МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії програмного
забезпечення, канд.техн.наук, доцент,
_____Є.О. Давиденко
«____»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІГРОВИЙ ЗАСТОСУНОК В ЖАНРІ АРКАДА

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.01 − 408.21810816

Студент _____Г. Ю. Круковська
підпис
«___» _____ 2022 р.

Керівник канд. техн. наук, доцент кафедри ІПЗ _____А. В. Швед
підпис
«___» _____ 2022 р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
підпис
«___» _____ 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії програмного
забезпечення, канд.техн.наук, доцент,
_____Є.О. Давиденко
«____»_____2022 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук
Круковській Ганні Юріївні

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
«Ігровий застосунок в жанрі аркада»

Затверджена наказом по ЧНУ від «01» грудня 2022 р. № 314

2. Строк представлення кваліфікаційної роботи «_»_____2022 р.
3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Вхідні дані до роботи – ім'я гравця, налаштування гри. Результат – функціонуючий застосунок гра.

4. Перелік питань, що підлягають розробці
   - аналіз предметної області та існуючих аналогів;
   - розробка вимог до ігрового застосунку;

- моделювання та проектування застосунку;
- розробка дизайну ігрових сцен;
- програмна реалізація, тестування та відлагодження прототипів гри.

5. Перелік графічних матеріалів:

Слайди презентації

.

6. Завдання до спеціальної частини

Дослідження питань охорони праці, які безпосередньо пов'язані з діяльністю розробника програмного забезпечення .

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи доцент кафедри ІПЗ Швед Альона Володимирівна
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Круковська Ганна Юріївна
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: «Гра «Змійка» на мові програмування Python»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	08.11.2021	16.11.2021	Виконано
2.	Пошук літератури за темою роботи	17.11.2021	08.12.2021	Виконано
3.	Складання календарного плану КРБ	09.12.2021	10.12.2021	Виконано
4.	Аналіз предметної області	13.12.2021	27.12.2021	Виконано
5.	Створення основних проєктних рішень	28.12.2021	07.01.2022	Виконано
6.	Моделювання та конструювання ігрового застосунку	10.01.2022	21.01.2022	Виконано
7.	Кодування та тестування застосунку	24.01.2022	21.02.2022	Виконано
8.	Аналіз результатів тестування	22.02.2022	15.03.2022	Виконано
9.	Відгук керівника КРБ	16.03.2022	31.03.2022	Виконано
10.	Оформлення КРБ та презентації	04.04.2022	18.04.2022	Виконано
11.	Попередній захист	19.05.2022	27.06.2022	Виконано
12.	Рецензування	27.05.2022	27.06.2022	Виконано
13.	Завершення оформлення КРБ та презентації	11.06.2022	27.06.2022	Виконано
14.	Захист кваліфікаційної роботи	27.06.2022	27.06.2022	Виконано

Розробив студент Круковська Ганна Юріївна _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи: доцент (б. в. з.) кафедри ІПЗ Швед Альона Володимирівна
(посада, прізвище, ім'я, по батькові) (*підпис*)

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Ігровий застосунок в жанрі аркада»

Студент 408 гр.: Круковська Ганна Юріївна

Керівник: доцент кафедри ІПЗ

Швед А. В.

Кваліфікаційна робота бакалавра присвячена дослідженню аналогів та розробці ігрового застосуноку в жанрі аркада.

Об’єктом роботи є процес розробки ігрового застосунку в жанрі головоломка.

Предметом роботи є підходи та інформаційні технології розробки ігрових застосунків.

Метою роботи є активізація пізнавальної діяльності та удосконалення рівня розумових та логічних (аналіз, синтез) операцій за рахунок розробки та впровадження комп’ютерних ігрових технологій в жанрі головоломка.

Практичне значення програмного застосунку полягає у можливості його використання в часи світової пандемії та війни в Україні як варіант проводження вільного часу.

Кваліфікаційна робота складається з фахового розділу та спеціальної частини з охорони праці та безпеки в надзвичайних ситуаціях.

Пояснювальна записка кваліфікаційної роботи бакалавра складається зі вступу, трьох фахових розділів, висновків, додатків, спеціального розділу з охорони праці та переліку джерел посилання.

У вступі визначається актуальність теми, предмет та об’єкт роботи та проводиться короткий огляд поставленої задачі.

У першому розділі проводиться аналіз предметної області, аналізуються наявні аналоги системи і з’ясовуються етапи розробки веб-застосунків.

У другому розділі проводиться аналіз вимог до системи та етапів її створення. Визначається технічне завдання до програмного забезпечення, що проєктується, та наводяться сценарії його використання. Також проводиться вибір засобів для розробки, створення інтерфейсу інформаційного порталу, розробка діаграми класів та проєктування бази даних.

У третьому розділі наведений опис процесу розробки програмного застосунку.

У висновках проводиться аналіз проведеної роботи та отриманих результатів.

У спеціальній частині з охорони праці та безпеки в надзвичайних ситуаціях йдеться про техніку безпеки при роботі в офісних приміщеннях із комп'ютерним обладнанням.

Кваліфікаційна робота бакалавра викладена на 53 сторінки, вона містить 4 розділи, 19 ілюстрацій, 4 таблиці, 20 джерел в переліку посилань.

Ключові слова: розробка ігор, аркади, Python.

ASTRACT

of the Bachelor’s Thesis

" Game application in the arcade genre "

Student of group 408: Krukovska Hanna Yuriivna

Supervisor: associate professor, Shved A. V.

The qualification work of the bachelor is devoted to the research of analogues and the development of a game application in the arcade genre.

The object of the work is the process of developing a game application in the puzzle genre.

The subject of the work are approaches and information technologies of game application development.

The aim of the work is to intensify cognitive activity and improve the level of mental and logical (analysis, synthesis) operations through the development and implementation of computer game technology in the genre of puzzle.

The practical significance of the software application lies in the possibility of its use during the world pandemic and the war in Ukraine as an option for spending free time.

Qualification work consists of a professional section and a special section on labor protection and safety in emergencies.

The explanatory note of the bachelor's qualification work consists of an introduction, three professional sections, conclusions, appendices, a special section on labor protection and a list of reference sources.

The introduction determines the relevance of the topic, subject and object of the work and provides a brief overview of the task.

The first section analyzes the subject area, analyzes the existing analogues of the system and clarifies the stages of development of web applications.

The second section analyzes the requirements for the system and the stages of its creation. The technical task for the designed software is defined and the scenarios

of its use are given. There is also a choice of tools for development, creation of the information portal interface, development of class diagrams and database design.

The third section describes the process of developing a software application.

The conclusions analyze the work done and the results obtained.

The special section on health and safety in emergencies deals with safety when working in offices with computer equipment.

The qualifying work of the bachelor is presented on 53 pages, it contains 4 sections, 19 illustrations, 4 tables, 20 sources in the list of references.

Keywords: game development, arcade, Python

ЗМІСТ

ВСТУП

У сучасному світі набирає обертів діяльність програміста: створення ПЗ, верстка сайтів, загальний розвиток цивілізованого світу - це ключові моменти його роботи. Широке розповсюдження комп’ютерної роботизованої індустрії вимагає ще більш високого попиту на програми штучного інтелекту, взаємодії машини та людини. Зараз вона поглинає всі сфери людського життя, без виключення: здоров'я, навчання, роботу, спілкування, розваги, і багато іншого.

Розваги в сьогоденні не можуть не включати в себе ігри на ПК. Ігри дають додаткову можливість знайти собі хобі і в дитинстві, і в зрілому віці. Ігри до вподоби людям також через те, що вони дають шанси пізнати більше, розширити кругозір, думати нетипово й індивідуально, знаходити рішення на складні питання й вирішувати непрості довгі квести. Вони не втратять популярність ні серед дітей, ні серед дорослої молоді ще багато часу, адже розвиваються пліч о пліч з усіма технологіями. Ігри - це добрий спосіб розвити логіку, пам'ять та реакцію

Тема роботи є **актуальною**, оскільки в наш час важливе постійне покращення логічного та критичного типів мислення.

Метою кваліфікаційної роботи є активізація пізнавальної діяльності та удосконалення рівня розумових та логічних (аналіз, синтез) операцій за рахунок розробки та впровадження комп’ютерних ігрових технологій в жанрі аркада.

Для досягнення поставленої мети необхідно розв’язати наступні **завдання**:

- аналіз предметної області та існуючих аналогів;
- розробка вимог до ігрового застосунку;
- моделювання та проектування застосунку;
- розробка дизайну ігрових сцен;

- програмна реалізація, тестування та відлагодження прототипів гри..

Об’єктом роботи є процес розробки ігрового застосунку в жанрі аркада.

Предметом роботи є підходи та інформаційні технології розробки ігрових застосунків.

Метою спеціального розділу про охорону праці є дослідження норм охорони праці, які пов’язані з роботою розробника програмного забезпечення.

1. АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ РОЗРОБКИ ІГРОВИХ ЗАСТОСУНКІВ

1.1 Опис предметної сфери розробки комп’ютерних ігор

Відеоігри зараз є важливим засобом як в економічному, так і в культурному плані, як невід’ємна частина індустрії розваг та креативного сектора. Однак естетика та техніки відеоігор також застосовуються в набагато ширшому спектрі діяльності – вони підтримують процес навчання дітей, молоді та дорослих у класі та на робочому місці, забезпечують веселий спосіб засвоєння складного змісту, і мотивувати пацієнтів у медичних та терапевтичних установах.

1.2 Жанри комп’ютерних ігор

Хоча колись жанри відеоігор були досить чіткими, сьогодні це не так. Існує зростаюча різноманітність жанрів і піджанрів, які потрібно зрозуміти, особливо коли розробники ігор змішують різні типи ігор новими та несподіваними способами.

Це означає, що ландшафт відеоігор постійно розвивається. Студії працюють за щільним графіком і слідкують за тенденціями, коли з’являється можливість. Нозглянемо 10 найрелевантніших категорій відеоігор:

- пісочниця;
- стратегія реального часу;
- шутери;
- багатокористувацька онлайн-бойова арена;
- рольові ігри;
- симулятор і спорт;
- головоломки;

- екшн-пригодницькі;
- виживання і жахи;
- платформер.

Багато жанрів певною мірою перетинаються один з одним. Навіть у перші дні ігор такі терміни, як «дія» та «бій», могли охоплювати багато назв. Ось чому багато геймерів вважають за краще використовувати більш конкретні назви, подібні до тих, які ми розглянемо в цій статті, щоб розрізнити стиль гри.

Термін «пісочниця» може бути більш впізнаваним за його використанням у техніці або навіть як відкритий режим, доступний у певних іграх. Це часто асоціюється з вибором гравця, відкритим середовищем і нелінійним ігровим процесом. Жанр пісочниці виріс із невеликої ніші, щоб охопити величезну різноманітність назв.

У цих іграх гравці часто мають менш конкретні цілі та шляхи розповіді. Замість того, щоб бити боса та рятувати принцесу, ви можете зіткнутися з різними завданнями, які можна виконати різними способами. Це залучає гравців до більш захоплюючого досвіду, заохочуючи експериментувати з тим, що може бути незнайомою механікою.

Назви пісочниці іноді можуть бути дуже концептуальними і навіть не мати деяких з найбільш впізнаваних елементів ігрового процесу. Ключовим прикладом є гра Elite 1984 року з простим дизайном і грою, зосередженою на бою, розвідці та торгівлі.

Ігри-симулятори, такі як The Sims, також все частіше рекламуються як ігри з пісочницею, як і багато популярних франшиз, включаючи Minecraft і Grand Theft Auto.

Приклади пісочниці: Майнкрафт, Grand Theft Auto, The Sims.

Спочатку придумані як маркетинговий термін для Dune II від Westwood Studios, стратегічні ігри в реальному часі існували роками, перш ніж більшість

гравців дізналися, що це за жанр. Завдяки своїй незмінній популярності та зростанню нових піджанрів, ігри RTS залишаються помітною частиною ландшафту відеоігор.

У типовій назві RTS люди Dune II і гравці з штучним інтелектом контролюють різні фракції та змагаються один з одним одночасно в «реальному часі», звідси й термін «стратегія в реальному часі», на відміну від покрокової стратегії. Ці ігри, як правило, включають управління ресурсами та картами, і вони часто мають вигляд зверху вниз.

Warcraft, Age of Empires і Command & Conquer є одними з найпопулярніших ігор у реальному часі, але на цьому список не закінчується. А коли справа доходить до покрокових стратегічних ігор, ентузіасти рекламують серію Civilization та інші відомі франшизи. Є також назви, які навмисно поєднують елементи обох стилів для змішаного ігрового процесу, як-от франшиза Total War. Щоб дізнатися більше про ігри, перегляньте наш список найкращих ігор у реальному часі.

Приклади: Warcraft, Command & Conquer.

Багатокористувацькі онлайн-ігри на бойовій арені, що стають все більш популярним піджанром із підключенням до багатьох інших стилів, мають багато функцій зі стратегіями в реальному часі. Існує перспектива зверху вниз, яка підкреслює управління картами та ресурсами, а також конкуренцію між гравцями в реальному часі.

Основна відмінність між іграми MOBA та RTS – це характер і роль гравця. У MOBA у вас може бути розподіл фракцій і багато основ RTS, але зазвичай ви керуєте лише одним персонажем. Це значний контраст з більшістю ігор RTS, де ви створюєте спільноти та керуєте кількома підрозділами.

Ігри MOBA також надають пріоритет багатокористувацької та командної гри. Хоча деякі добре відомі ігри мають елементи, керовані штучним

інтелектом, зазвичай ви граєте з іншими гравцями-людьми та змагаєтеся з ними, щоб виконати набір умов перемоги.

У цій ніші домінують Dota 2 і League of Legends, обидві з яких мають значну аудиторію в кіберспорті.

1.3 Аналіз ігрових застосунків в жанрі аркада

Ігри в жанрі аркада різко набули популярності ще в 1970-і та 1980-ті роки, «золотий вік аркади». Незважаючи на увагу, яку привертає екран відеоігор, ранні аркадні ігри охоплювали багато вимірів фізичного досвіду, як-от гра для восьми гравців Indie 800 1975 року, яка мала кермо та дві педалі для кожного гравця, а також екран. Ці відео-аркадні автомати були відгалуженням від попередніх механічних ігор, таких як пінбол. Тому дизайнери звернули увагу на відчутні аспекти взаємодії гри. Такі машини та їх попередні механічні аналоги демонстрували в аркаді відчуття видовища, і деякі автори помітили, що ранні аркади були дуже соціальним явищем.

Рисунок 1.1 – Електромаханічна аркда.

Система в першу чергу призначена для організації вільного часу користувача. Створюване ПЗ повинно бути короткою по часу, але інтенсивною по діяльності грою.

Основні характеристики існуючого аналогу:

- назва: Shotgun King: The Final Checkmate;
- жанр: аркади, екшени, інді, стратегії;
- розробник: Punkcake Delicieux;
- видавник: Punkcake Delicieux;
- дата виходу: 12 травня 2022;
- мова реалізації: англійська;
- перелік функцій, характеристик: реєстрація користувача, авторизація користувача, перегляд рівней;
- аналіз переваг та недоліків даного ПЗ:
  - переваги: стабільність, юзер-френдлі інтерфейс;
  - недоліки: відсутність можливості спілкуватись в системі, можливість гри на двох;
- джерело застосунку: https://store.steampowered.com/app/1972440/Shotgun_King_The_Final_Checkmate/.

Інтерфейс цього застосунку зображено на рис. 1.2.

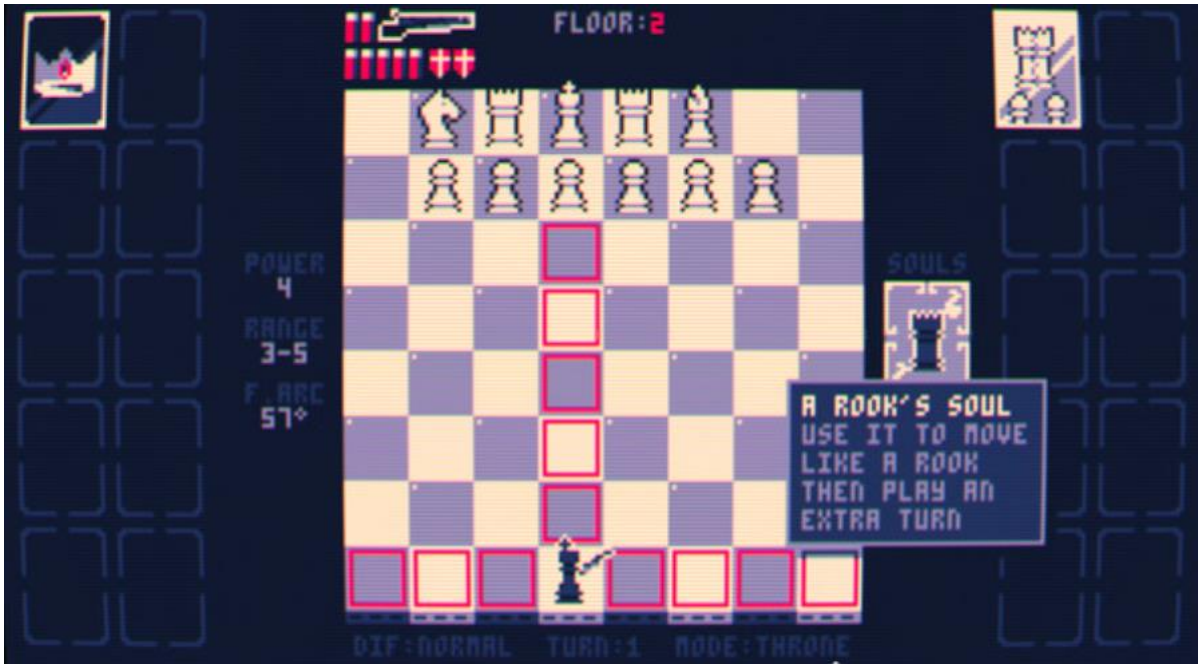


Рисунок 1.2 – Інтерфейс гри Shotgun King: The Final Checkmate.

Основні характеристики існуючого аналогу:

- назва: Golf Gang;
- жанр: аркади, екшени, казуальні ігри, гонки, спортивні ігри;
- розробник: Lazy Monday Games;
- видавник: Curve Games;
- дата виходу: 19 травня 2022;
- мова реалізації: англійська;
- перелік функцій, характеристик: реєстрація користувача, авторизація користувача, перегляд рівней;
- аналіз переваг та недоліків даного ПЗ:
  - переваги: стабільність, юзер-френдлі інтерфейс, простота гри;
  - недоліки: відсутність можливості гри на двох з одного ПК;
- джерело застосунку: https://store.steampowered.com/app/1151050/Golf_Gang/

Рисунок 1.3 – інтерфейс гри Golf Gang.

Основні характеристики існуючого ПЗ:

- назва: Void Prison;
- жанр: аркади, екшени, казуальні ігри, спортивні ігри;
- дата виходу: 20 травня. 2022
- розробник: Voidrock Studios
- видавник: Voidrock Studiosмова реалізації: англійська;
- перелік функцій, характеристик: реєстрація користувача, авторизація користувача, додавання друзів, створення команди;
- аналіз переваг та недоліків даного ПЗ:
  - переваги: стабільність, юзер-френдлі інтерфейс, простота гри;
  - недоліки: відсутність можливості зберегти прогрес для подальшої гри;
- джерело застосунку: https://store.steampowered.com/app/1964260/Void_Prison/
-

1.4 Специфіка вимог до програмного забезпечення комп'ютерної гри

Для успішної реалізації задуманого проекту треба розуміти етапи його створення.

Основні етапи створення веб-застосунку такі:

- Аналіз вимог і написання ТЗ.
- Прототипування.
- Розробка дизайну інтерфейсу.
- Програмування.
- Тестування.

1. Розробка дизайну.

На цьому етапі дизайнер пророблює представлення інтерфейсу гри, що проєктується.

2. Програмування.

Було вирішено розробляти застосунок на мові програмування Pyton. Мова програмування Python - це універсальна високорівнева скриптова мова, яка ідеально підходить для будь-яких платформ (будь то iOS, Android або навіть операційні системи серверів), для вирішення різних типів задач. Її використовують наприклад в розробці багатьох застосунків: мобільних та десктопних; вона також підходить для програмування різних ігор, для машинного навчання (нейронні мережі та штучний інтелект) та аналітики.
Python - це мова, що не компілюється, тобто інтерпретована. Це значить, що до запуску код являє собою текстовий файл. Розробка цією мовою відбувається помітно швидше за розробки, наприклад, мовами C або Java, так як кодування достатньо добре спроектовано: коду доводиться писати набагато менше [1]. За допомогою мови Python можна реалізовувати проекти майже на всіх платформах, вона логічна й до того ж ідеальна для новачків.

Цю мову можна зустріти майже будь-де: у мережі, на мобільних пристроях, в багатьох застосунках, у розробках, пов'язаних із навчанням машин, та в якості систем, що вбудовуються. Але частіше за всього мова Python використовується у веб-розробці, для написання парсерів, що збирають і колекціонують контент в інтернеті [2].

3. Тестування.

Ресурс перевіряється на відповідність вимогам, продуктивності. Остаточна перевірка якості — це оцінка готового результату як з точки зору програміста, так і з точки зору користувача.

Висновки до розділу 1

У розділі 1 було проведено аналіз предметної сфери розробки ігрових за стосунків. Було з’ясовано значення поняття «аркада», визначення наведеного терміну допомогло з’ясувати його значення та актуальність.

Було обрано три аналоги гри-застосунку, які було проаналізовано окремо та визначено для них характерні риси, такі як назва, жанр, розробник та видавник, мова випуску, перелік функцій та інше. Було також виокремлено характерні переваги та недоліки ігрових застосунків, що було розглянуто.

У заключній частині розділу 1 було розглянуто етапи розробки веб-застосунків для подальшої успішної реалізації власної системи.

2. МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ ГРИ

2.1 Діаграма прецедентів системи

Технічне завдання (ТЗ) визначає атрибути, методи та структуру проекту. Він створений на основі результатів фахового аналізу та досліджень з урахуванням уподобань клієнтів, особливостей бізнес-процесів у кожній деталі. Простіше кажучи, це точний посібник, який визначає чіткі та послідовні методи створення продукту, а також обладнання та технологію, які будуть використані, та передбачувані результати.

Програмне забезпечення гри «Змійка», що проєктується, має систематизувати взаємодію системи із користувачем.

Система повинна містити такі сторінки:

- ігровий інтерфейс з екраном початку гри;
- головна сторінка процесу гри;
- сторінка програшу;
- сторінка виграшу.

Користувачі системи: незареєстрований гравець, зареєстрований гравець, система.

Незареєстрований користувач має можливість почати гру чи авторизуватися та почати гру.

Вже зареєстрований користувач продовжити гру зі збереженої точки.

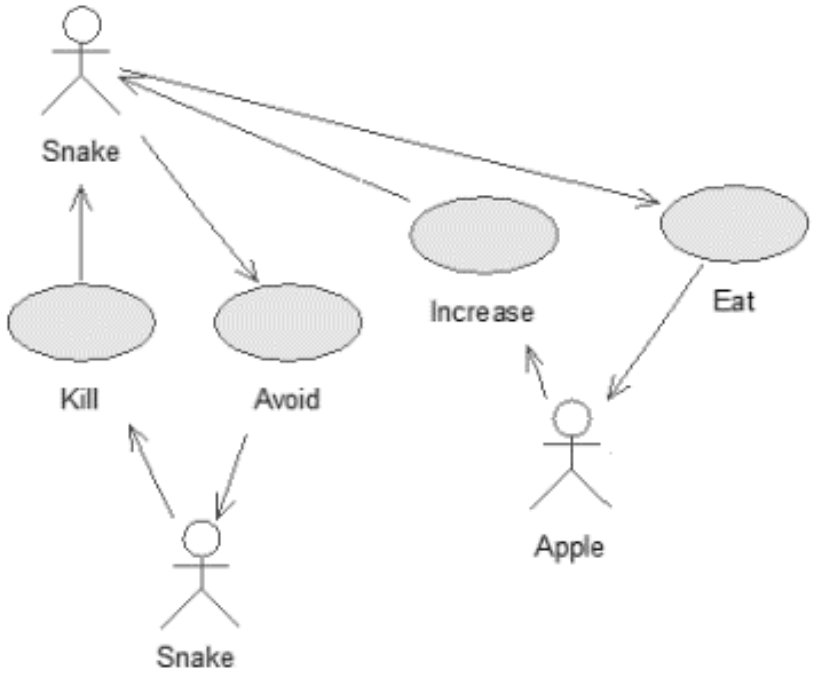


Рисунок 2.1 – Діаграма прецендентів системи

2.2 Діаграма станів

Сценарій використання, або короткий сценарій, показує, як один або кілька людей або організацій взаємодіють з технологією в реальному світі. Вони описують події, кроки та/або дії, які відбуваються під час контакту. Сценарії використання можуть бути досить ретельними, пояснюючи, як саме хтось взаємодіє з інтерфейсом користувача, або вони можуть бути досить високого рівня, визначаючи важливі бізнес-завдання, але не те, як вони виконуються[5].

У грі, що проєктується, визначається три ролі: незареєстрований користувач, зареєстрований користувач, система. Система виконує свої завдання на програмному рівні та контролює сам процес роботи застосунку. На таблицях 2.1-2.6 можна побачити приклади сценаріїв використання системи.

Таблиця 2.1 – Сценарій №1: Створення гра.

Діючі особи	Незареєстрований користувач, система
Мета	Створити гру.
Передумова	Користувач не авторизований.
Успішний сценарій: 1. Користувач бачить сторінку реєстрації. 2. Заповнює свої дані. 3. Система зберігає обліковий запис. 4. Система переводить користувача на гру автоматично.	
Результат	Створено нову гру.
Розширення	
***а**	Система не зберігає новий обліковий запис. **Результат:** користувач не може зареєструватись.

Таблиця 2.2 – Сценарій №2: Створити гру

Діючі особи	Користувач, система
Мета	Авторизуватися.
Передумова	Користувач авторизований.
Успішний сценарій: 1. Користувач знаходить необхідні для авторизації дані (ім’я). 2. Користувач авторизувався в системі.	
Результат	Користувач авторизувався в системі.

Таблиця 2.3 – Сценарій №3: Закінчити гру успішно

Діючі особи	Користувач, система
Мета	Завершити гру
Передумова	Немає
Успішний сценарій: 1. Користувач переходить до облікового запису. 2. Користувач набирає достатню кількість балів для виграшу. 3. Система показує вікно виграшу.	
Результат	Гру пройдено успішно

Таблиця 2.4 – Сценарій №4: Оперування інтерактивним розкладом

Діючі особи	Користувач, система
Мета	Завершити гру.
Успішний сценарій: 1. Користувач переходить до облікового запису. 2. Користувач натрапляє на тіло змії. 3. Система показує вікно програшу.	
Результат	Користувач завершив гру.

Для розуміння можливостей кожного з учасників досить створити діаграму використання або ж діаграму прецедентів. Діаграма варіантів використання — це наочне уявлення деталей системи та її користувачів. Зазвичай він зображується як графічне зображення взаємодій між різними елементами в системі. Діаграми варіантів використання описують події, які відбуваються в системі та як вони протікають, але не визначають, як ці події

реалізуються[6]. На рис. 2.1 зображена діаграма використання гри, що розробляється.

Як вже було зазначено, основними дійовими особами є неавторизований користувач, авторизований користувач, система.

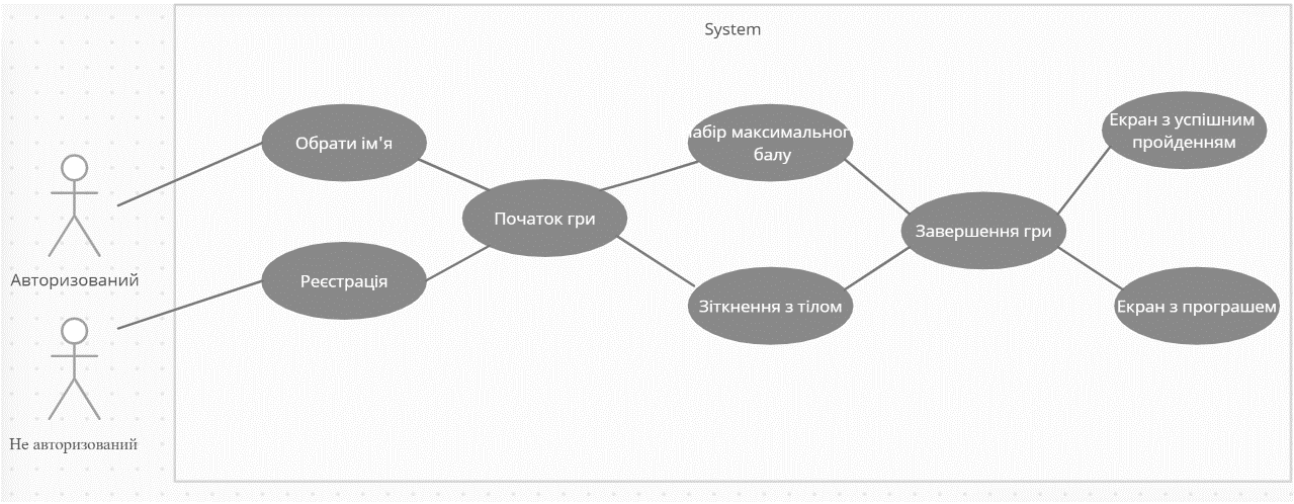


Рисунок 2.2 – Діаграма варіантів використання

2.3 Вибір програмних засобів для розробки

Розробка веб-застосунків вимагає ретельного вибору технологій і підходів. Важливо також зрозуміти, як вони були відібрані для проекту.

При виборі технології важливо враховувати наступне[8]:

- розмір і складність проєкту;
- швидкість розвитку;
- вартість і наявність фахівців;
- тенденція розвитку;
- наявність ґрунтовної документації;
- вартість підтримки;
- потреби в навантаженні;
- вимоги безпеки;

– потенціал для інтеграції з іншими рішеннями.

Мова програмування Python - це універсальна високорівнева скриптова мова, яка ідеально підходить для будь-яких платформ (будь то iOS, Android або навіть операційні системи серверів), для вирішення різних типів задач. Її використовують наприклад в розробці багатьох застосунків: мобільних та десктопних; вона також підходить для програмування різних ігор, для машинного навчання (нейронні мережі та штучний інтелект) та аналітики.

Python - це мова, що не компілюється, тобто інтерпретована. Це значить, що до запуску код являє собою текстовий файл. Розробка цією мовою відбувається помітно швидше за розробки, наприклад, мовами C або Java, так як кодування достатньо добре спроектовано: коду доводиться писати набагато менше. За допомогою мови Python можна реалізовувати проекти майже на всіх платформах, вона логічна й до того ж ідеальна для новачків.

Цю мову можна зустріти майже будь-де: у мережі, на мобільних пристроях, в багатьох застосунках, у розробках, пов'язаних із навчанням машин, та в якості систем, що вбудовуються. Але частіше за всього мова Python використовується у веб-розробці, для написання парсерів, що збирають і колекціонують контент в інтернеті.

На Python часто розробляють вбудовані системи, як наприклад системи управління банкоматами “Сбербанк” або для неймовірно маленького комп’ютера Raspberry Pi. Мова також використовується у вбудованих системах верстатів, таких як засоби автоматичного регулювання температури, тиску чи витрати рідини та навіть у телекомунікаційному обладнанні.

Функціонал мови також розповсюджується на написання скриптів, плагінів і додаткових модулів для вже повністю готових програм, наприклад

для реалізації ігрової логіки. Також наявні варіанти написання скриптів, які мають бути вбудовані в програми, що написані іншими мовами, щоб автоматизувати потрібні задачі.

Python з великим успіхом використовується системними адміністраторами для автоматизації задач, так як він простий та потужний для такого функціоналу. Він також підтримує спеціальні пакети, котрі можуть підвищити його ефективність в роботі. Завдяки лаконічності мови можна з високою швидкістю прочитати готовий код і знайти слабкі місця й відлагодити їх. Форматування в мові - це важлива частина синтаксису.

Також велика кількість ігор були повністю або частково написані на Python. В цей список входять ігри Battlefield 2, World of Tanks, Civilization та інші.

2.4 Діаграма класів

Діаграма класів — це свого роду діаграма, яка визначає та надає огляд і структуру системи з точки зору класів, характеристик, методів і взаємодій між різними класами. Вона є частиною уніфікованої мови моделювання (UML).

У свою чергу UML — це мова графічного опису для моделювання об’єктів у розробці програмного забезпечення, бізнес-процесів, системної інженерії та візуалізації організаційної структури. У своєму описі вона містить:

- класи;
- їх атрибути;
- операції (або методи);
- взаємозв'язки між об'єктами;

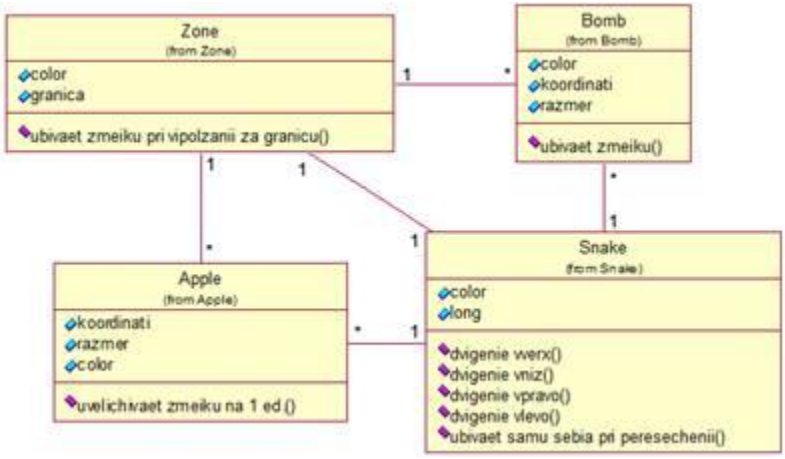


Рисунок 2.3 – Діаграма класів

На рис. 2.3 зображена діаграма класів до програмного забезпечення.

Висновки до розділу 2

У ході виконання розділу 2 було розроблено технічне завдання для ігрового застосунку, що проєктується. Було визначено основних дійових осіб системи та основні вимоги до веб-застосунку.

Далі було наведено сценарії використання системи, наведено діаграми прецендентів та станів, які визначили та надали свого роду структурності системі, показали взаємодію класів, їх атрибутів та взаємозв’язок взагалі.

На основі другого розділу можна приступати до безпосередньої розробки ігрового застосунку в жанрі аркада.

3. ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ ГРИ

3.1 Принципи проєктування інтерфейсів ігрових застосунків

Дизайн – це вирішення проблем, а не просто малювання картинок. Хороший дизайн служить рішенням проблем користувачів з метою полегшити їм життя.

Правила дизайну інтерфейсу користувача єдині, маючи на увазі користувача. Правила мають на меті допомогти розробити кращі інтерфейси, які вирішують прості або складні проблеми, які потенційно можуть вплинути на загальну роботу користувача. Важливо спочатку зрозуміти проблему, щоб створити хороший інтерфейс.

Візуальна привабливість інтерфейсу відіграє важливу роль у загальній привабливості дизайну. Однак зосередження лише на зовнішньому вигляді інтерфейсу найчастіше відриває від мети чи значення дизайну. Візуальні теми привабливі, але вони не дають уявлення про те, що, як і чому стоїть за вибором конкретної теми дизайну.

Візуальна тема інтерфейсу — це набір піктограм, колірної палітри, типографіки, фотографій, кнопок, вкладок, повзунків тощо. Але простий погляд на ці візуальні елементи нічого не говорить про проблеми, для вирішення яких вони створені. Ми не розуміємо, як були прийняті ці дизайнерські рішення, і знаємо, чи привабливі картинки, які ми дивимося на екрані, доречні та доречні.

Потрібно чітко бачити створення візуальної теми та стилю інтерфейсу, маючи на увазі мету та контекст використання програми чи веб-сайту. Кожен візуальний елемент дизайну повинен передавати голос і повідомлення, а не лише додавати естетичну цінність інтерфейсу.

3.2 Стратегії розробки інтерфейсів ігрових застосунків

Відеоігри в основному розглядаються як творча платформа для інтерактивного оповідання, але з технічної точки зору це лише програмне забезпечення, яке вимагає переважно традиційних, але іноді й сучасних типів введення від користувача. Навіть у найпростіших іграх вам знадобиться використовувати навігаційне меню, щоб запустити, завантажити або зберегти гру.

Оскільки у людини обмежене сенсорне сприйняття в грі, ми повинні покладатися на HUD, коли розміщуємо елементи в цифровому середовищі. Дисплей Heads-Up призначений не виключно для ігор; його перше застосування було в сучасних літаках, де пілоти-винищувачі могли перевіряти важливу інформацію на фактичному вітровому склі, зменшуючи відволікання під час польоту. Комерційна версія також з’явилася в автомобілях, де найважливіші статуси транспортного засобу проектуються на лобове скло, щоб водії могли отримати доступ до цієї інформації, не відриваючи очей від дороги. Мета HUD в іграх однакова: переглядати важливу інформацію та статуси без найменшої перешкоди ігровому процесу. Стан здоров’я або напрямок ворожого вогню – це те, що відчує та відчує наша персона в грі, але це має бути візуалізовано для когось, обмеженого лише поглядом на екран.

Інтерфейс користувача, система меню та навігація в цій системі визначаються фактичною складністю ігрового процесу або навіть характером розповіді. Уявіть собі більш кінематографічний ігровий процес, де гра виглядає майже як фільм, і ви відчуваєте це дуже лінійно. Ці ігри зазвичай не зосереджені на системах підвищення рівня, побічних квестах або іграх із відкритим світом, тому структура інтерфейсу, швидше за все, буде дуже простою.

Додавання складних елементів ігрового процесу значно ускладнить ситуацію. Control — це гра, випущена в 2019 році, але інтерфейс користувача, система меню та карта були далекі від досконалості. Розширена система оновлення спочатку була надзвичайною, і надмірне використання індикаторів не допомогло і без того відсутній візуальній ієрархії. Але все ж найочевиднішою проблемою було візуальне недостатнє уявлення майстерно складних, багатошарових і розширюючих рівнів. Обмежена, інколи зболіва, темно-монохромна карта не могла перевести 3D-дизайн рівнів у 2D-карту, що викликало багато розчарувань у геймерів. З моменту його першого випуску розробники налаштували та вирішили проблеми дизайну карти в кількох оновленнях, роблячи її зручнішою від оновлення до оновлення.

Безлад — причина номер один, чому інтерфейс гри може вийти з ладу. Багатокористувацькі онлайн-ігри є основними порушниками візуального безладу: окрім участі в ігровому процесі, ви також повинні звернути увагу на оновлення, розробку стратегії, використання спеціальних атак, читання карти та спілкування між людьми одночасно, будучи обмеженими. на одному екрані, щоб виконати всі ці дії.

Розробники знають, що в їхній грі може бути безлад. Gran Turismo Sport включає в себе багато параметрів налаштування автомобілів, що ускладнює навігацію за допомогою геймпада. Щоб допомогти користувачам, Polyphony Digital включила в систему меню «курсор миші». Очевидно, що це не найкраще рішення в ексклюзивній назві для консолі, оскільки трохи незручно обробляти його за допомогою контролера, але це ціна включення занадто великої кількості параметрів налаштування. Великий список опцій також може паралізувати рішення для гравця, який, у свою чергу, може знехтувати налаштуваннями покращення ігрового процесу.

Висновки до розділу 3

Протягом роботи над розділом було визначено поняття ігрового інтерфейсу. Це спеціальний зв’язок між програмним забезпеченням та користувачем.

Зазвичай, користувачі думають про інтерфейс лише як про візуальний компонент на екрані та інтерактивні складові, з якими обов’язково потрібно взаємодіяти. Інтерфейс в свою чергу також повинен реагувати на взаємодію з собою та вчасно повертати фідбек користувачу.

На цьому етапі розробки також було визначено головні стратегії розробки інтерфейсів ігрових застосунків та принципи проєктування інтерфейсів.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ ГРИ

4.1. Опис програмного функціоналу

Для початку роботи з написання гри потрібно завантажити бібліотеку PyGame на комп'ютер. Відкриваємо командний рядок комбінацією клавіш Windows та R. В рядку пошуку вводимо cmd, що дає нам можливість відкрити командне вікно. Тепер встановлюємо бібліотеку PyGame: вводимо команду pip install pygame та чекаємо завершення процесу завантаження і все готово для подальшої роботи.

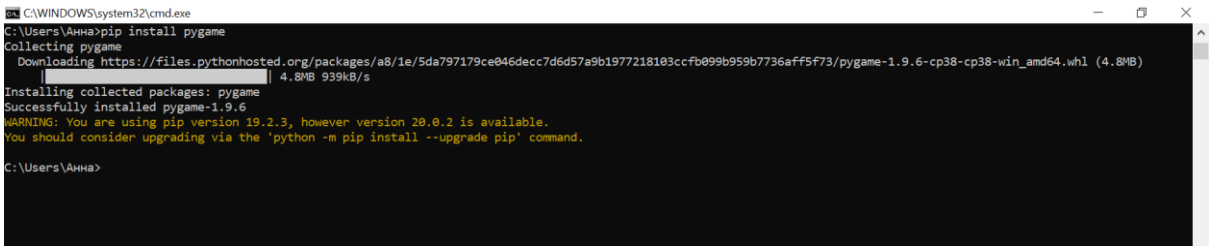


Рис.4.1 – Текст командного рядка

Створюємо новий Python файл та назвемо його main. Тепер необхідно імпортувати бібліотеку PyGame, яку далі потрібно ініціалізувати, тобто включити.

Для гри потрібно вікно, а також посилання на це вікно, щоб ми змогли ним управляти. В других дужках встановлюється його ширина та висота, які задаються методом SetMode. Щоб відобразити вікно, в PyGame існує метод Flip, який ми використовуємо.

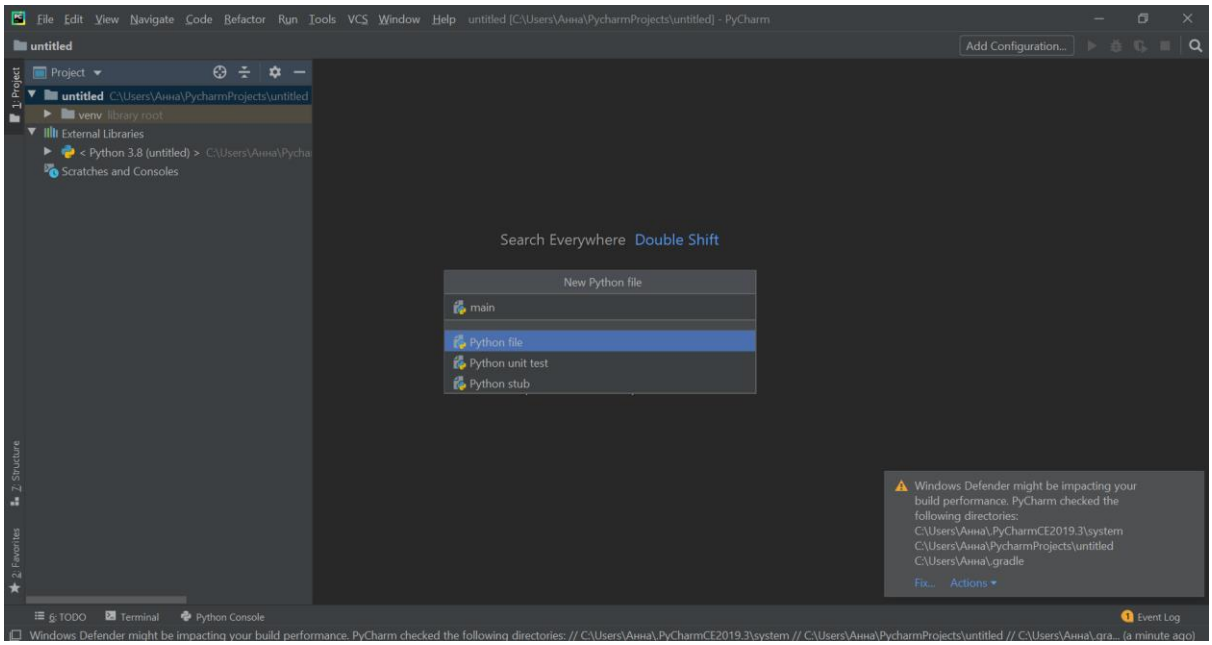


Рис. 4.2 – Створення файлу

При запуску програми на цьому етапі створене вікно відкрилося й одразу ж закрилося, адже в програмі відсутній основний цикл.

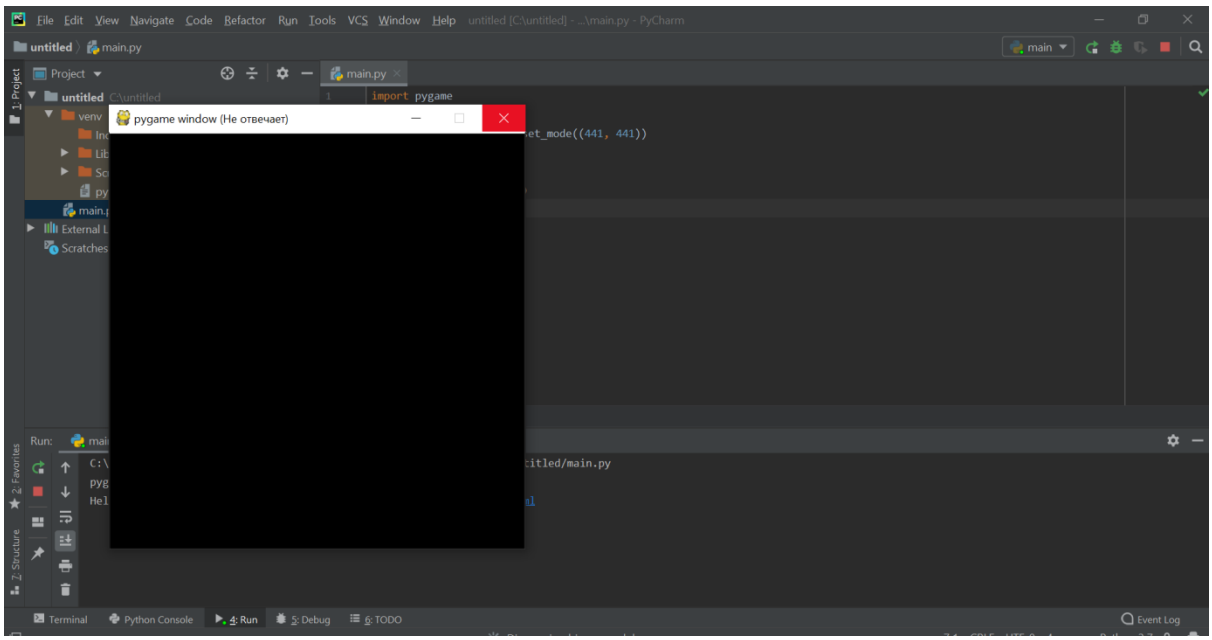


Рис. 4.3 – Вікно на початковому етапі розробки

Ми використаємо прапорець і далі запишемо потрібний нам цикл: перенесемо рядок коду, який відображає вікно гри і запустимо програму. Тепер вікно не закривається але також ми не маємо змоги закрити його, натиснувши “хрестик”. У бібліотеці PyGame для цього створений модуль Event, який може сприймати всі сигнали з клавіатури та миші. Щоб отримати ці сигнали, в модулі є метод Get, який повертає ці сигнали у вигляді списку. Тому для роботи з цим списком потрібно створити цикл For і умову: якщо тип події - клік на “хрестик”, то прапорець гри змінюємо на False, тобто завершуємо основний цикл нашої гри:

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        flag_game = False
```

При запуску вікно працююче, а також ми можемо закрити його звичним натиском на “хрестик”.

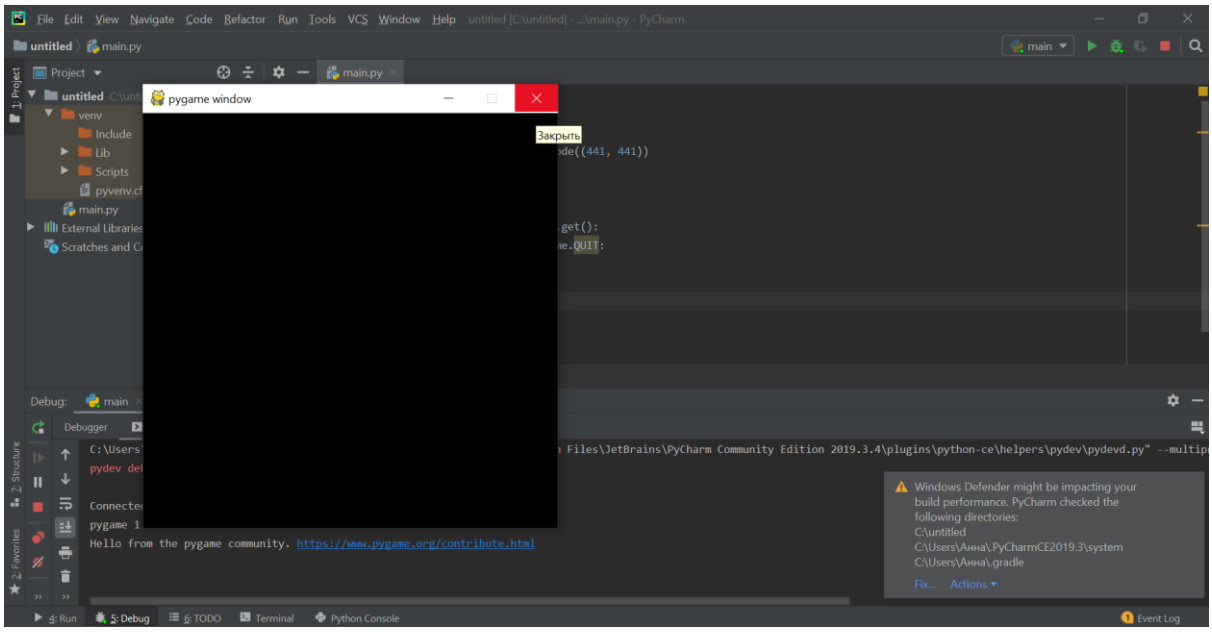


Рис. 4.4 – Вікно, що коректно закривається

Далі потрібно дати ім’я вікну гри, для цього записуємо рядок pygame.display.set_caption(), в дужках вказавши назву - “Змійка”. Після запуску назва вікна змінилася.

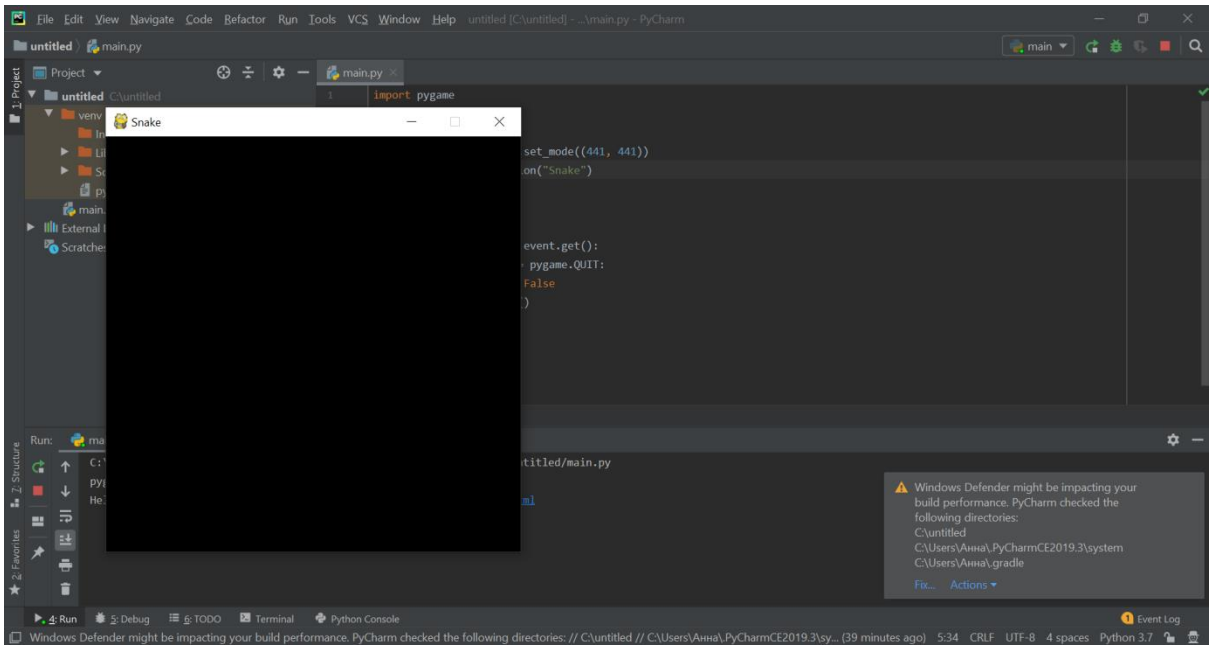


Рис. 4.5 – Вікно з назвою

Почнемо створювати головного персонажа гри - змійку, будуючи її по сегментам-квадратам. Для цього в бібліотеці PyGame існує модуль Draw, в якому є метод Rect. Цей метод вимальовує квадрат по вказаним координатам, який в подальшому буде головою нашої змії. Записуємо head і створюємо список з координатами x і y. Далі в основному циклі нам потрібно прописати малювання квадрату, вказавши поверхню, на котрій це буде зроблено. У нашому випадку це вікно. Вказуємо також колір нашого квадрату - зелений, в PyGame є вбудовані кольори. Далі визначемо квадрат,котрий ми будемо малювати, координати x та y та розмір квадрату в пікселях.

Тепер потрібно анімувати наш квадрат: координату x будемо змінювати на 25 пікселів.

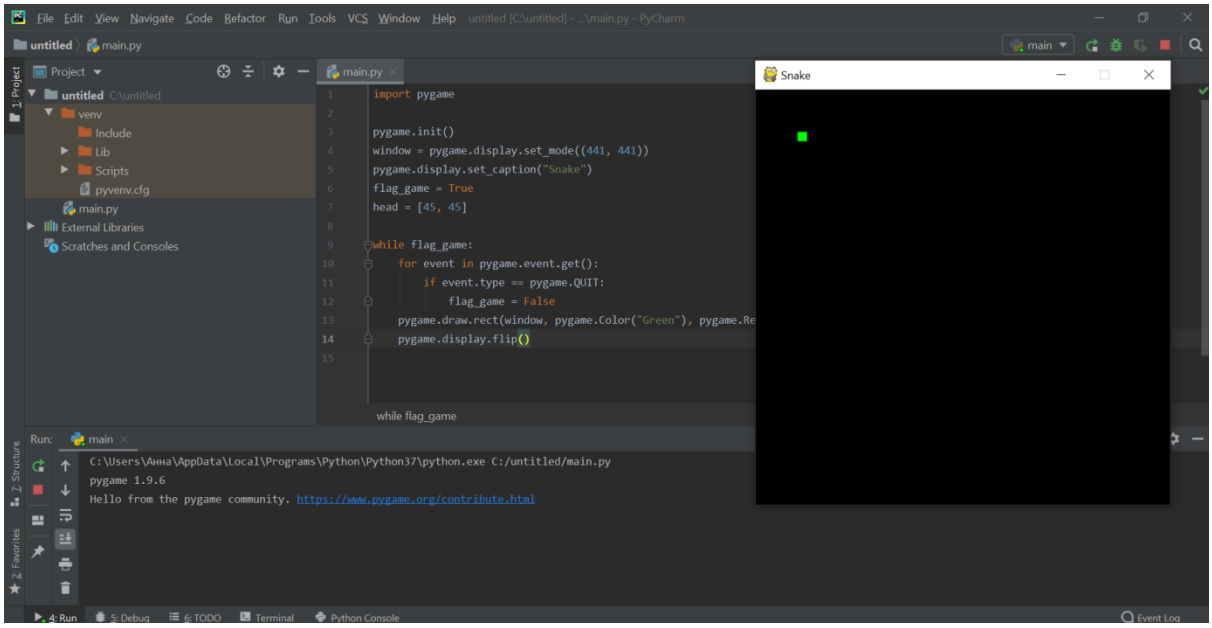


Рис. 4.6 – Вікно гри з елементом майбутньої змійки

Після запуску бачимо, що анімація занадто швидка, тому змінюємо її швидкість, додавши змінну val_speed=0. Додаємо умову: якщо наша зміна ділиться 350 на без остачі, то виконуємо додавання до координати x. Після запуску анімація стала повільнішою.

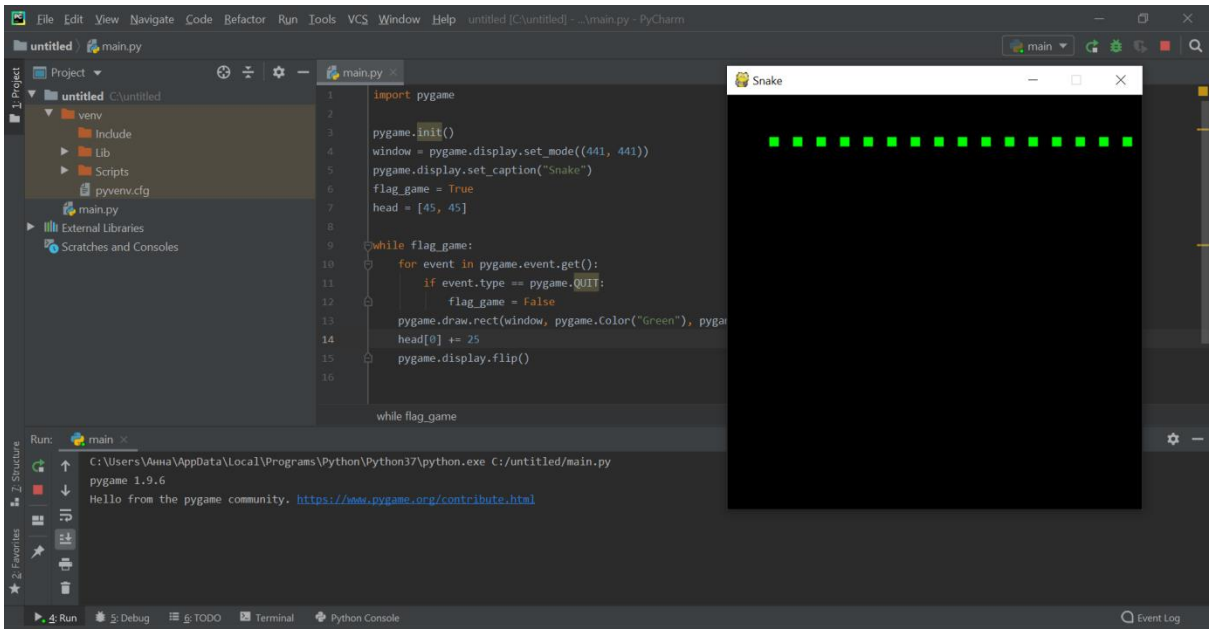


Рис. 4.7 – Анімація на початковому етапі

Але разом з тим ми отримали відмалювання багатьох квадратів, а нам потрібен тільки один. Для цього на вікні створимо нову поверхню за допомогою методу Fill, в дужках якого вказуємо колір - чорний. Після запуску маємо один квадрат, що рухається.

Перш ніж продовжити, спростимо наш код, тобто зробимо рефакторінг. Почнемо зі створення нового Python файлу, назвемо його control. Імпортуємо бібліотеку PyGame та створюємо клас Control та конструктор цього класу def __init__(self), в якому ми пропишемо два прапорця, необхідних для управління:

self.flag_game = True

self.flag_direction = "RIGHT"

Другий прапорець - направлення руху і ми присвоюємо йому значення за замовчуванням - право.

Далі створюємо метод control для управління залежно від прапорця. В цей метод переносимо код:

```
for event in pygame.event.get():
  if event.type == pygame.QUIT:
    self.flag_game = False
```

Прописуємо власне дії в залежності від натиснутої кнопки: якщо кнопка стрілки натиснута, то прапорець направлення змінюється в відповідну сторону:

```
elif event.type == pygame.KEYDOWN:
  if event.key == pygame.K_RIGHT:
    self.flag_direction = "LEFT"
  elif event.key == pygame.K_UP:
    self.flag_direction = "UP"
  elif event.key == pygame.K_DOWN:
    self.flag_direction = "DOWN"
  elif event.key == pygame.K_ESCAPE:
    self.flag_game = False
```

Останні рядки вказують на те, що гра закриватиметься натисканням на кнопку Escape.

Так як з написанням даного класу завершено, імпортуємо його в файл main, створюємо об'єкт цього класу і додаємо метод. При запуску програма працює і може закриватися натисканням на кнопку Escape.

Створюємо так само новий файл Snake, імпортуємо PyGame і створюємо клас Snake. Задаємо конструктор, створюємо метод move, який буде описувати рух змії в залежності від напрямку. Для цього нам потрібно передати аргумент control в метод, а також прапорець, який вже реалізований:

```
def move(self, control):
  if control.flag_direction == "RIGHT":
     self.head[0] += 25
  elif control.flag_direction == "LEFT":
     self.head[0] -= 25
  elif control.flag_direction == "UP":
     self.head[1] -= 25
  elif control.flag_direction == "DOWN":
     self.head[1] += 25
```

Далі створюємо новий метод draw, який вимальовує нашу змію на екрані, і передаємо сюди вікно для можливості відмальовки. Тепер записуємо pygame.draw.rect(window, pygame.Color("Green"), pygame.Rect(self.head[0], self.head[1], 10, 10)). Тут ми вказали поверхню, на котрій відбуватиметься відмальовка, в нашому випадку - вікно, колір змії, квадрат та його координати (координата x - елемент нульовий, координата y - елемент перший) і вказуємо його розмір 10, 10.

Усе готово, імпортуємо клас в головний файл, створюємо об'єкт класу і пишемо створені методи, в які передаємо відповідні аргументи. При запуску все працює, змія в грі рухається за заданими з клавіатури напрямками.

У класі Snake створюємо ще один список, котрий буде тілом нашої змії: self.body = [[45, 45], [34, 45], [23, 45]]. Звертаємо увагу на те, що в другій і третій парі координат координата x зменшується на 11. Це зроблено для того, щоб зробити між сегментами змії маленьку відстань в 1 піксель.

Тепер змінимо метод відмалювання змії: для цього створюємо цикл і дотримуємося табуляції наступного рядка. В цьому рядку змінюємо self.head на self.segment. Запускаємо програму та бачимо три правильних квадрата, що складають змію.

Коли власне тіло змії готове, потрібно правильно його проанімувати. Для цього ми додамо голову, в якої постійно змінюються координати, на початок списку після кожного проходження циклу. А хвіст в свою чергу ми будемо видаляти, щоб уникнути росту змії. І таким чином уся змійка буде рухатися коректно.

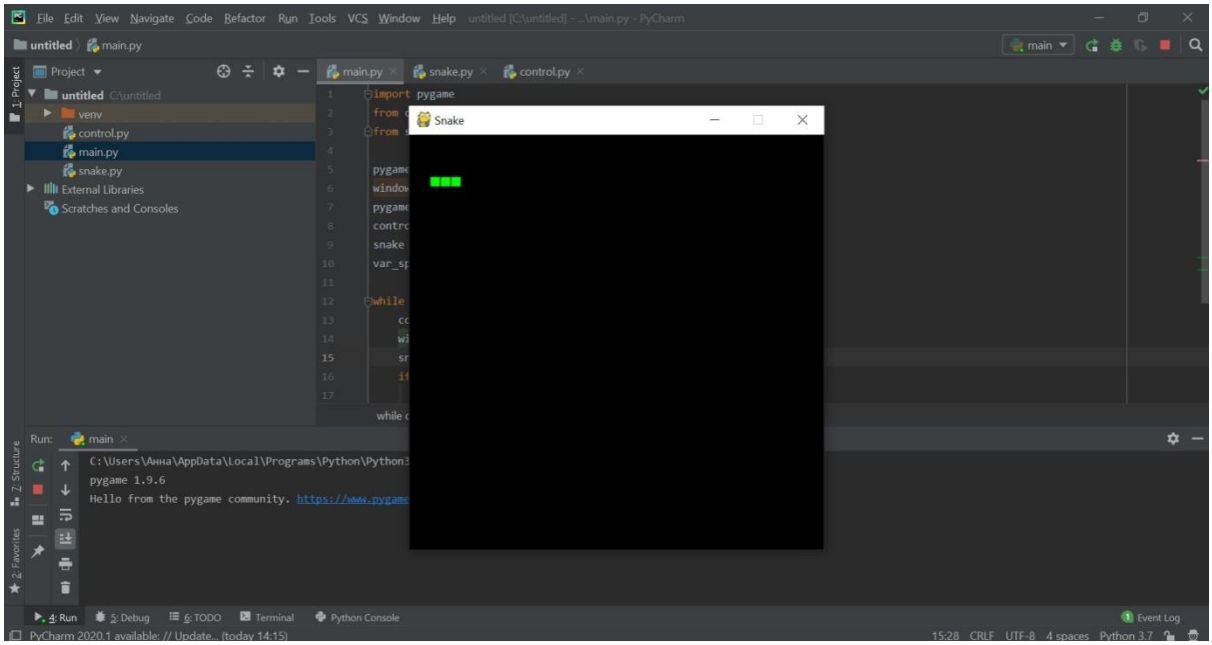


Рис. 4.8 - Три сегменти змії

Створюємо новий метод animation, в якому описуємо, що нульовий елемент буде додаватися, і використовуємо метод pop, який за замовчуванням видаляє останній елемент списку. Цей метод додаємо також у файл main в умову, так як метод анімації напряму залежить від методу руху move. Запускаємо й переконуємося, що все працює, але маємо недолік в управлінні: змія може змінювати рух на протилежний, тобто при натисканні кнопки вгору, коли змійка рухається вниз, вона змінює напрямок, це є помилкою для даної гри.

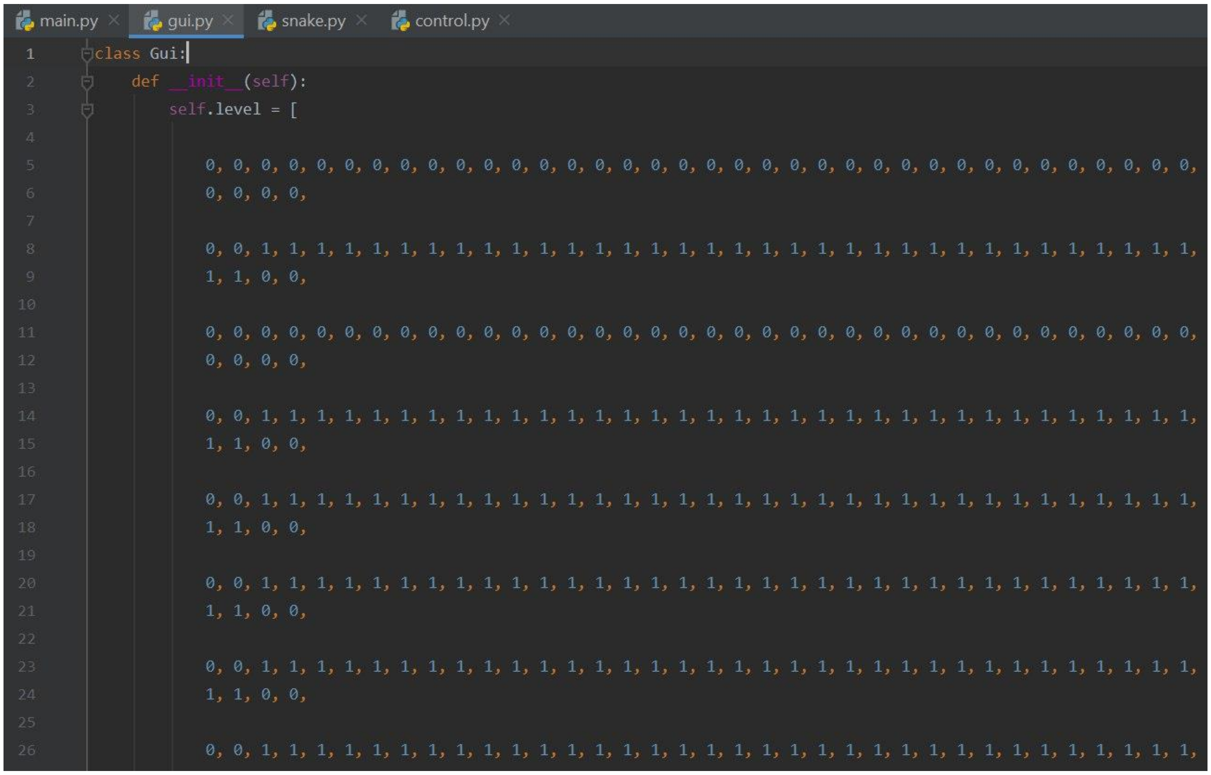
У control вказуємо додаткові умови: якщо змія повзе вправо, вона не може змінити напрямок на ліво, і навпаки. Так само і для сторін верх-низ. Доповнимо код: if event.key == pygame.K_RIGHT and self.flag_direction != "LEFT" і аналогічно для інших напрямків.

Додамо функцію паузи для нашої гри: для цього створимо ще один прапорець із назвою pause. Присвоїмо йому значення за замовченням True: self.flag_pause = "TRUE". І доповнимо список гарячих клавіш пробілом, який і буде грати роль кнопки паузи:

```
elif event.key == pygame.K_SPACE:
  if self.flag_pause:
     self.flag_pause = False
  elif self.flag_pause == False:
     self.flag_pause = True
```

Далі в main записуємо додаткову умову: if var_speed % 350 == 0 and control.flag_pause.

На даному етапі створимо графічний інтерфейс для нашої гри. Для цього створимо новий Python файл gui.py, в якому потрібно записати даний довгий список.




```
class Gui:
    def __init__(self):
        self.level = [

            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0,

            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 0, 0,

            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0,

            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 0, 0,

            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 0, 0,

            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 0, 0,

            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 0, 0,

            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Рис. 4.9 - Список

На основі цього списку ми і будемо будувати майбутній інтерфейс. В цьому списку нуль - це квадрат сірого кольору розміром 10x10, а одиниця - це порожнє місце.

Тепер створимо метод create_image, який на основі цього списку буде відмальовувати картину нашого графічного інтерфейсу. Далі нам потрібно створити поверхню, на котру ми будемо наносити наше зображення: pygame.surface((441, 441), pygame.SCALPHA, 32). Тут в перших дужках вказуємо розмір поверхні, тобто 441x441, після коми вказано, що одиниця

відповідає за прозорий елемент. Далі створюємо змінні x та y, присвоюємо їм 1, створюємо цикл для проходження по цим елементам: якщо значення і рівне нулю, то ми малюємо квадрат сірого кольору розміром 10x10 на поверхні, котру ми створили. Далі нам потрібно зберегти наше майбутнє зображення, для цього створюємо в проекті папку image, пишемо метод pygame.image.save(screen, “image/gui.png”). В main імпортуємо наш клас і викличемо метод, при запуску програма створила і зберегла зображення в папці. Тепер ми маємо вивести нове зображення на екран.

Для цього завантажуємо наше зображення і створюємо метод draw_level для відмальовки нашого графічного інтерфейсу. І вкажемо його в main перед методом відмалювання змійки. Запускаємо і переконуємося, що наш графічний інтерфейс відображаєтьсякоректно.

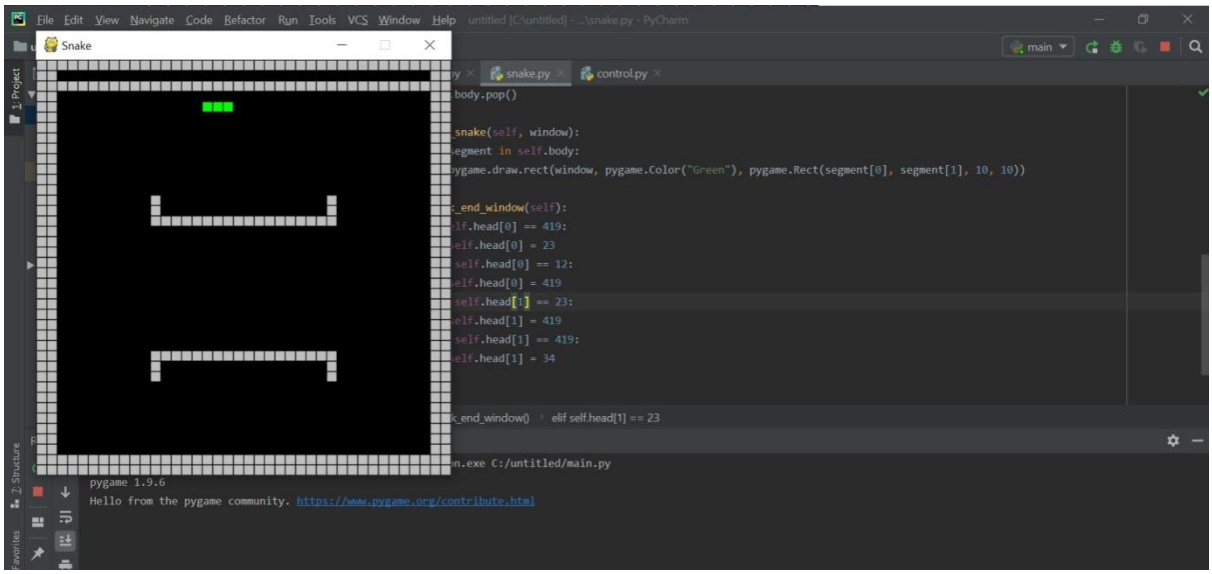


Рис. 4.10 - Графічний інтерфейс

Але як бачимо змія почала рухатися дуже повільно через високе навантаження на систему, тому повертаємося на початок створення коду і

змінюємо число в рядку з var_speed на 50 (для комп'ютерів з різною продуктивністю це число може відрізнятися). При запуску змія рухається з достатньою швидкістю для комфортної гри.

Тепер для співпадіння нашої гри з оригіналом, потрібно реалізувати метод, за допомогою якого змійка, стикаючись з якоюсь стінкою вікна, виповзала з протилежної стінки. Для цього створимо метод check_end_window, в якому запишемо наступний код:

```
def check_end_window(self):
    if self.head[0] == 419:
        self.head[0] = 23
    elif self.head[0] == 12:
        self.head[0] = 419
    elif self.head[1] == 23:
        self.head[1] = 419
    elif self.head[1] == 419:
        self.head[1] = 34
```

Цей метод викличемо в main і при запуску гра відпрацьовуєкоректно.

Для створення логічності в грі нам знадобиться два списки: в одному з них зберігатимуться координати всіх перешкод ігрового поля, а в другому - всі можливі координати нашого поля, окрім нулів. Для цього записуємо:

```
self.barrier = []
self.field = []
```

Далі створимо метод, який по запуску гри заповнить ці списки координатами: def init_field(self). Даний метод схожий на попередній create_image: так само створюємо змінні x та y зі значенням 1 и цикл for:

```
def init_field(self):
  x = 1
  y = 1
  for i in self.level:
    if i == 0:
      self.barrier.append([x, y])
    elif i == 1 and y != 12:
      self.field.append([x, y])
    x += 11
    if x == 441:
      y += 11
      x = 1
```

Але нам потрібно виключити потрапляння в верхні рядки, тому що ми їх в майбутньому будемо використовувати для інших цілей. Тому ми додали до умови y != 12. І після кожного проходу циклу ми додаємо до координати x 11 і якщо x=441, тобто доходить до краю екрану, то ми додаємо 11 до координати y і обнуляємо координату x. Додаємо метод в основний цикл гри в файлі main.

Далі створюємо новий файл з назвою food, де буде знаходитися клас їжі, котру буде поїдати наша змія. Окрім бібліотеки PyGame імпортуємо ще і Random і створюємо клас з конструктором, в якому запишемо список, де

будуть зберігатися координати їжі self.food_position = []. Далі створимо метод, котрий буде видавати нам координати і присвоюватиме їх цьому списку:

```
def get_food_position(self, gui):
    self.food_position = random.choice(gui.field)
```

Цей метод видає випадкові координати для їжі змійки, які він бере зі списку field, який був створений вище.

Створимо метод для відмальовки їжі (квадратик помаранчевогокольору) draw_food:

```
def draw_food(self, window):
    pygame.draw.rect(window, pygame.Color("Orange"),
pygame.Rect(self.food_position[0],  self.food_position[1], 10, 10))
```

Імпортуємо Food у файл main і вказуємо методи get_food_position і draw_food, щоб отримувати координати їжі одразу на початку гри.

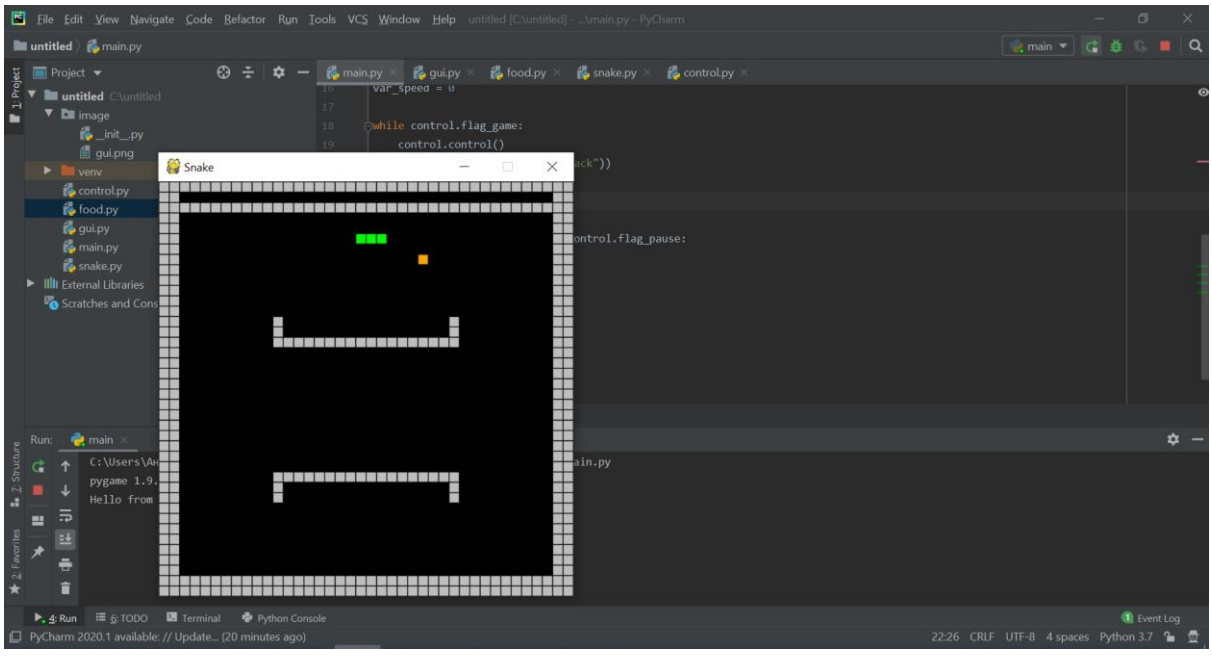


Рис. 4.11 - Відмальовка їжі

Тепер потрібно реалізувати поїдання змією квадратиків їжі. Створюємо в файлі snake новий метод eat і вказуємо умову: якщо голова змії знаходиться в позиції координат їжі, то до тіла змії додається один сегмент. Текст коду виглядає так:

```
def eat(self, food, gui):
  if self.head == food.food_position:
    self.body.append(food.food_position)
    food.get_food_position(gui)
    gui.get_new_indicator()
```

Додаємо метод в main і перевіряємо програму, переконуємося в тому, що все працює коректно.

Рядок згори ми лишили для позначення індикатору, який буде заповнюватися по мірі поїдання змійкою квадратів. Тобто гру пройдено, коли наш індикатор заповнений повністю і програно, якщо довжина змії менша за 3 сегменти. Реалізуємо метод, при якому змія, зіткнувшись з перешкодою або зі своїм тілом буде втрачати один сегмент свого тіла.

Створюємо ще один список indicator, котрий зберігатиме координати нашого індикатору. Створюємо також метод get_new_indicator, який заповнятиме список координатами індикатора, з функцією append.

Запишемо метод відмальовки індикатора draw_indicator з циклом for:

```
def draw_indicator(self, window):
  for i in self.indicator:
    pygame.draw.rect(window, pygame.Color("Orange"), pygame.Rect(i[0], i[1], 10, 10))
```

Цей метод додамо в файл snake та main, запускаємо програму і бачимо, що вона відпрацьовує: змія подовжується і шкала індикатору заповнюється помаранчевими квадратиками.

Тепер реалізуємо метод, який віднімає сегменти змії при зіткненні з перешкодою і своїм тілом:

```
def check_barrier(self, gui):
    if self.head in gui.barrier:
        self.body.pop()
    if self.head in self.body[1:]:
        self.body.pop()
```

Готовий метод перевіряє, чи не врізалася кудись змійка, метод pop віднімає сегмент в такому випадку. Цей метод вказуємо в main і перевіряємо програму на цьому етапі: змійка скорочується, але нам ще потрібно віднімати сегмент індикатора одночасно з сегментом тіла змії. Тому доповнимо наш метод рядком gui.indicator.pop().

На даному етапі розробки гри останніми кроками є додавання програшного екрану й екрану перемоги: якщо індикатор заповниться - перемога, а якщо змія коротша за три - програш. Для цього нам потрібно два готових зображення. Щоб вони вписувалися в процес гри краще, було обрано png зображення без фону.

Рис. 4.12 - Зображення перемоги і програшу

Завантажимо ці два зображення так само, як ми робили із зображенням карти гри на початку розробки, попередньо додавши їх і папку image:

```
self.win = pygame.image.load("image/win.png")
self.lose = pygame.image.load("image/lose.png")
```

Тепер створимо прапорець “GAME”. Нам потрібно зупинити процес гри, коли буде показана картинка кінця гри, тому створимо додаткову умови. Для початку створимо методи, що створюватимуть екран кінця гри для виграшу та програшу:

```
def draw_win(self, window):
  window.blit(self.win, (23, 100))
def draw_lose(self, window):
  window.blit(self.lose, (23, 100))
```

Ці методи вкажемо в main:

```
if gui.game == "GAME":
  snake.draw_snake(window)
  food.draw_food(window)
elif gui.game == "WIN":
  gui.draw_win(window)
elif gui.game == "LOSE":
  gui.draw_lose(window)
```

Створимо останній метод для умови на початку гри, який перевіряє, виграв гравець ци програв, за допомогою індикатора:

```
def check_win_lose(self):
    if len(self.indicator) == 0:
        self.game = "LOSE"
    elif len(self.indicator) == 37:
        self.game = "WIN"
```

Рис. 4.13 - Екран після проходження гри

Рис. 4.14 - Екран програшу

Вставляємо метод на початок циклу і перевіряємо. Як бачимо, гра працює, повністю налагоджена і готова як цілісний програмний продукт.

4.2. Тестування комп’ютерної гри

Тестування програмного забезпечення — це метод перевірки, чи відповідає фактичний програмний продукт очікуваним вимогам, і переконатися, що програмний продукт не містить дефектів. Він включає виконання програмних/системних компонентів з використанням ручних або автоматизованих інструментів для оцінки однієї або кількох властивостей, що цікавлять. Метою тестування програмного забезпечення є виявлення помилок, прогалин або відсутніх вимог на відміну від реальних вимог.

Деякі вважають за краще говорити визначення тестування програмного забезпечення як тестування білої скриньки та чорної скриньки. Простіше кажучи, тестування програмного забезпечення означає перевірку програми, що тестується.

Тестування програмного забезпечення важливе, оскільки якщо в програмному забезпеченні є помилки або помилки, їх можна виявити завчасно та вирішити до доставки програмного продукту. Правильно перевірений програмний продукт забезпечує надійність, безпеку та високу продуктивність, що в подальшому призводить до економії часу, економічної ефективності та задоволення клієнтів.

Тестування важливе, оскільки помилки програмного забезпечення можуть бути дорогими або навіть небезпечними. Помилки програмного забезпечення потенційно можуть призвести до грошових і людських втрат.

Висновки до розділу 4

Протягом роботи над розділом було вирішено використовувати мову програмування Python та середовище розробки PyCharm., яке дало змогу

впевненіше почувати себе в написанні програмного продукту. PyCharm також дає можливість швидше й точніше орієнтуватися в коді, скоріше виявити недоліки коду. Він постійно дає підказки, має неперевершену функцію автозаповнення та складає поради щодо наступних кроків написання коду.

В ході програмування було використано додаткову бібліотеку, що обов’язкова при створенні ігор на Python. Ця бібліотека PyGame завантажується вручну й імпортується в кожен окремий файл. Саме бібліотека PyGame дає ширший вибір функцій та можливостей, які дозволяють створити цілісну гру. З нею можна отримати повністю продумане середовище для розробки своїх ігор.

Програмний продукт було ретельно протестовано, гра відповідає початковим вимогам, має простий юзер-френдлі інтерфейс та не вимагає підключення до інтернету.

ВИСНОВКИ

В дипломній роботі розв’язано актуальну прикладну задачу – розроблений ігровий застосунок в жанрі аркада. Під час виконання кваліфікаційної роботи також був проведений аналіз предметної області.

В аналітичній частині були описані вимоги до розробки ігрового застосунку і обґрунтований вибір програмного забезпечення для реалізації завдання. Завдяки аналізу засобів реалізації було обрано певний набір інструментарію для виконання проекту, а саме – мова програмування, середовище розробки.

Проведено аналіз предметної сфери розробки ігрових за стосунків. Було з’ясовано значення поняття «аркада», визначення наведеного терміну допомогло з’ясувати його значення та актуальність.

Розроблено технічне завдання для ігрового застосунку, що проєктується. Було визначено основних дійових осіб системи та основні вимоги до веб-застосунку. Було наведено сценарії використання системи, наведено діаграми прецендентів та станів, які визначили та надали свого роду структурності системі, показали взаємодію класів, їх атрибутів та взаємозв’язок взагалі.

Визначено поняття ігрового інтерфейсу.

Проведено реалізацію та власне тестування програмного забезпечення.

Завдання, які були поставленні в ході виконання кваліфікаційної роботи бакалавра, були виконані у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кольцов Д.М. Python: Создаем программы и игры. Просто о сложном. 2017. С. 34-66
2. Connor A.M., Gavin, J. Reinventing the arcade: Computer game mediated play spaces for physical interaction. EAI Endorsed Transactions on Creative Technologies. 2015. С. 6-28
3. Use case diagram. URL: https://www.techtarget.com/whatis/definition/use-case-diagram (дата звернення: 13.05.2022)
4. Basic stages of web development. URL: https://web-systems.solutions/blog/veb-rozrobka-etapy-i-standarty/ (дата звернення: 14.05.2022)
5. Москаль В. Р. Метод проектування веб-застосунків. Кваліфікаційна робота магістра. Харків: Харківський національний університет радіоелектроніки, 2021. 62 с.
6. Метиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. 2017. С. 96-108
7. Pycharm. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 23.05.22)
8. ОП Python. URL: https://ru.coursera.org/learn/python-osnovy-programmirovaniya (26.05.20)
9. Створення ігор на Python 3 та Pygame. URL: https://habr.com/ru/post/347138/ (23.05.20)
10. Лутц М. Вивчаєио Python. Диалектика-Вильямс, 2020. 832 с.
11. Ігри та програмування на Python. URL: https://otus.ru/journal/igrovoj-kontent-i-piton/ (23.05.20)
12. Розробка ігор на Python з PyGame. URL: https://itproger.com/course/pygame (24.05.20)
13. PyGame. URL: https://www.pygame.org/news (24.05.20)

14. Основи PyGame. URL: https://python_course.readthedocs.io/projects/elementary/en/latest/lessons/18-pygame.html (24.05.20)
15. Основи програмування ігор. URL: https://hsbi.hse.ru/articles/yazyki-programmirovaniya-dlya-sozdaniya-igr/ (24.05.20)
16. Москаль В. Р. Метод проектування веб-застосунків. 2021. С. 14-15
17. Python data analysis library. URL: https://pandas.pydata.org/about/ (24.05.20)
18. Тестування програмного забезпечення на Python. URL: https://www.softwaretestinghelp.com/python-docstring-tutorial/ (26.05.20)
19. Python docstrings. URL: https://www.geeksforgeeks.org/python-docstrings/ (26.05.20)
20. Reenskaug T., Coplien J. O. The DCI Architecture: A new vision of object-oriented programming. Artima Developer. 2009.

ДОДАТКИ

Додаток А
main.py

```
import pygame
from control import Control
from snake import Snake
from gui import Gui
from food import Food

pygame.init()
window = pygame.display.set_mode((441, 441))
pygame.display.set_caption("Snake")
control = Control()
snake = Snake()
gui = Gui()
food = Food()
gui.init_field()
food.get_food_position(gui)
var_speed = 0

while control.flag_game:
gui.check_win_lose()
control.control()
window.fill(pygame.Color("Black"))
if gui.game == "GAME":
snake.draw_snake(window)
food.draw_food(window)
elif gui.game == "WIN":
gui.draw_win(window)
elif gui.game == "LOSE":
gui.draw_lose(window)
gui.draw_indicator(window)
gui.draw_level(window)
if var_speed % 50 == 0 and control.flag_pause and gui.game == "GAME":
snake.move(control)
snake.check_barrier(gui)
```

```
snake.eat(food, gui)
snake.check_end_window()
snake.animation()
var_speed += 1
pygame.display.flip()
```

Додаток Б

gui.py

```
import pygame


class Gui:
def __init__(self):
self.level = [

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
0, 0, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
0, 0, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,
```

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

, 1, 1, 1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

```
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1,
1, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
0, 0, 0, 0

]

self.gui_image = pygame.image.load("image/gui.png")
self.win = pygame.image.load("image/win.png")
self.lose = pygame.image.load("image/lose.png")
self.barrier = []
self.field = []
self.indicator = [[12, 12]]
self.game = "GAME"

def create_image(self):
screen = pygame.Surface((441, 441), pygame.SRCALPHA, 32)
x = 1
y = 1
for i in self.level:
if i == 0:
pygame.draw.rect(screen, pygame.Color("Grey"), pygame.Rect(x, y, 10, 10))
x += 11
if x == 441:
y += 11
x = 1
pygame.image.save(screen, "image/gui.png")

def draw_level(self, window):
```

```
window.blit(self.gui_image, (0, 0))

def init_field(self):
x = 1
y = 1
for i in self.level:
if i == 0:
self.barrier.append([x, y])
elif i == 1 and y != 12:
self.field.append([x, y])
x += 11
if x == 441:
y += 11
x = 1

def get_new_indicator(self):
self.indicator.append([self.indicator[-1][0] + 11, 12])

def draw_indicator(self, window):
for i in self.indicator:
pygame.draw.rect(window, pygame.Color("Orange"), pygame.Rect(i[0], i[1], 10, 10))

def draw_win(self, window):
window.blit(self.win, (23, 100))

def draw_lose(self, window):
window.blit(self.lose, (23, 100))

def check_win_lose(self):
if len(self.indicator) == 0:
self.game = "LOSE"
elif len(self.indicator) == 37:
self.game = "WIN"
```

Додаток В
food.py

```
import pygame
import random
```

```
class Food:
def __init__(self):
self.food_position = []

def get_food_position(self, gui):
self.food_position = random.choice(gui.field)

def draw_food(self, window):
pygame.draw.rect(window, pygame.Color("Orange"),
pygame.Rect(self.food_position[0], self.food_position[1], 10, 10))
```

Додаток Г
snake.py

```
import pygame


class Snake:
def __init__(self):

self.head = [45, 45]
self.body = [[45, 45], [34, 45], [23, 45]]

def move(self, control):
if control.flag_direction == "RIGHT":
self.head[0] += 11
elif control.flag_direction == "LEFT":
self.head[0] -= 11
elif control.flag_direction == "UP":
self.head[1] -= 11
elif control.flag_direction == "DOWN":
self.head[1] += 11

def animation(self):
```

```
self.body.insert(0, list(self.head))
self.body.pop()

def draw_snake(self, window):
for segment in self.body:
pygame.draw.rect(window, pygame.Color("Green"), pygame.Rect(segment[0],
segment[1], 10, 10))

def check_end_window(self):
if self.head[0] == 419:
self.head[0] = 23
elif self.head[0] == 12:
self.head[0] = 419
elif self.head[1] == 23:
self.head[1] = 419
elif self.head[1] == 419:
self.head[1] = 34

def eat(self, food, gui):
if self.head == food.food_position:
self.body.append(food.food_position)
food.get_food_position(gui)
gui.get_new_indicator()

def check_barrier(self, gui):
if self.head in gui.barrier:
self.body.pop()
gui.indicator.pop()
if self.head in self.body[1:]:
self.body.pop()
gui.indicator.pop()
```

Додаток Д
control.py

```
import pygame
from pygame.locals import *


class Control:
def __init__(self):
self.flag_game = True
self.flag_direction = "RIGHT"
self.flag_pause = "TRUE"

def control(self):
for event in pygame.event.get():
if event.type == pygame.QUIT:
self.flag_game = False
elif event.type == pygame.KEYDOWN:
if event.key == pygame.K_RIGHT and self.flag_direction != "LEFT":
self.flag_direction = "RIGHT"
elif event.key == pygame.K_LEFT and self.flag_direction != "RIGHT":
self.flag_direction = "LEFT"
elif event.key == pygame.K_UP and self.flag_direction != "DOWN":
self.flag_direction = "UP"
elif event.key == pygame.K_DOWN and self.flag_direction != "UP":
self.flag_direction = "DOWN"
elif event.key == pygame.K_ESCAPE:
self.flag_game = False
elif event.key == pygame.K_SPACE:
if self.flag_pause:
self.flag_pause = False
elif self.flag_pause == False:
self.flag_pause = True
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ІНФОРМАЦІЙНИЙ ПОРТАЛ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

СПЕЦІАЛЬНА ЧАСТИНА З ОХОРОНИ ПРАЦІ

ПИТАННЯ ОХОРОНИ ПРАЦІ У ДІЯЛЬНОСТІ РОЗРОБНИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.01 − 408.21810816

Студент _____Г.Ю.Круковська

підпис

«___» _____ 2022 р.

Консультант канд. техн. наук, доцент

_____А. О. Алексєєва

підпис

«___» _____ 2022 р.

Миколаїв – 2022

ЗМІСТ

ВСТУП

Сучасну цивілізацію характеризує високий рівень використання технологічних засобів, призначених для задоволення життєвих потреб людини. З кожним роком технологічні засоби стають все більш енергоємними та автоматизованими. Людина, однак, залишається найважливішим компонентом.

Існування сучасних новітніх технологій засноване на діяльності людини. Не менше половини життя людина проводить у виробничій сфері. Найнебезпечніший час для людини – це час, коли вона зайнята трудовою діяльністю. Оскільки така діяльність насичене рядом енергоємних технологічних методів, сучасне виробництво характеризується найвищим рівнем небезпеки. Дослідження умов праці, травматизму, нещасних випадків, професійних захворювань виявляє, що першочерговими причинами є недотримання правил техніки безпеки, а також незнання техногенних ризиків і засобів захисту від них. Крім того, за багатьох обставин людський аспект є основним джерелом небезпеки. І оскільки такі випадки стають все частіше, питання безпеки на робочому місці стає все більш важливим. Саме в цьому полягає **актуальність** розділу з охорони праці.

Метою спеціальної частини є аналіз умов праці та питань безпеки діяльності розробника програмного забезпечення.

1. ОРГАНІЗАЦІЙНО-ПРАВОВІ ЗАСАДИ ОХОРОНИ ПРАЦІ

Відповідно до Закону України "Про охорону праці" охорона праці – це система правових, соціально-економічних, організаційно технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на збереження життя, здоров'я й працездатності людини в процесі трудової діяльності [1].

Вимога щодо впровадження заходів з охорони праці передбачається, зокрема, статтею 13 ЗУ «Про охорону праці». У відповідності з цим законом, кожна компанія, в рамках якої реалізуються трудові відносини, зобов’язана вжити всіх необхідних заходів з охорони праці та розробити відповідні документи: положення про охорону праці, інструкції з охорони праці по кожній з професій та в цілому, накази з охорони праці, журнали інструктажу, реєстрацій та інше.

Державна політика в галузі охорони праці визначається Верховною Радою України відповідно до Конституції України і спрямована на створення належних, безпечних і здорових умов праці, запобігання нещасним випадкам та професійним захворюванням [2].

Законодавство про охорону праці складається з Закону, Кодексу законів про працю України, Закону України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності" та прийнятих відповідно до них нормативно-правових актів.

Дія закону про охорону праці поширюється на всіх юридичних та фізичних осіб, які відповідно до законодавства використовують найману працю, та на всіх працюючих.

Державна політика в галузі охорони праці базується на принципах [3]:

- пріоритету життя і здоров'я працівників, повної відповідальності роботодавця за створення належних, безпечних і здорових умов праці;
- підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції, а також сприяння підприємствам у створенні безпечних та нешкідливих умов праці;
- комплексного розв'язання завдань охорони праці на основі загальнодержавної, галузевих, регіональних програм з цього питання та з урахуванням інших напрямів економічної і соціальної політики, досягнень в галузі науки і техніки та охорони довкілля;
- соціального захисту працівників, повного відшкодування шкоди особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;
- встановлення єдиних вимог з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності та видів діяльності;
- адаптації трудових процесів до можливостей працівника з урахуванням його здоров'я та психологічного стану;
- використання економічних методів управління охороною праці, участі держави у фінансуванні заходів щодо охорони праці, залучення добровільних внесків та інших надходжень на ці цілі, отримання яких не суперечить законодавству;
- інформування населення, проведення навчання, професійної підготовки і підвищення кваліфікації працівників з питань охорони праці;
- забезпечення координації діяльності органів державної влади, установ, організацій, об'єднань громадян, що розв'язують проблеми охорони

здоров'я, гігієни та безпеки праці, а також співробітництва і проведення консультацій між роботодавцями та працівниками (їх представниками), між усіма соціальними групами під час прийняття рішень з охорони праці на місцевому та державному рівнях;

– використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці на основі міжнародного співробітництва.

Державне управління охороною праці в Україні здійснюють:

– Кабінет Міністрів України;
– центральний орган виконавчої влади, що реалізує державну політику у сфері охорони праці;
– міністерства та інші центральні органи виконавчої влади.

За недотримання вимог законодавства України щодо охорони праці, посадові особи ІТ компанії можуть бути притягнуті до адміністративної відповідальності, що полягає у накладенні штрафу в розмірі 2 % від місячного фонду заробітної плати компанії.

У разі, коли в результаті порушення умов охорони праці постраждав працівник, посадові особи ІТ компанії можуть бути притягнені до кримінальної відповідальності, в тому числі, у вигляді позбавлення волі.

2. ВИМОГИ БЕЗПЕКИ ПРАЦІ РОЗРОБНИКА ПЗ

Зараз комп’ютерні технології широко використовуються в багатьох сферах життя людини. Робота з комп’ютером піддає людину впливу різноманітних небезпечних і шкідливих виробничих змінних, включаючи електромагнітні поля, інфрачервоне та іонізуюче випромінювання, шум і вібрацію, статичну електрику тощо.

При роботі з клавіатурою комп’ютера оператори відчувають колосальне психічне та нервово-емоційне напруження, а також високу інтенсивність зорової роботи та відносно велике навантаження на м’язи рук. Розумна конструкція та організація частин робочого місця має вирішальне значення для збереження ідеального робочого положення програміста.

2.1 Організація робочого місця

Організація місця інженера програмного забезпечення має відповідати таким вимогам:

1. Стіни приміщень для роботи з ПК мають бути пофарбовані чи обклеєні шпалерами пастельних кольорів з коефіцієнтом відбиття 40 - 60 %. У випадках, коли такі приміщення зорієнтовані на південь, вікна повинні обладнуватися сонцезахисними пристроями (жалюзі, штори тощо).
2. Для освітлення приміщень з ПК необхідно використовувати люмінесцентні світильники. Освітленість робочих місць у горизонтальній площині на висоті 0,8 м від підлоги повинна бути не менше 400 лк. Вертикальна освітленість у площині екрану не більше 300 лк.
3. У приміщеннях для роботи з ПК необхідно проводити щоденне вологе прибирання та регулярне провітрювання протягом робочого дня. Вида-

лення пилу з екрану необхідно проводити не рідше одного разу на день.

4. Приміщення, в якому проводилася робота, має загальну площу 18 м^2, висоту стелі 2,5 м, у довжину 4,5 м. У приміщенні знаходиться 2 робочих місць з ПК. Кожне робоче місце обладнане робочім столом площею 1,2 м^2, стільцем та персональним комп'ютером, що складається з монітора, системного блоку, клавіатури та миші. Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м , а об'єм не менший за 20м, тобто площі та об'єму даного приміщення вистачає для розташування 2 робочих місць операторів ПК [5].

Аналіз умов праці показує, що у приміщенні на програміста можуть негативно впливати наступні фізичні та психофізіологічні фактори:

- підвищена або знижена температура повітря робочої зони;
- підвищена або знижена вологість повітря;
- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- підвищена іонізація повітря;
- підвищений рівень електромагнітних випромінювань;
- нервово-психічні перевантаження (розумова перенапруга, перенапруга аналізаторів);
- фізичні перевантаження (одноманітна поза викликає статичну втому).

2.2 Освітленість робочого місця

Промислове освітлення повинно [4]:

- бути достатнім (задовольняти характер зорової роботи);
- бути однорідним і послідовним на робочих поверхнях;

- уникати відблисків від джерел світла та відбивних поверхонь;
- уникати різких тіней на робочому місці;
- бути в безпеці;
- бути надійним, простим в експлуатації, доступним і красивим.

Нормованим параметром природного освітлення згідно ДБН В.2.5-28-2006 являється коефіцієнт природного освітлення (КПО) [6]. КПО встановлюється в залежності від розряду виконуваних зорових робіт. Робота програміста відноситься до робіт середньої точності (IУ розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5-1,0мм), для яких при використанні бокового освітлення КПО=1,5%. Для штучного освітлення нормованим параметром виступає $E_{мін}$ -мінімальний рівень освітленості, та $К_п$ - коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%. Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряду зорових робіт вона складає 300-500 лк (люкс).

Нормовані технічні показники:

1. Середній рівень освітленості в кімнаті повинен бути в межах 800-100 лк, але зі значенням не менше 200 лк в безпосередній близькості від робочого місця.
2. Відблиски (або відблиски) слід звести до мінімуму. Якщо в кімнаті є люмінесцентні лампи або прожектори, які відбиваються на екрані монітора, їх необхідно міняти з різними джерелами освітлення. При спробі підтримувати наявні ліхтарі за допомогою картриджів типу E (E27, E14), рекомендується з самого початку замінити вольфрамові лампи на світлодіодні SMD-моделі з матовим розсіювачем.
3. Ідеальному розподілу освітлення буде сприяти симетричне

розташування подібних пристроїв, а також керування їх потужністю та колірною температурою. Якщо рівномірність світла становить 0,4 одиниці в радіусі півметра від людини і 0,1 від нього, в типовому офісі це вважається нормальним. Також важливо звернути увагу на пропорції між значеннями в безпосередній близькості від людини та на периферії: неактивно задіяні області зазвичай освітлені приблизно на 45% від світла в робочій зоні.

4. Для всіх професій індекс передачі кольору повинен бути високим. Переважна більшість світлодіодних ліхтарів створена зі спектром, який спочатку очікується приблизно 80-90 Ra.
5. Відсутність пульсацій у світловому спектрі є останньою важливою ознакою. Якщо появі стробоскопічного ефекту в офісах загрожують лише перепади настрою та тимчасове зниження працездатності, це явище набагато більш проблематично на підприємствах з великою кількістю рухомих частин. Під час роботи з вібраційними системами працівники, які піддаються впливу мерехтливого світла, можуть втратити контроль, втратити свідомість або погано контролювати свої рухи.

У табл. 2.1 можна побачити норми освітленості офісних приміщень.

Таблиця 2.1 – Норми освітленості офісних приміщень

Тип офісу	**Середня освітленість, люкс**	**Коефіцієнт нерівномірності освітленості ($E_{мін}/E_{ср}$)**	**Клас приміщення**
Невеликий офіс	500	0,8	A-B
Великий відкритий	750	0,8	A-B

офіс			
Конференц-зала	300	0,5	A-B
Приймальня	300-500	0,5	A-B
Технічне приміщення	100-200	0,5	C-D-E

2.3 Мікроклімат робочої зони

Мікрокліматичні умови промислових будівель визначають такі показники [7]:

- температура повітря;
- відносна вологість;
- швидкість руху повітря;
- інтенсивність теплового (інфрачервоного) випромінювання;
- температура поверхні.

Мікрокліматичні умови поділяються на ідеальні або прийнятні залежно від ступеня впливу на тепловий стан людини.

Оптимальні (табл. 2.2) та прийнятні мікрокліматичні умови з урахуванням тяжкості завдання та пори року. Для одночасного виконання робіт у робочій зоні різної тяжкості з урахуванням найбільшої групи персоналу необхідно виготовляти показники мікроклімату.

Таблиця 2.2 – Оптимальні величини температури, відносної вологості та швидкості руху повітря в робочій зоні виробничих приміщень

Період року	**Категорія робіт**	**Температура повітря, ℃**	**Відносна вологість,%**	**Швидкість руху, м/сек.**
Холодний період	Легка Іа	22-24	60-40	0,1
	Легка Іб	21-23		0,1

року	Середньої важкості IIа	19-21		0,2
	Середньої важкості IIб	17-19		0,2
	Важка III	16-18		0,3

Теплий період року	Легка Iа	23-25	60-40	0,1
	Легка Iб	22-24		0,2
	Середньої важкості IIа	21-23		0,3
	Середньої важкості IIб	20-22		0,3
	Важка III	18-20		0,4

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), технологічного обладнання (екрани тощо), зовнішніх поверхонь технологічного устаткування, огороджуючих конструкцій не повинна виходити більш ніж на 2°С за межі оптимальних величин температури повітря для даної категорії робіт, вказаних в табл. 2.1.

2.4 Рівень випромінювання

У своїй роботі розробник ПЗ постійно користується ПК (персональним комп’ютером). І звісно, тут слід враховувати допустимий рівень випромінювання.

Електростатичні та електромагнітні поля створюють комп’ютери.

Магнітні поля можна регулювати в двох діапазонах частот: від 5 Гц до 2 кГц і від 2 до 400 кГц. Заміри користувачів стаціонарних і портативних персональних комп’ютерів проводяться на їх робочих місцях. Напруженість електричного та магнітного поля, а також напруженість електростатичного поля регулюються.

Серед незадовільних результатів вимірювань електромагнітного поля персональних комп'ютерів найбільша напруженість спостерігається в електричному полі, рідше в магнітному. Як показав досвід, на електромагнітне середовище на робочому місці значно впливає наявність заземлення в приміщенні при використанні сертифікованих моніторів. Важливо міняти монітор, якщо приміщення має хороше заземлення і напруженість електричного поля 2-400 кГц перевищує допустимі пороги. Напруженість електричного поля збільшується на обох частотах, якщо в приміщенні немає заземлення.

Важливо мати якісне заземлення комп’ютера, щоб **усунути згубний вплив** цих елементів на організм людини. Не варто самостійно заземлювати обладнання; замість цього скористайтеся професіоналом. Провід заземлення не можна підключати до громовідводу, газопроводу або опалювальних труб. Все це може призвести до поломки комп’ютера та інших несприятливих наслідків.

2.5 Рівень шуму

Шум є одним з найнебезпечніших аспектів сучасної культури. Промисловий шум – це безпорядкова сукупність шумів різної гучності та частоти, які проникають у повітря і мають безпосередній вплив на продуктивність.

Вимірювання шуму на робочих місцях здійснюється шумовимірювачами та аналізаторами спектра шуму. Рівень шуму на робочих місцях потрібно контролювати не менше одного разу на рік. В умовах виробництва, як правило, мають місце шуми різної інтенсивності і спектри, які виникають унаслідок дії

різноманітних механізмів, агрегатів та інших пристроїв.

Класи умов праці залежно від рівня шуму поділяються на допустимі, які відповідають ГДР згідно з Державними санітарними нормами ДСН 3.3.6 037-99, шкідливі та небезпечні [8]. На табл. 2.3 зображено допустимі рівні звукового тиску.

Таблиця 2.3 – Допустимі рівні звукового тиску

Приміщення	**Рівні звукового тиску (дБ) в смугах із середньогеометричними частотами (Гц)**								**Рівень звуку та еквівалентні рівні звуку ДБА**
	65	125	250	500	1000	2000	4000	8000	
Приміщення програмістів обчислювальних машин	71	61	54	49	45	42	40	38	50

Від шуму страждає травна та кровоносна системи, а також серцево-судинна система. Постійний фоновий шум до 70 дБ порушує роботу ендокринної та неврологічної систем; до 90 дБ впливає на слух; і до 120 дБ викликає фізичний дискомфорт, який може бути нестерпним. Шум не тільки негативно впливає на здоров’я людини, але і впливає на продуктивність на 10-15%. Боротьба з нею важлива не лише з санітарно-гігієнічних міркувань, а й з технологічних та економічних міркувань.

Рекомендовані такі діапазони шуму для приміщень різних призначень: для сну та відпочинку – 30-40 дБ, для розумової праці – 45-55, для робітників цехів, гаражів, магазинів – 56-70, у службових приміщеннях касового вузла банку - 60, виробничих приміщеннях касового вузла – 75 дБ.

3. ПОЖЕЖНА БЕЗПЕКА

Приміщення з ЕОМ повинні бути оснащені системою автоматичної пожежної сигналізації відповідно до вимог Переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 $м^2$ площі приміщення з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог Правил пожежної безпеки в Україні [9]. В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі.

Основними причинами виникнення пожежі в кабінеті інженера-програміста можуть бути:

- замикання або загоряння електрообладнання, що експлуатується (монітор, принтер, клавіатура тощо);
- додаткові опалювальні прилади;
- система штучного освітлення;
- загоряння паперових документів.

Пожежа в лабораторії, може привести до дуже несприятливих наслідків (втрата коштовної інформації, псування майна, загибель людей і т.д.), тому необхідно:

- виявити й усунути всі причини виникнення пожежі;
- розробити план заходів для ліквідації пожежі в будинку; план евакуації людей з будинку.

Для профілактики пожежі надзвичайно важлива правильна оцінка пожежонебезпеки будинку, визначення небезпечних факторів і обґрунтування способів і засобів пожежопопередження і захисту.

Одне з умов забезпечення пожежобезпеки – ліквідація можливих джерел запалення. У лабораторії джерелами запалення можуть бути:

- несправне електроустаткування, несправності в електропроводці, електричних розетках і вимикачах. Для виключення виникнення пожежі з цих причин необхідно вчасно виявляти й усувати несправності, проводити плановий огляд і вчасно усувати всі несправності;
- несправні електроприлади. Необхідні міри для виключення пожежі містять у собі своєчасний ремонт електроприладів, якісне виправлення поломок, не використання несправних електроприладів;
- обігрівання приміщення електронагрівальними приладами з відкритими нагрівальними елементами. З метою профілактики пожежі пропоную не використовувати відкриті обігрівальні прилади в приміщенні лабораторії;
- коротке замикання в електропроводці. З метою зменшення імовірності виникнення пожежі внаслідок короткого замикання необхідно, щоб електропроводка була схованою;
- влучення в будинок блискавки. У літній період під час грози можливе влучення блискавки внаслідок чого можливий пожежа. Щоб уникнути цього я рекомендую установити на даху будинку блискавковідвід;
- недотримання мір пожежної безпеки і паління в приміщенні також може спричинити пожежу. Для усунення загоряння в результаті паління в приміщенні лабораторії пропоную категорично заборонити паління, а дозволити тільки в строго відведеному для цього місці.

З метою запобігання пожежі пропоную проводити з інженерами, що працюють у лабораторії, протипожежний інструктаж, на якому ознайомити

працівників із правилами протипожежної безпеки, а також навчити використанню первинних засобів пожежогасіння.

ВИСНОВКИ

Під час виконання спеціального розділу з охорони праці було проаналізовано умови, у яких безпечно працювати розробнику програмного забезпечення, тобто виконувати свою безпосередню діяльність. Отже, поставлену ціль було реалізовано.

Було визначено, що основною метою охорони праці є створення безпечних умов праці на кожному робочому місці, забезпечення безпечної експлуатації обладнання, зменшення або повна нейтралізації впливу шкідливих і небезпечних виробничих факторів на організм людини, зниження рівня виробничого травматизму та професійних захворювань. результат.

Для безпечної діяльності робоче місце повинно бути обладнано належним чином, а саме: стіни приміщень для роботи з ПК мають бути пофарбовані чи обклеєні шпалерами пастельних кольорів з коефіцієнтом відбиття 40 - 60 %, для освітлення приміщень з ПК необхідно використовувати люмінесцентні світильники, у приміщеннях для роботи з ПК необхідно проводити щоденне вологе прибирання та регулярне провітрювання протягом робочого дня.

Також важливою складовою є освітленість робочого місця. Для невеликого приміщення воно має бути 500 люкс, для більшого – 750 люкс. Мікрокліматичні умови безпосередньо впливають на стан розробника, тож за цим особливо треба слідкувати.

Звісно, великий вплив на здоров’я працівника здійснює електромагнітне випромінювання, отже, необхідно знати, як захистити людину від нього. І мають бути дотримані рівня шуму та правила пожежної безпеки.

Завдяки виконанню цього розділу було висвітлено основні вимоги до організації охорони праці, які забезпечать безпеку та захистять життя інженера програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про охорону праці. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 01.06.2022)
2. Конституція України. URL: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96-%D0%B2%D1%80#Text (дата звернення: 01.06.2022)
3. Кодекс законів про працю України. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 01.06.2022)
4. Ударцева Т. Є. Розрахунок та гігієнічна оцінка умов освітлення робочого місця. 2011.
5. Обґрунтування заходів з покращення умов охорони праці. URL: https://lektsii.org/4-601.html (дата звернення: 01.06.2022)
6. ПРИРОДНЕ І ШТУЧНЕ ОСВІТЛЕННЯ. ДБН В.2.5-28-2006. URL: http://kbu.org.ua/assets/app/documents/dbn2/95.1.%20%D0%94%D0%91%D0%9D%20%D0%92.2.5-28-2006.%20%D0%9F%D1%80%D0%B8%D1%80%D0%BE%D0%B4%D0%BD%D0%B5%20%D1%96%20%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%B5%20%D0%BE%D1%81%D0%B2%D1%96%D1%82%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F.pdf (дата звернення: 01.06.2022)
7. Методичні матеріали та завдання до практичних занять на тему «Мікроклімат робочої зони, оздоровлення повітряного середовища та нормалізація мікроклімату й теплозахисту» з дисципліни "Основи охорони праці"/ Уклад.: Праховнік Н.А., Качинська Н.Ф., Арламов О.Ю., Чикунова Васильєва Н.П. К.:2017. - 30 с.
8. ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку (31668). URL: https://dnaop.com/html/31668/doc-%D0%94%D0%A1%D0%9D%0A%0A3.3.6.037-99 (дата звернення: 01.06.2022)
9. Про затвердження Правил пожежної безпеки в Україні. URL: https://zakon.rada.gov.ua/laws/show/z0252-15#Text (дата звернення: 01.06.2022)