouse.screenX - canvas.width / 2, 512));
                    }
                  else if (!this.platform.AnnihilatorTime)
                    {
                      this.platform.x = Math.max(30, Math.min(mouse.screenX - canvas.width / 2 + this.platform.w / 2, 622));
                    }
                }
```

```
    if (!this.Start) this.AlignBall();
  }
);
document.addEventListener('keydown', (Gamekeys) =>
  {
    if (Gamekeys.key === ' ' && KeyUnpressed)
      {
        if (!this.Start)
          {
            this.ball.Start();
            this.Start = true;
          }
        else if (this.Start && this.LaserCount)
          {
            KeyUnpressed = false;
            this.lasers.push(new LASER((this.platform.x + this.platform.w / 2 - 35), this.platform.y));
            this.lasers.push(new LASER((this.platform.x + this.platform.w / 2 + 35), this.platform.y));
            this.PlaySound('sounds/LaserSound.mp3');
            this.LaserCount--;
          }
        else if (this.Start && this.platform.isAnnihilator && !this.platform.AnnihilatorTime)
          {
            this.platform.AnnihilatorTime++
            this.PlaySound('sounds/Annihilator.mp3');
            this.saveSpeedY = this.ball.speedY
            this.saveSpeedX = this.ball.speedX
          }
      }
    else if (Gamekeys.key === 'Escape' && KeyUnpressed)
      {
        KeyUnpressed = false;
        this.GamePause();
        if (Pause) this.PlaySound('sounds/PauseSound.mp3');
        else this.PlaySound('sounds/UnpauseSound.mp3');
      }
  }
);
document.addEventListener('keydown', (DeveloperConsole) =>
  {
    const keyIndex = parseInt(DeveloperConsole.key, 10) - 1;
    if (keyIndex >= 0 && keyIndex < this.bonusClasses.length && KeyUnpressed) {
      KeyUnpressed = false;
      const BonusClass = this.bonusClasses[keyIndex];
      const bonusArray = this.bonusArrays[keyIndex];
      const bonus = new BonusClass((this.platform.x + this.platform.w / 2), 100);
      bonus.x -= bonus.w / 2;
      bonusArray.push(bonus);
    }
    else if (DeveloperConsole.key === '0' && KeyUnpressed)
      {
        KeyUnpressed = false;
```

```
            SpawnEnemy()
           }
          else if (DeveloperConsole.key === '+' || DeveloperConsole.key === '=')
           {
            DevConsoleSpeed = DevConsoleSpeed + 0.1;
            DevConsoleSpeed = Math.round(DevConsoleSpeed * 10) / 10
           }
          else if (DeveloperConsole.key === '-')
           {
            DevConsoleSpeed = Math.max(DevConsoleSpeed - 0.1, -1);
            DevConsoleSpeed = Math.round(DevConsoleSpeed * 10) / 10
           }
          else if (DeveloperConsole.key === 't')
           {
           }
         }
        );
        const UnpressKeys = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', ' ', 'Escape'];
        const Gamekeys = [' ', 'Escape'];
        document.addEventListener('keyup', (DeveloperConsole) =>
         {
          if (UnpressKeys.includes(DeveloperConsole.key) || UnpressKeys.includes(Gamekeys.key))
           {
            KeyUnpressed = true;
           }
         }
        );
       }
       LoadImages(images, finishloading)
       {
        const imageNames = [
                'start', 'gameover', 'winscreen', 'pausescreen', 'levelcompleted', 'ball', 'powerball', 'chargedball',
'chargedpowerball', 'platform', 'largeplatform',
          'laserplatform', 'annihilateplatform', 'lifebar', 'enlargebonus', 'powerballbonus', 'speedbonus', 'boombonus',
'laserbonus', 'annihilatebonus', 'bonusball',
         'life', 'enemy', 'explosion', 'laser', 'annihilatorray', 'bomb', 'bombexplosion', 'bricks'
        ];
        images = imageNames.map(name => ({ name, path: `images/${name}.png` }));
        const promises = [];
        Array.isArray(images) && images.forEach((image) =>
         {
          promises.push(new Promise((load) =>
           {
            var img = new Image();
            img.src = image.path;
            img.onload = () =>
             {
              this.images[image.name] = img;
              load();
             };
           }
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* 1 | |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | *Циклова комісія Комп'ютерної інженерії* 63 | | |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |

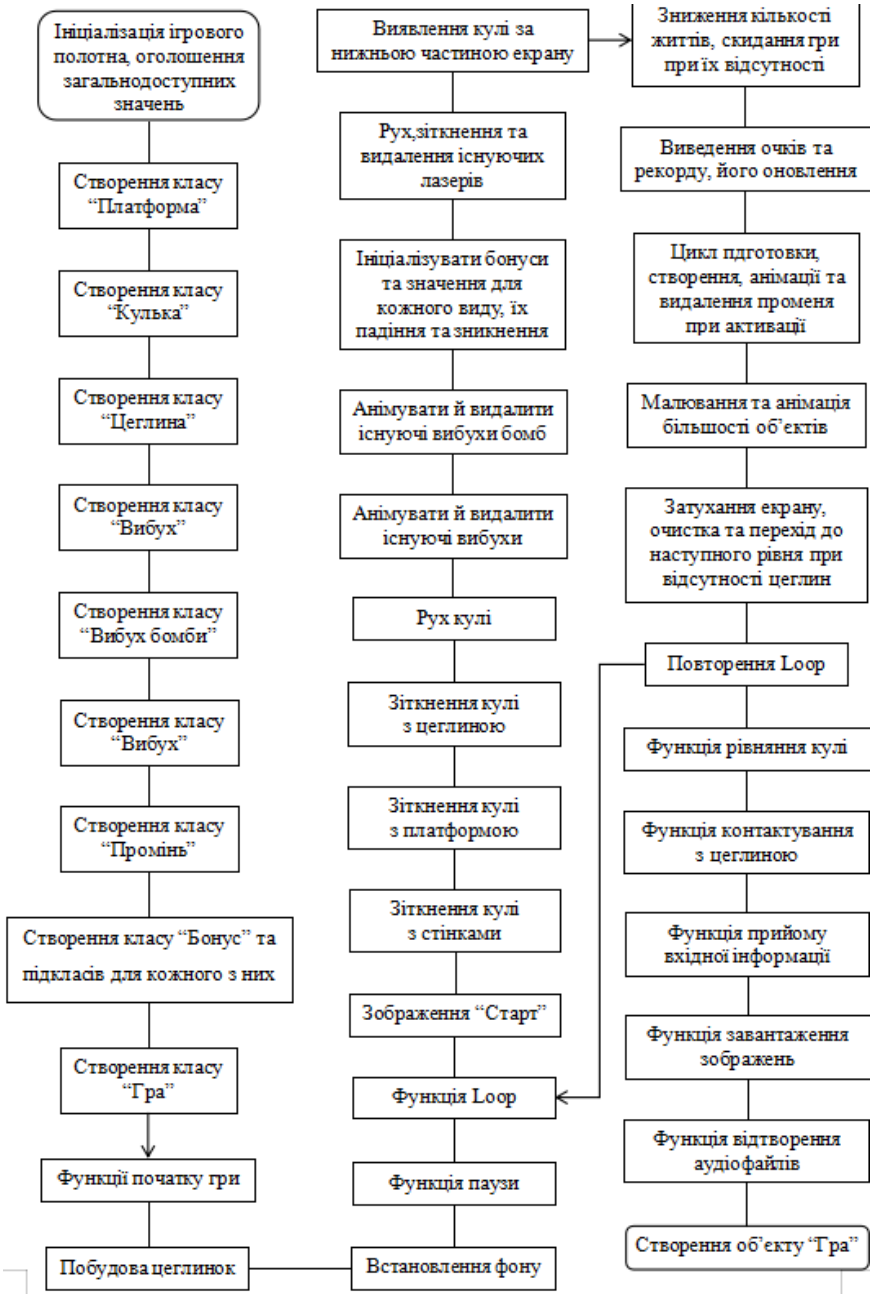
```
            ));
          }
        );
        Promise.all(promises).then(function()
        {
          finishloading();
        });
      }
      PlaySound(SoundLocation)
      {
        let audio = new Audio(SoundLocation);
        audio.play();
      }
}
new GAME();
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ* | *Літ.* | *Маса* | *Масштаб* |
| *Змн.* | *Арк.* | *№ докум.* | *Підпис* | *Дат* | | | | |
| *Розроб.* | | *Деркач М. А.* | | | | | | |
| *Перевір.* | | *Мельничук А.Ю.* | | | | | | |
| *Т. Контр.* | | | | | | *Арк.* | *Аркушів* | 1 |
| *Реценз.* | | | | | | | | |
| *Н. Контр.* | | | | | | *Циклова комісія Комп'ютерної інженерії* | | 64 |
| *Затверд.* | | *Тащук О.Ю.* | | | | | | |

**Додаток Б**

# Функціональні схеми

## Б1. Узагальнена функціональна схема проекту




| Змн. | Арк. | № докум. | Підпис | Дат |
|---|---|---|---|---|
| Розроб. | | Деркач М. А. | | |
| Перевір. | | Мельничук А.Ю. | | |
| Т. Контр. | | | | |
| Реценз. | | | | |
| Н. Контр. | | | | |
| Затверд. | | Тащук О.Ю. | | |

*Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ*

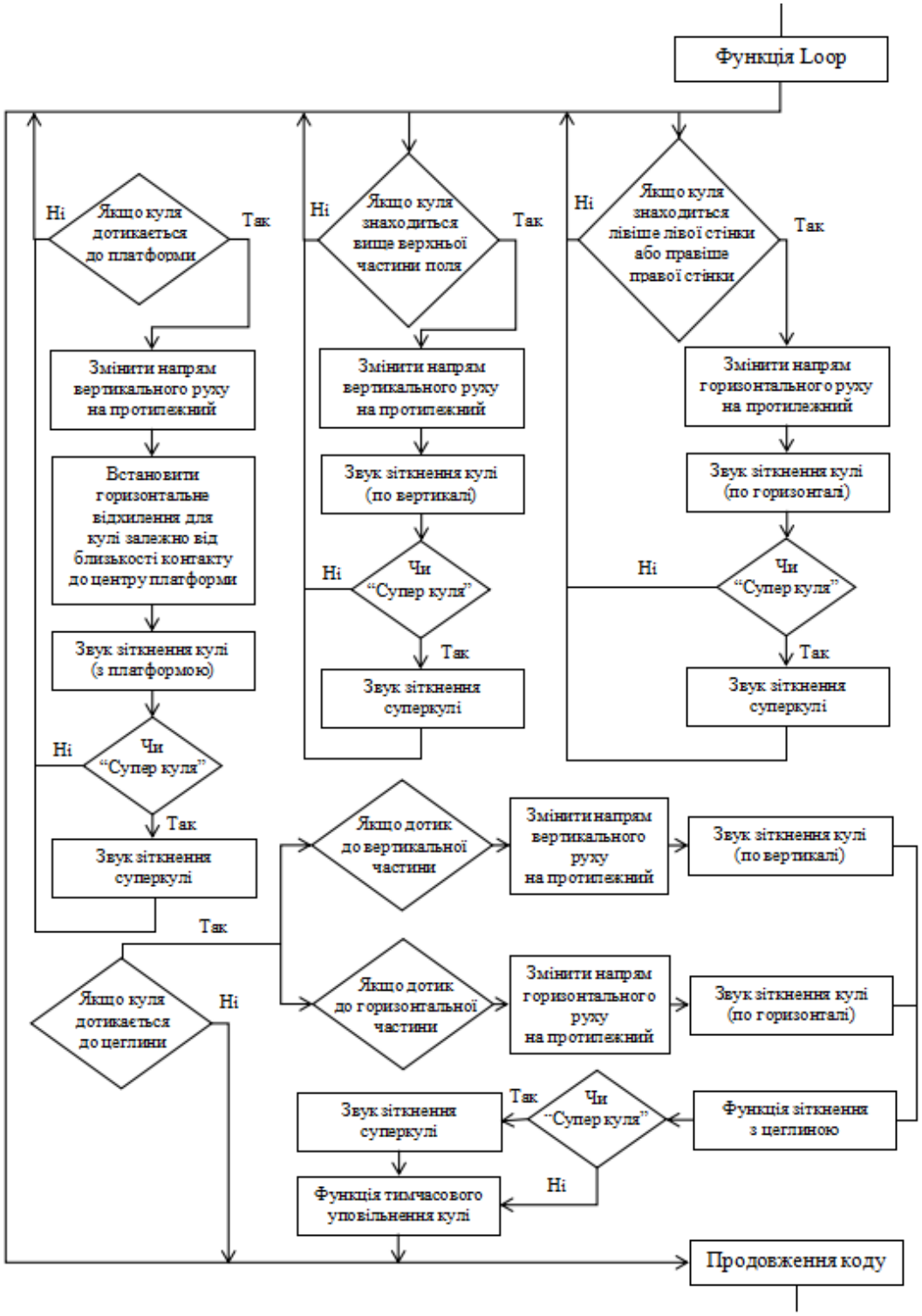
| Літ. | Маса | Масштаб |
|---|---|---|
| | | |

Арк. Аркушів 1



Циклова комісія Комп'ютерної інженерії

## Б2. Функціональна схема фізики кулі




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| Змн. | Арк. | № докум. | Підпис | Дат | Розробка версії гри «Арканоїд» з використанням сучасних технологій і платформ | Літ. | Маса | Масштаб |
| Розроб. | | Деркач М. А. | | | | | | |
| Перевір. | | Мельничук А.Ю. | | | | Арк. | Аркушів | 1 |
| Т. Контр. | | | | | | | | |
| Реценз. | | | | | | Циклова комісія Комп'ютерної інженерії | | 66 |
| Н. Контр. | | | | | | | | |
| Затверд. | | Тащук О.Ю. | | | | | | |