**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

*Факультет комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету )
*Кафедра комп'ютерних систем та мереж*
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

*магістр*
(назва освітнього ступеня)

на тему: *Методи і засоби управління та моніторингу руху об'єктів в ігрових комп'ютерних системах*

Виконав: студент *VI* курсу, групи *СІм-61*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

(підпис) *Цимбалістий В.О.* (прізвище та ініціали)

Керівник (підпис) *Яцишин В.В.* (прізвище та ініціали)

Нормоконтроль (підпис) *Тиш Є.В.* (прізвище та ініціали)

Завідувач кафедри (підпис) *Осухівська Г.М.* (прізвище та ініціали)

Рецензент (підпис) *Готович В.А.* (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
*Осухівська Г.М.*
(підпис) (прізвище та ініціали)
« » 2022 р.

# З А В Д А Н Н Я
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Цимбалістому Віктору Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Методи і засоби управління та моніторингу руху об'єктів в ігрових комп'ютерних системах*

Керівник роботи *Яцишин Василь Володимирович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *6* » *грудня* 2022 року № *4/7-986*

2. Термін подання студентом завершеної роботи *22.12.2022* р.

3. Вихідні дані до роботи *Метод, штучні нейронні мережі, машинне навчання, ігрові комп'ютерні системи.*

4. Зміст роботи (перелік питань, які потрібно розробити)
*Вступ. 1. Аналіз методів та засобів імплементації нейронних мереж в ігрових комп'ютерних системах. 2. Математичне забезпечення ігрової комп'ютерної системи. 3. Апробація ігрової комп'ютерної системи. 4. Охорона праці та безпека життєдіяльності. Висновки.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
*1. Тема кваліфікаційної роботи, актуальність, об'єкт та предмет дослідження.*
*2. Мета, задачі дослідження та наукова новизна.*
*3. Концепція проектованої ігрової комп'ютерної системи.*
*4. Схема нейронної мережі.*
*5. Алгоритм навчання нейронної мережі.*
*6. Алгоритм обчислення добутку двох матриць.*
*7. Алгоритм приведення вектору до матриці.*
*8. Висновки.*

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| *Основи охорони праці,* | *Осухівська Г.М. доцент* | | |
| *Безпека життєдіяльності* | *Клепчик В.М., старший викладач* | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 15.11.2022р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | *Затвердження теми кваліфікаційної роботи* | | |
| 2 | *Аналіз літературних джерел* | *15.11.22-18.11.22* | *Виконано* |
| 3 | *Обгрунтування актуальності дослідження* | *19.11.22* | *Виконано* |
| 4 | *Аналіз предмету дослідження та предметної області* | *20.11.22-23.11.22* | *Виконано* |
| 5 | *Оформлення першого розділу роботи* | *24.11.22-28.11.22* | *Виконано* |
| 6 | *Оформлення другого розділу роботи* | *29.11.22-2.12.22* | *Виконано* |
| 7 | *Оформлення третього розділу роботи* | *3.12.22-10.12.22* | *Виконано* |
| 8 | *Оформлення розділу «Охорона праці та безпека* | | |
| | *життєдіяльності»* | *11.12.22-12.12.22* | *Виконано* |
| 9 | *Нормоконтроль* | *13.12.22* | *Виконано* |
| 10 | *Попередній захист роботи* | 15.12.22 | *Виконано* |
| 11 | *Захист кваліфікаційної роботи* | 22.12.22 | *Виконано* |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ *Цимбалістий Віктор Олександрович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Яцишин Василь Володимирович*
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Методи і засоби управління та моніторингу руху об’єктів в ігрових комп’ютерних системах // Кваліфікаційна робота // Цимбалістий Віктор Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІм-61 // Тернопіль, 2022 // с. – 56, рис. – 26, аркушів А1. – 8, додат. – 2, бібліогр. – 20.

Ключові слова: метод, штучні нейронні мережі, машинне навчання, ігрові комп’ютерні системи

У кваліфікаційній роботі магістра досліджено методи і засоби моніторингу в ігровій комп’ютерній системі.

Проведено аналіз сучасних тенденцій проектування комп’ютерних систем, з використанням найбільш оптимальних інструментів вирішення поставлених задач, за час яких встановлено, що для розробки інтелектуального модуля потрібно роглянути структуру нейронної мережі, яка в подальшому допомогамає провести формалізацію даних на етапі математичного аналізу.

Спроектовано математичну модель нейронної мережі, яка здійснює перехоплення і знищення літаючих об’єктів ігрової комп’ютерної системи.

Реалізовано програмну імплементацію нейронної мережі прямого поширення з методом навчання зворотнього поширення, за допомогою мови JavaScript.

# ABSTRACT

Controlling and monitoring methods and means of the object movements in the game computer systems // Tsymbalistyi Viktor Oleksandrovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Systems and Networks Department, Group SIm-61 // Ternopil, 2022 // p. – 56, fig. – 26, posters A1 – 8, addition. – 2, ref. – 20.

Keywords: method, artificial neural networks, machine learning, game computer systems

In the master's qualification work, the methods and means of monitoring in the gaming computer system were investigated.

An analysis of modern trends in the design of computer systems was carried out, using the most optimal tools for solving the tasks, during which it was established that for the development of an intelligent module it is necessary to consider the structure of a neural network, which in the future will help formalize the data at the stage of mathematical analysis.

A mathematical model of a neural network that intercepts and destroys flying objects of a gaming computer system has been designed.

The software implementation of the forward propagation neural network with the back propagation learning method using the JavaScript language has been implemented.

ЗМІСТ

# ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

REST API – Representational state transfer application programming interface.

SOLID – (мнемонічний акронім) Single responsibility; Open-closed; Liskov substitution; Interface segregation; Dependency inversion.

JVM – Java Virtual Machine.

FFNN – Feed Forward Neural Network

# ВСТУП

На сучасному етапі розвитку інформаційних технологій спостерігаються тенденції щодо впровадження засобів автоматизації для підвищення ефективності і надійності виконання як бізнес, так і технологічних процесів у сферах, які до цього часу слабо піддавались автоматизації. Це в першу чергу пов’язано зі стрімким зростанням та підвищенням продуктивності методів і засобів штучного інтелекту, зокрема машинного навчання. Не менш важливим фактором ефективної імплементації технологій автоматизації є потужний розвиток апаратного і програмного забезпечення, зокрема cloud-технологій, Big Data, Smart-систем та ін. Оскільки використання хмарних ресурсів і сервісів, все ще є доволі дорогим задоволенням, то доцільним є початкове моделювання процесів управління різними пристроями на звичайних машинах з подальшою міграцією у хмару.

Розвиток технологій реалізації безпілотних літальних апаратів на сьогодні є доволі актуальним, як у побутових, так і у військових цілях. Важливим компонентом такої комп’ютерної системи є підсистема управління, яка б в залежності від зовнішніх умов, давала б змогу апарату самостійно приймати рішення щодо вибору оптимальної траєкторії руху, уникнення перешкод і виконання поставленого завдання. Оскільки, самі пристрої на сьогодні, також є недешевими, то доцільно реалізувати модель управління та поведінки літального апарату у вигляді комп’ютерної гри «Астероїди».

У роботі пропонується реалізувати управління астероїдами на основі підходу нейронних мереж засобами мови програмування JavaScript та браузера Google Chrome.Браузер виступає середовищем, в якому працює програмне забезпечення (ПЗ) розроблене на основі JavaScript. Ще однією перевагою такої реалізації є низький поріг входження у тематику щодо інтелектуального управління об’єктами і дозволяє зрозуміти базові принципи функціонування нейронних мереж з практичної точки зору.

**Метою роботи** є дослідження методів і засобів управління та моніторингу руху об’єктів в ігрових комп’ютерних системах.

Для досягнення зазначеної мети у роботі сформульовано наступні **задачі**:

– аналіз наукових праць і практичних рішень щодо реалізації систем управління та моніторингу руху об’єктів в ігрових комп’ютерних системах;

– аналіз існуючих методів і технологій інтелектуального керування об’єктами;

– побудова моделі нейронної мережі та визначення відповідних параметрів її налаштування для забезпечення ефективності процесу управління та моніторингу руху об’єктів;

– обґрунтування методу навчання нейронної мережі при управлінні та моніторингу руху об’єктів;

– побудова архітектури програмного засобу емуляції роботи при управлінні та моніторингу руху об’єкту як комп’ютерної ігрової системи;

– реалізація засобами мови JavaScript моделі нейронної мережі у вигляді комп’ютерної ігрової системи.

**Об'єкт дослідження** – процеси управління та моніторингу руху об’єктів з використанням підходів штучного інтелекту.

**Предмет дослідження** – моделі, методи і засоби побудови та реалізації нейронних мереж для управління рухомими об’єктами.

**Наукова новизна одержаних результатів:**

– уперше запропоновано алгоритм навчання нейронної мережі на основі зворотного поширення помилки та визначено функції активації нейронів, що дало змогу забезпечити ефективність управління та моніторингу руху об’єктів у комп’ютерній ігровій системі.

– уперше спроектовано концепт нейронної мережі для забезпечення виконання математичних операцій над матрицями, що дало змогу визначити її архітектуру за складом шарів і забезпечити оптимальний час навчання нейронної мережі.

**Методи дослідження.** Для виконання задач кваліфікаційної роботи магістра використано наступні методи:

– абстракції та узагальнення – при аналізі підходів до імплементації процесів управління та моніторингу руху об’єктів;

– формалізація і моделювання – при обґрунтуванні і побудові архітектури нейромережі;

– аналіз і проектування – при розробці комп’ютерної гри з елементами штучного інтелекту;

– експеримент – при апробації запропонованих методів і засобів;

**Практичне значення одержаних результатів.** Практичне значення одержаних результатів полягає у реалізації комп’ютерної веб-орієнтованої ігрової системи «Астероїди».

**Публікації. Публікації.** Результати кваліфікаційної роботи апробовані на X науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (8-9 грудня 2022 року) та у міжнародній конференції-воркшопі ITTAP 2022 як тези конференцій.

1. Yatsyshyn V., Pastukh O., Lutskiv A, Tsymbalistyy V., Martsenko N. A Risks management method based on the quality requirements communication method in agile approaches The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems.2022. PP

2. Яцишин В.В., Цимбалістий В.О., Яцишин Вік.В. Комп’ютерні ігри як спосіб моделювання поведінки реальних комп’ютерних систем. Матеріали X науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (8-9 грудня 2022 року). Тернопіль: ТНТУ. 2022. С. 97.

**Структура роботи.** Кваліфікаційна робота містить розрахунково-пояснювальну записку та графічний матеріал. До складу записки входить вступу, 4 розділи, загальні висновки, список використаних джерел і додатки. Обсяг роботи: розрахунково-пояснювальна записка – 86 арк. формату А4, графічна частина – 8 аркушів формату А1.

# РОЗДІЛ 1
# АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ІМПЛЕМЕНТАЦІЇ НЕЙРОННИХ МЕРЕЖ В ІГРОВИХ КОМП'ЮТЕРНИХ СИСТЕМ

## 1.1. Аналіз особливостей ігрової комп'ютерної системи

Комп'ютерними ігровими системами вважається програмне забезпечення (ПЗ), яке орієнтоване на дозвілля людини в побуті. Дане ПЗ реалізується на основі певної предметної області, яка являє собою частину реального світу, що розглядається в межах певного контексту. Контекстом можна розуміти область дослідження, яка є об'єктом певної діяльності. Іншими словами ігрова система може розроблятись на основі книг наукової фантастики, певних світових подій, тощо. Будь-які ігрові системи грунтуються на основі певної множини об'єктів, їх властивостей і відношень між ними.

В кваліфікаційній роботі предметною областю ігрової комп'ютерної системи розглядається «гра астероїди»[1]. Як зазначено в статті [1], метою гри є знищення астероїдів трикутним кораблем, яким керує гравець

Ігрові комп'ютерні системи часто складаються з великої кількості функціональних можливостей, які називаються механіками гри. За механіку вважається певного роду дія, як приклад знищення астероїду. Дана функція складається з кількох частин:

- постріл, що супроводжується візуальним ефектом у вигляді кулі;
- ураження астероїда, яке містить візуальний ефект.

Проте дана механіка описує лише останній етап життєвого циклу астероїда, що є проміжним етапом ігрової комп'ютерної системи. Ключовою механікою системи, яка розглядається в кваліфікаційній роботі є політ астероїда. Дана механіка є досить проста, адже астероїд летить лише в одному напрямку з однією швидкістю заданою, при його створенні ігровим двигуном.

Модуль нейронної мережі[2], що потрібно імплементувати в ігрову комп’ютерну систему реалізує іншу механіку, якої немає в ігровому двигуні.

Механіка реалізує визначення потрібного кута нахилу корабля до астероїда, щоб здійснити його ураження.

На основі вище зазначеного побудовано візуальне представлення механік реалізованих ігровою комп’ютерною системою, та імплементацією нейронної мережі, що показано на рис. 1.1.

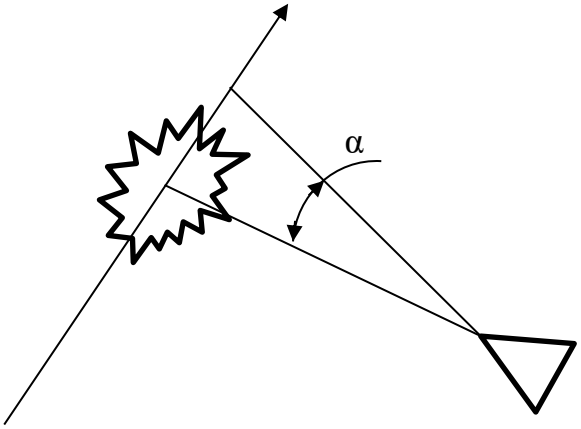



Рис. 1.1. Візуальне представлення механік реалізованих ігровою комп’ютерною системою та імплементацією нейронної мережі

## 1.2. Аналіз предметної області

На основі попередньо розглянутих ігрових механік, що відображені на рис. 1.1, проведемо формалізацію предметної області «гри астероїди» до певного об’єкту з набором сутностей, що наведено нижче:

– ігрове поле – відповідає за відображення ігрового процесу. Являє собою матрицю з певними значеннями;

– корабель – об’єкт керований гравцем або модулем інтелектуального перехоплення об’єктів, що рухаються;

– астероїд – некерований об’єкт створений ігровим двигуном з заданим йому певними характеристиками;

– двигун гри – структура розроблена для виконання низки задач, таких як: відображення ігрового поля та об’єктів на ньому, до яких відносяться корабель та астероїди. Також двигун здійснює контроль та опрацювання життєвого циклу об’єктів, куди також входить їх колізія та переміщення.

Імплементація інтелектуального модуля для даної ігрової комп’ютерної системи передбачає необхідність побудови UML діаграми, на основі вище описаних сутностей, для подальшого її використання в визначенні типу нейронної мережі.

Розроблену UML діаграму сутностей, зображено на рис. 1.2.

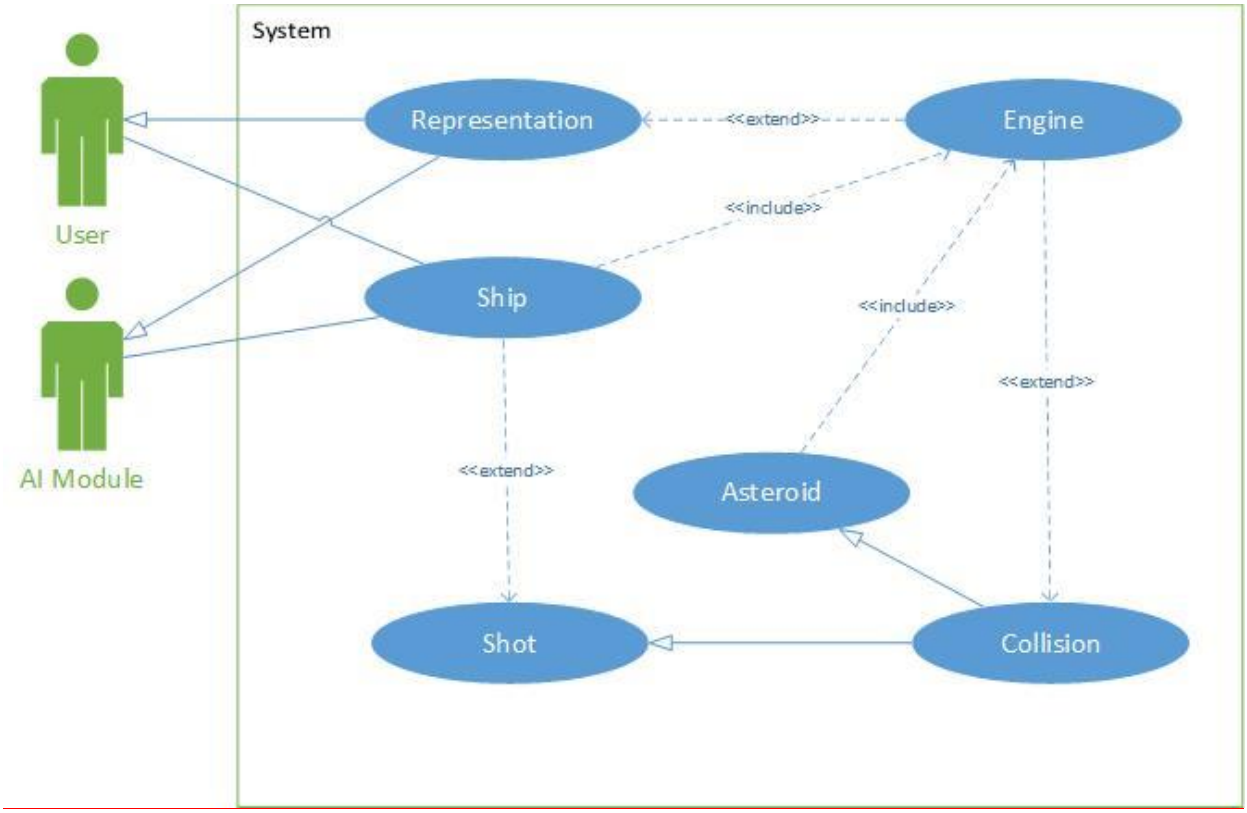



Рис. 1.2 UML діаграма сутностей

## 1.3. Аналіз технологій та методів розробки проектованої системи

### 1.3.1. Технології розробки програмного забезпечення

Тендеції сучасного світу до розробки ігрових комп’ютерних систем передбачають кросплатформову підтримку, для охоплення якомога набільшої частини ринку. Кросплатформова система також є більш конкурентно здатною, оскільки не обмежується функціонуванням на лише одній системі.

Головною вимогою ігрової комп’ютерної системи полягає в простоті її реалізації та здатності до імплементацій нових методів і засобів управління.

Кросплатформова реалізація[3] передбачає функціонування однієї програми на різних відмінних між собою пристроях за допомогою мережевих технологій, якщо необхідний доступ інформації до одного центрального ресурсу. Часто дане ПЗ розробляється з підходом до REST API проектування систем по клієнт серверній архітектурі.

В даному випадку немає необхідності до реалізації даної архітектури ПЗ, оскільки ігрова комп’ютерна система повина виконуватись лише на стороні клієнта. Оскільки розробляти ігрову комп’ютерну систему для кожного пристрою зі своїм набором інструментів не є доцільним, слід розглянути одни з таких найбільш широко доступних засобів доступу до масової інформації, як веб-браузер. Веб переглядач також широко використовується приватними компаніями для веденя локальної електронної бази даних, тощо за допомогою веб додатків, які функціонують в браузері, що дозволяє забезпечити кросплатформовість. Розробка веб-додатків може реалізовуватись за допомогою різних мов програмування. Згідно вище зазначених вимог проведено аналіз мов розробки програмного забезпечення серед, яких є наступні:

1. C# – Дана мова є оптимізованою з точки продуктивності до системи, проте позиціонується, як пропієтарна, адже часто використовує платі бібліотеки, для розробки ПЗ, також дана мова функціонує на стороні сервера, що не потрібно для розроблюваної системи.

2. Java – Позиціонується як мова для розробки великих, та стійких до розширення ПЗ згідно з використанням таких патернів проектування як SOLID, адже являється строго типізованою. Поміж переваг даної мови слід відмітити один суттєвий недолік, а саме функціонування на стороні сервера, хоч і можлива розробка клієнтської частини за допомогою фреймворків. Також для функціонування ПЗ на сервері потрібна імплементація необхідних інструментів до пакету яких входить JVM.

3. JavaScript – високорівнева мультипарадигмова мова програмування, яка підтримує об’єктно орієнтований, імперативний та функціональні стилі. Дана мова являє собою нативним засобом розробки, веб додатків для веб-переглядачів. В даному контексті слід виділити недолік. Виконання програмного коду здійснюється на стороні клієнта в реальному часі, що покладає обчислювальні навантаження на фізичну машину користувача.

Згідно до вище перерахованих варіантів, найкращим для реалізації підходить JavaScript[4]. Дана мова розроблялась як браузерна технологія, з рядом таких можливостей:

– повна інтеграція з HTML/CSS.

– підтримується усіма основними браузерами і увімкнутий по замовчуванню.

– простий та легкий в освоєнні.

– висока швидкість та продуктивність роботи завдяки двигуну V8, який поширюється з відкритим вихідним кодом.

Завдяки широкому поширенню володіє потужною інфраструктурою, що містить велику кількість фреймворків і бібліотек серед, яких для розробки глубоких нейронних мереж і машинного навчання brain.js, synaptic.js, tensorflow.js, neataptic.js.

### 1.3.2. Методи інтелектуальної імплементації в ігрових комп’ютерних системах

Обробники механік ігрових комп’ютерних систем часто будуються на основі певних алгоритмів, як приклад для ігрової комп’ютерної системи, яка базується на предметній області «змійка», можна використати Гамільтонів цикл[5], який дозволить досягнути 100% результату в даному випадку, проте в системах з однорідними динамічно змінюваними даними, де не можливе застосування таких алгоритмів, як зазначалось раніше. Кращим рішенням є проектування штучного інтелекту (ШІ)[6].

Об’єктом проектування виступає процес побудови, навчання і програмної реалізації штучної нейронної мережі (ШНМ), яка є сукупністю моделей біологічних нейронних мереж. Нейронна мережа представляє собою мережу елементів (штучних нейронів), пов’язаних між собою за допомогою синаптичних ребер. Мережа опрацьовує вхідну інформацію і в процесі зміни свого стану в часі формує сукупність вихідних сигналів.

Реалізація імплементації модуля ШНМ нативними засобами мови без використання сторонніх бібліотек, обумовлює аналіз структури нейронної мережі з її елементами.

Нейронні мережі виникли з дослідів в сфері штучного інтелекту, а саме зі спроб відтворити здатність біологічно нервових систем навчатись та виправляти помилки, моделюючи їх низькорівневу структуру мозку.

Математичну модель штучного нейрона вперше було запропоновано У. Маккалоком і У. Піттсом. Практична реалізація нейронної мережі була розроблена Френком Розенблатом в 1958 році, у вигляді комп’ютерної програми, а пізніше і як електронний пристрій – перцептрон. Перша реалізація нейрона могла оперувати лише сигналами логічного нуля та одиниці, оскільки була побудована на основі біологічного прототипу, який може перебувати лише в булевому стані – збудженому і незбудженому. Розглянемо структуру біологічного та штучного нейрона і зв’язок між ними.

Біологічний нейрон. Мозок складається з дуже великої кількості (приблизно 10 000 000 000), нейронів, сполучених численними зв’язками.

Нейрони – це спеціальні клітини, здатні розповсюджувати електрохімічні сигнали. Тіло клітини містить велику кількість розгалужених відростків двох типів – дендріти та аксони. Дендріти слугують вхідними каналами для нервових імпульсів від інших нейронів.

Ці імпульси надходять в тіло клітини розміром від 3 до 100 мікрон, викликаючи її специфічне збудження, яке розповсюджується по аксону. Аксони клітини з’єднуються з дендритами інших клітин за допомогою синапсів. При активуванні нейрон надсилає електрохімічний сигнла своєму аксону. Через синапси цей сигнал досягає інших нейронів, які можуть активуватись. Нейрон активується тоді, коли сумарний рівень сигналів, що надійшли в його ядро з дендритів, перевищує певний рівень (поріг активації).

Біологічний нейрон являється об’єктом на основі, якого реалізується штучний нейрон, що імітує властивості біологічного нейрона. Кожен вхід множиться на відповідну вагу, аналогічно синаптичній силі, і все інші входи сумуються, визначаючи рівень активації нейрону.

Модель штучного нейрона зображено на рис. 1.3. Вхідні сигнали x множаться на ваги w, після чого сумуються. Отримане число подається на вхід функції активації φ, яка розраховує вихідне значення.

Однією з найважливіших складових нейрона є його функція активації, яка визначає вихідне значення. Серед загальних можна визначити такі:

1. Гранична – одна з найпримітивніших, яка приймає лише два значення 0 та 1.

2. Лінійна – являє собою пряму лінію, яка пропорціональна до входу, а саме сумі значень на нейроні. Дана функція відображає декілька, значень, на відміну, від бінарної, якою є гранична.

3. Сигмоїдальна – нелінійна функція, і в цьому є її перевага, адже для функцій такого типу є характерним гладкий градієнт, що дозволяє об’єднувати шари нейронних мереж.

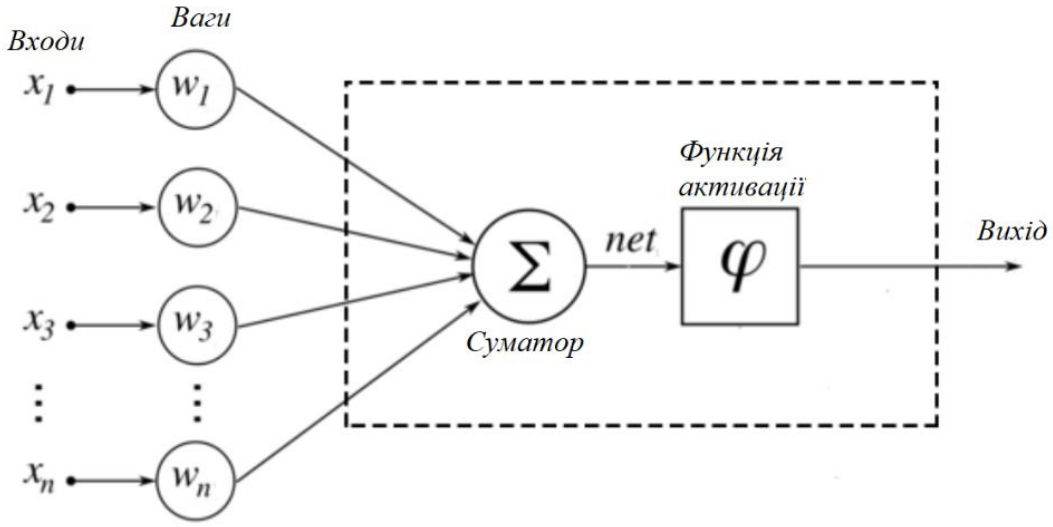



Рис.1.3 Модель штучного нейрона

Функції активації розраховує вихідний сигнал нейрону по рівню активності. На рис.1.4 зображено графіки деяких функцій активації.

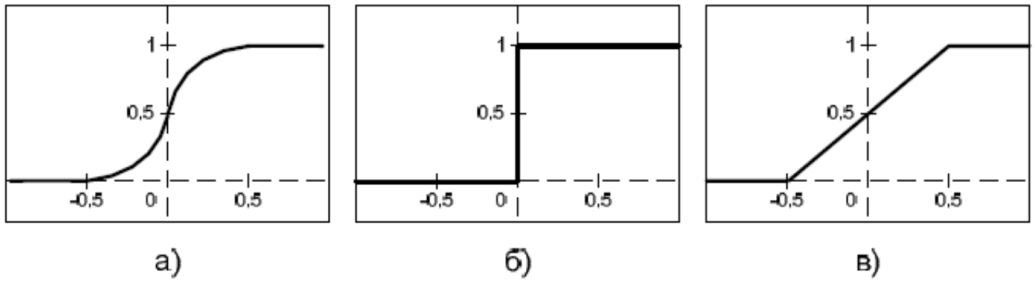



Рис.1.4. Функції активації: а) Сигмоїдальна; б) Гранична; в) Лінійна

Хоч одному перцептрону по силі виконувати лише елементарны функції, але не варто недооцінювати його потенціал, адже сила нейронних обчислень залежить від об’єднань нейронів в мережі. Прикладом простої мережі, яка

складається з групи нейронів, які утворюють собою шар як зображено на правій частині рис. 1.5.

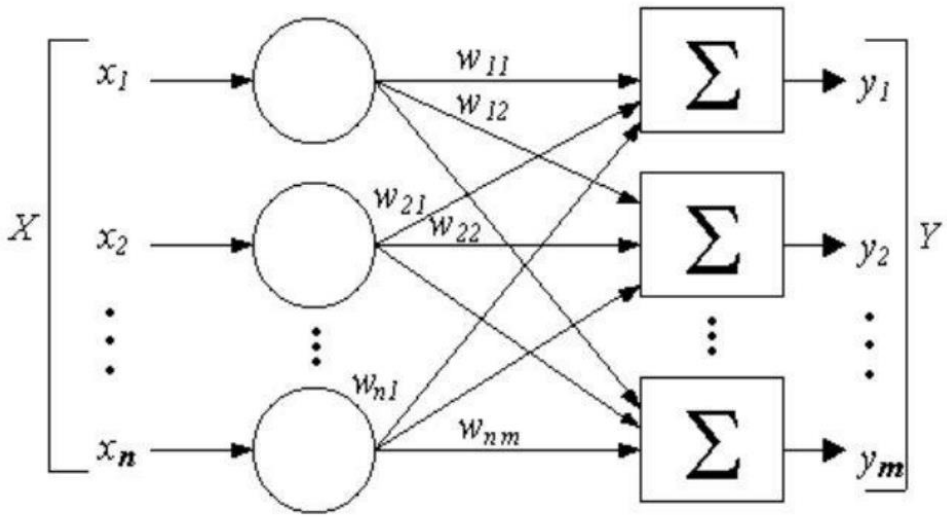



Рис. 1.5. Найпростіша одношарова нейронна мережа

Потрібно зазначити, що вершини-круги зліва виконують роль розподілення вхідних даних. Їх не рахують шаром, оскільки вони не здійснюють жодних обчислень. З цієї причини вони позначаються порожніми колами, для розмежування їх з обчислюваними нейронами, що позначаються квадратами. Кожен з елементів множини входів Х з’єднується окремими вагами з кожним штучним нейроном, а кожен нейрон в свою чергу видає зважену суму входів в мережу. В біологічних та штучних мережах багато зі зв’язків можуть бути відсутніми. Усі зв’язки зображено для узагальнення. Також можливе з’єднання між вхідними і вихідними елементами шару. З практичної точки зору зручніше буде вважати ваги за елементи матриці W. Матриця містить n рядків і m стовпців, де n – кількість входів, а m – кількість нейронів. Наприклад, w23 – це вага, яка пов’язує другий вихід з третім нейроном. Тобто таким чином, обчислення вихідного вектора Y, компонентами якого являються виходи y1 нейронів зводиться до множення матриць Y=XW.

Як правило більш складні нейронні мережі, володіють більшими обчислювальними можливостями. Хоч і створено мережі всіх можливих конфігурацій, які тільки можна уявити, пошарова організація нейронів повторює шарові структури конкретних відділів мозку. Виявилось, що такі багатошарові мережі володіють більшими можливостями на відміну від одношарових див. рис. 1.5, також в останні роки було розроблено різноманітні алгоритми[7] для їх навчання.

Багатошарові мережі також можуть складатись з каскадів інших шарів. Вихід першого шару, являє собою вхід для другого шару.

Проте багатошарові мережі не можуть призвести до збільшення обчислювальної потужності в порівнянні з одношаровою мережею лише в тому випадку, якщо активаційна функція між шарами не буде лінійною. Обчислення виходу шару полягає в множинні вхідного вектора на першу вагову матрицю з наступним множенням (якщо відсутня нелінійна активаційна функція) результуючого вектору на другу вагову матрицю.

Перед подальшим проектуванням необхідно проаналізувати поширені типи нейронних мереж[8] та здійснити їх порівняння, обравши найбільш оптимальний варіант.

Серед десятків видів нейронних мереж, які відрізняються своєю архітектурою можна виділити три найбільш поширені:

1. Мережі з методом прямого поширення (FFNN)[9] – тип нейронної мережі в якій вузли шару не пов’язані між собою, а передача інформації відбувається одразу на вихід. Також часто можуть використовуватися у комбінації з ще двома мережами.
2. Згорткові нейронні мережі (CNN). Дана архітектура передбачає п’ять типів шарів:
   a. вхідного;
   b. згорткового
   c. об’єднуючого;

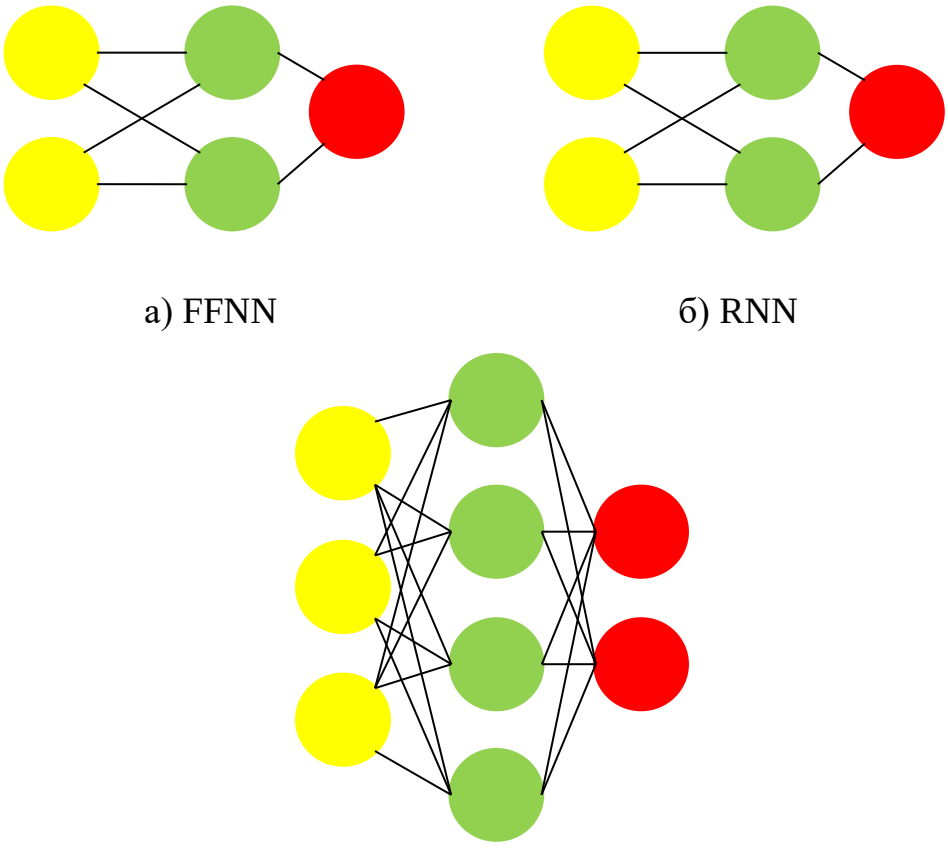
d. підключеного;

e. вихідного.

Кожен з цих шарів, виконує конкретну роль. Наприклад узагальнює, або об’єднує дані. Даний тип мереж використовується для класифікації зображень, прогнозування, обробки нативної мови та інших задач.

3. Рекурентні нейронні мережі (RNN). Використовують направлену послідовність зв’язків між вузлами. Завдяки цьому RNN можуть обробляти серії подій в часі і використовувати результат даних для наступної серії обробки цих даних. Такі мережі використовуються для мовного моделювання, генерування текстів, машинного перекладу, розпізнавання голосу та інших задач.

Вигляд архітектури даних мереж зображено на рис. 1.6.



а) FFNN б) RNN

в) CNN

Рис. 1.6. Архітектури нейронних мереж

Узагальнивши вище описані архітектури мереж та проаналізувавши UML діаграму сутностей, можна сказати, що реалізація CNN мережі не є доцільною в даному випадку, оскільки імплементація нейронної мережі передбачає безпосередній доступ до даних ігрової комп’ютерної системи. Використання цього типу мережі в парі з такою як FFNN, є доцільним лише тоді коли немає безпосереднього доступу до даних ігрової комп’ютерної системи, в результаті чого виникає необхідність обробки ігрового поля, після чого вже здійснюється узагальнення даних з подальшою їх обробкою в частині FFNN.

Згідно до здійснених аналізів ігрової комп’ютерної системи та предметної області, обраною архітектурою виступає FFNN.

Результатами виконання першого розділу є:

1. Проведено аналіз особливостей ігрової комп’ютерної системи за час, якої здійснено узагальнення предметної області.
2. Проаналізовано узагальнення предметної області, що дало змогу здійснити формалізацію об’єкта ігрової комп’ютерної системи.
3. Проведено аналіз технологій реалізації згідно до тенденцій розробки у сучасному світі.
4. Здійснено аналіз структури нейронної мережі, для забезпечення подальшої розробки математичного забезпечення.

# РОЗДІЛ 2
# МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ІГРОВОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

Реалізація нейронної мережі передбачає аналіз гри, як об'єкта, що в свою чергу дозволяє класифікувати вхідні і вихідні дані даної гри. Формалізація гри як об'єкта дослідження передбачає аналіз відображення ігрового поля.

Оскільки гра реалізована за допомогою скриптової мови JavaScript[10], відповідно рендер ігрового поля відбувається за допомогою елемента «canvas»[11], що призначений для створення растового двовимірного зображення. Загострення уваги на даному елементі здійснено, через спосіб відображення полотна гри в лівому верхньому куті робочого поля браузера, відповідно система координат буде приймати наступні значення, як зображено на рис. 2.1.

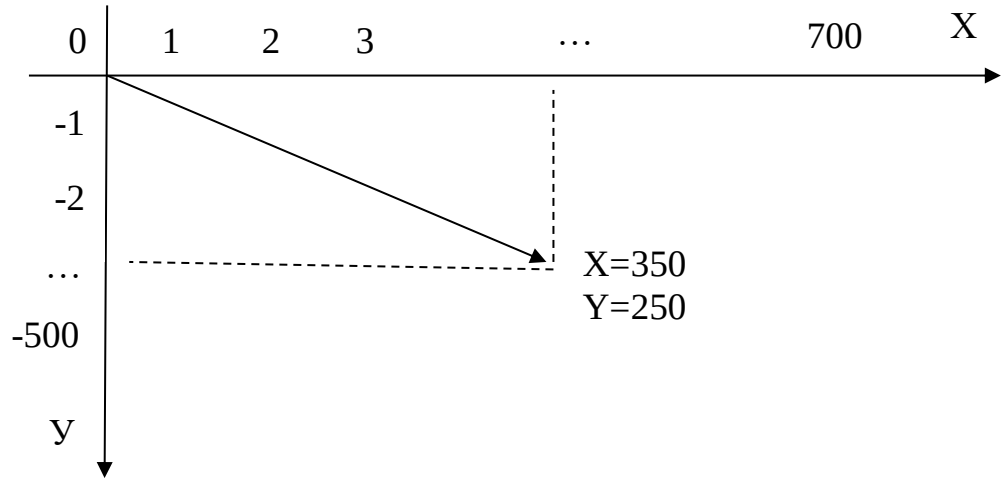



Рис. 2.1. Поле canvas

Аналіз відображення ігрового поля є ключовим елементом який розглянуто на даному етапі. В подальшому при написанні програми для розрахунку кута між об'єктами буде застосовано математичну функцію atan2[12], яка нативно підтримується мовою JavaScript.

## 2.1. Математична модель нейронної мережі

Як зазначено в розділі 1.3.2 нейронні мережі являють шари перцептронів, які поєднані між собою за допомогою ваг, які містять певний коефіцієнт. З математичної точки зору нейронні мережі містять входи, приховані шари та виходи. Якщо прихованих шарів більше одного, тоді нейронна мережа з глибоким навчанням. В даному випадку використовується мережа з одним прихованим шаром, як зображено на рис. 2.2.

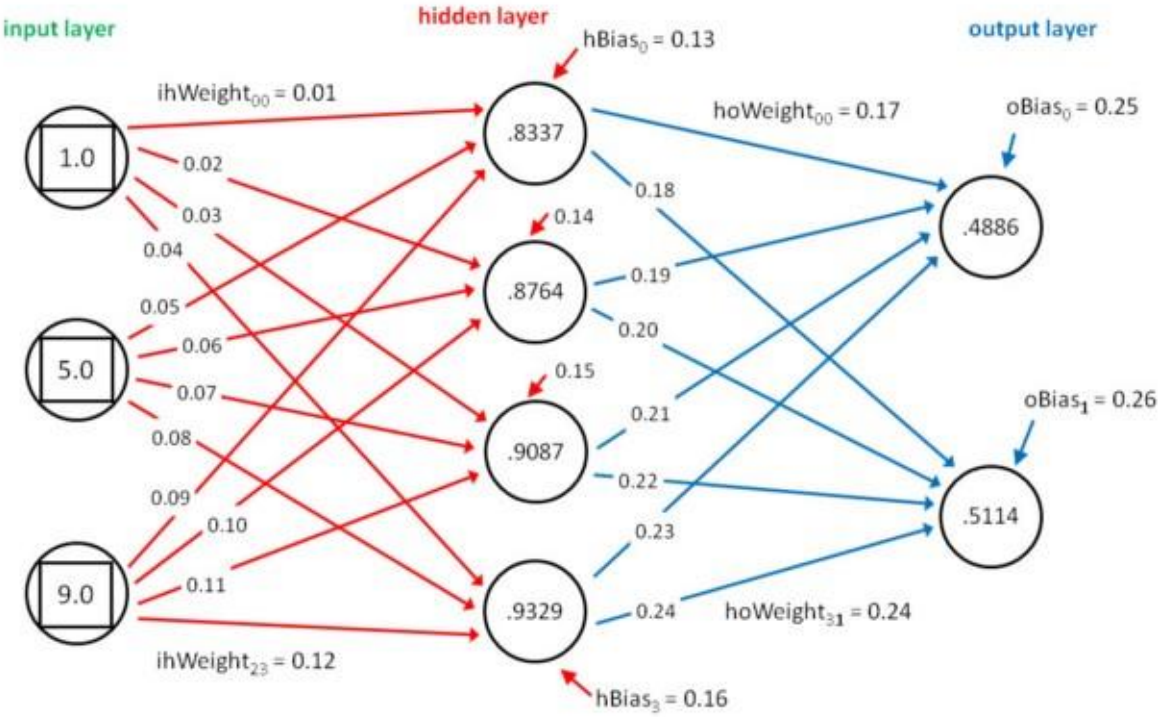



Рис. 2.2. Схема нейронної мережі

Дана мережа містить, нейрони зміщення, які з’єднані з нейронами прихованого шару та нейронами виходу. Даний нейрон зміщення використовується і задіяний в процесі навчання мережі. Даний нейрон відображає на скільки відрізняється очікуваний результат від отриманого на виході мережі.

В комп’ютерних системах і лінійній алгебрі нейронні мережі можна описати за допомогою векторів і матриць, з якими можна проводити необхідні обчислення[13,14]. Навчання даної нейронної мережі здійснюється по алгоритму FeedForward (подача вперед). Даний алгоритм реалізується наступним чином:

$$\begin{bmatrix} w11 & w12 & w13 \\ w21 & w22 & w23 \\ w31 & w32 & w33 \end{bmatrix} \cdot \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} w11 \times x1 + w12 \times x2 + w13 \times x3 \\ w21 \times x1 + w22 \times x2 + w23 \times x3 \\ w31 \times x1 + w32 \times x2 + w33 \times x3 \end{bmatrix} \quad (2.1)$$

Як видно з формули (1.2), w11 … w13, w21 … w23, w31 … w33 відповідають з’єднанням між входами там прихованим шаром. Дана формула відображає суму ваги для кожного зі з’єднань, де на прикладі формули рядок ваг перемножено на стовпець входів. На основі даного прикладу функціонує алгоритм прямої подачі, але слід зазначити, що дана формула є не повноцінна адже відсутні нейрони зміщення та активаційна функція див. розділ 1.3.2.

Як говориться вище в нейронній мережі також задіяно нейрон зміщення та активаційна функція сигмоїда див. розділ 1.3.2.

Формула прийняття рішень для першого шару виглядає наступним чином:

$$\begin{bmatrix} w11 & w12 & w13 \\ w21 & w22 & w23 \\ w31 & w32 & w33 \end{bmatrix} \cdot \begin{bmatrix} x1 \\ x2 \\ x3 \\ 1 \end{bmatrix} = \begin{bmatrix} w11 \times x1 + w12 \times x2 + w13 \times x3 + b1 \\ w21 \times x1 + w22 \times x2 + w23 \times x3 + b2 \\ w31 \times x1 + w32 \times x2 + w33 \times x3 + b3 \end{bmatrix} \quad (2.2)$$

Де 1 це нейрон зміщення, а b1 … b3 вага зміщення. Результат кожного вектора обчислюється за допомогою сигмоїди. Формула сигмоїди[15] має наступний запис:

$$sigmoid(x) = \frac{1}{1+e^{-x}} \quad (2.3)$$

Причина використання нелінійної сигмоїдної функції, полягає в її особливості «зглажування» значення до діапазону від 0 і наближено до 1, що допомагає стиснути великі значення ваг з'єднань.

Тепер коли зрозуміло як відбувається прийняття рішень потрібно відобразити скорочений запис функції розрахунку активації нейронів для обох шарів. Формула функції активації для прихованого шару виглядає наступним чином:

$$\mathrm{H} = \sigma(W_{ij}^{IH} \cdot I_i + B_i^H) \tag{2.4}$$

де $W$ – це вага між з'єднанням $i$ входу та вузла прихованого шару $j$.;

$I_i$ – вхідний вузол

$B_i$ – зміщення вузла прихованого шару

Формула функції активації для вихідних нейронів виглядає наступним чином:

$$O = \sigma(W_{ij}^{HO} \cdot H_i + B_i^O) \tag{2.5}$$

де $W$ – це вага між з'єднанням $i$ вузла прихованого шару та виходом $j$.;

$H_i$ – вузол прихованого шару

$B_i$ – зміщення вузла виходу

Оскільки результат на виході нейронної мережі часто далекий від ідеалу, в таких випадках застосовуються алгоритми навчання.

Навчання нейронної мережі здійснюється за допомогою алгоритму зворотнього поширення (backpropagation)[16]. Даний алгоритм реалізується згідно прикладу нейронної мережі, що зображена на рис. 2.3.

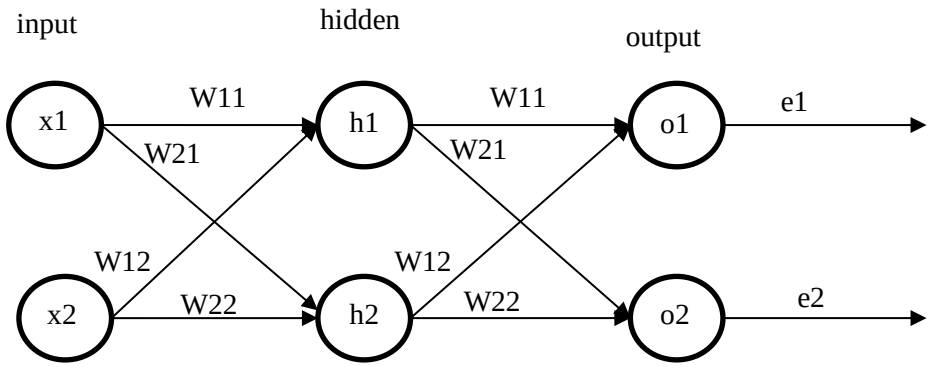



Рис. 2.3. Нейронна мережа в зі зворотнім поширенням

Формула врахування помилок для мережі зі зворотнім поширенням згідно до рис. 2.3:

$$E = output - target \tag{2.6}$$

$$\Delta w_{ij}^{ho} = lr \cdot E \cdot (output + (1 - output)) \cdot H^T \tag{2.7}$$

Де W вага між з’єднаннями *i* вузла прихованого шару і виходом *j*

lr – швидкість навчання

H – транспонована матриця

$$\Delta w_{ij}^{ih} = lr \cdot H\_E \cdot (H \cdot (1 - H)) \cdot I^T \tag{2.8}$$

де W вага між з’єднанням *i* входу і *j* вузла прихованого шару

H_E – вага помилки для прихованого шару

Вище описані формули дозволяють корегувати вагу шарів нейронної мережі, за допомогою множення матриць.

Задля реалізації нейронної мережі згідно вище описаних формул, розробляються блок-схеми методів нейронної мережі та головного тіла програми, які будуть використані при розробці програми на мові JavaScript.

## 2.2. Обґрунтування моделі розроблюваного модуля

Головний алгоритм роботи програми зображено на рис. 2.4.

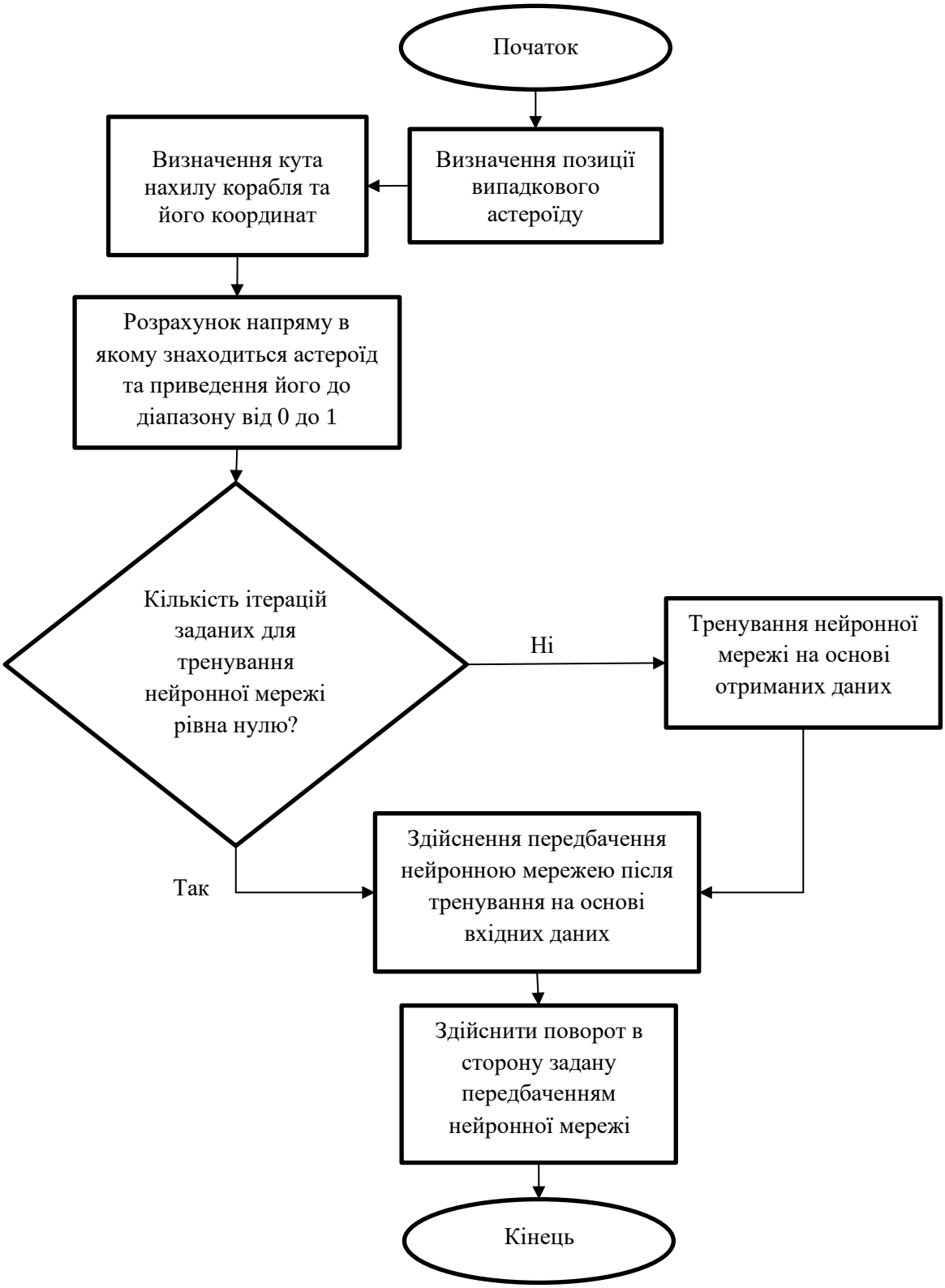



Рис. 2.4. Блок-схема алгоритму роботи головної програми

На рис. 2.5 зображено блок-схему методу обчислення добутку двох матриць різної розмірності.

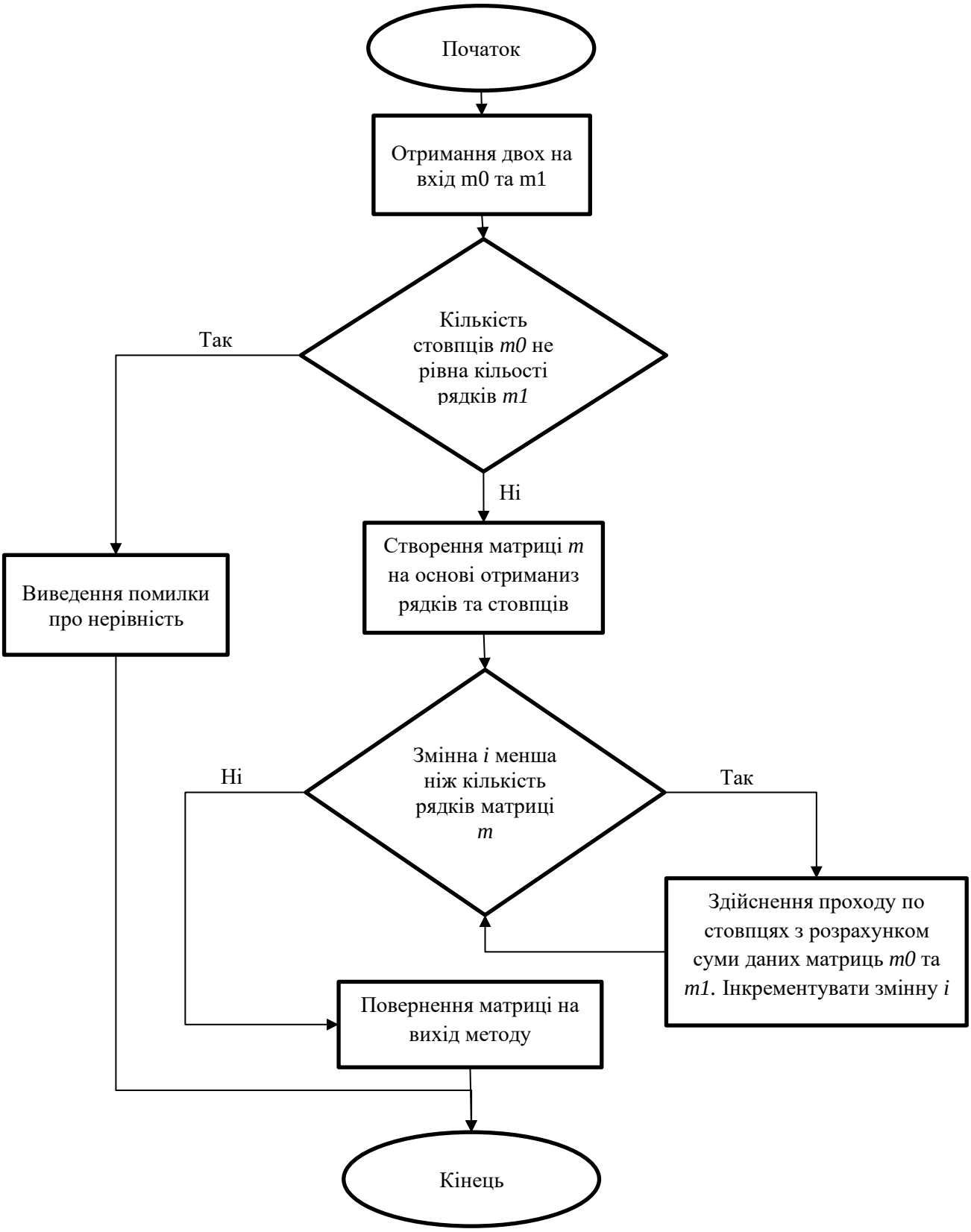



Рис. 2.5. Метод обчислення добутку двох матриць різної розмірності

На рис. 2.6 зображено алгоритм методу приведення вектора до матриці.

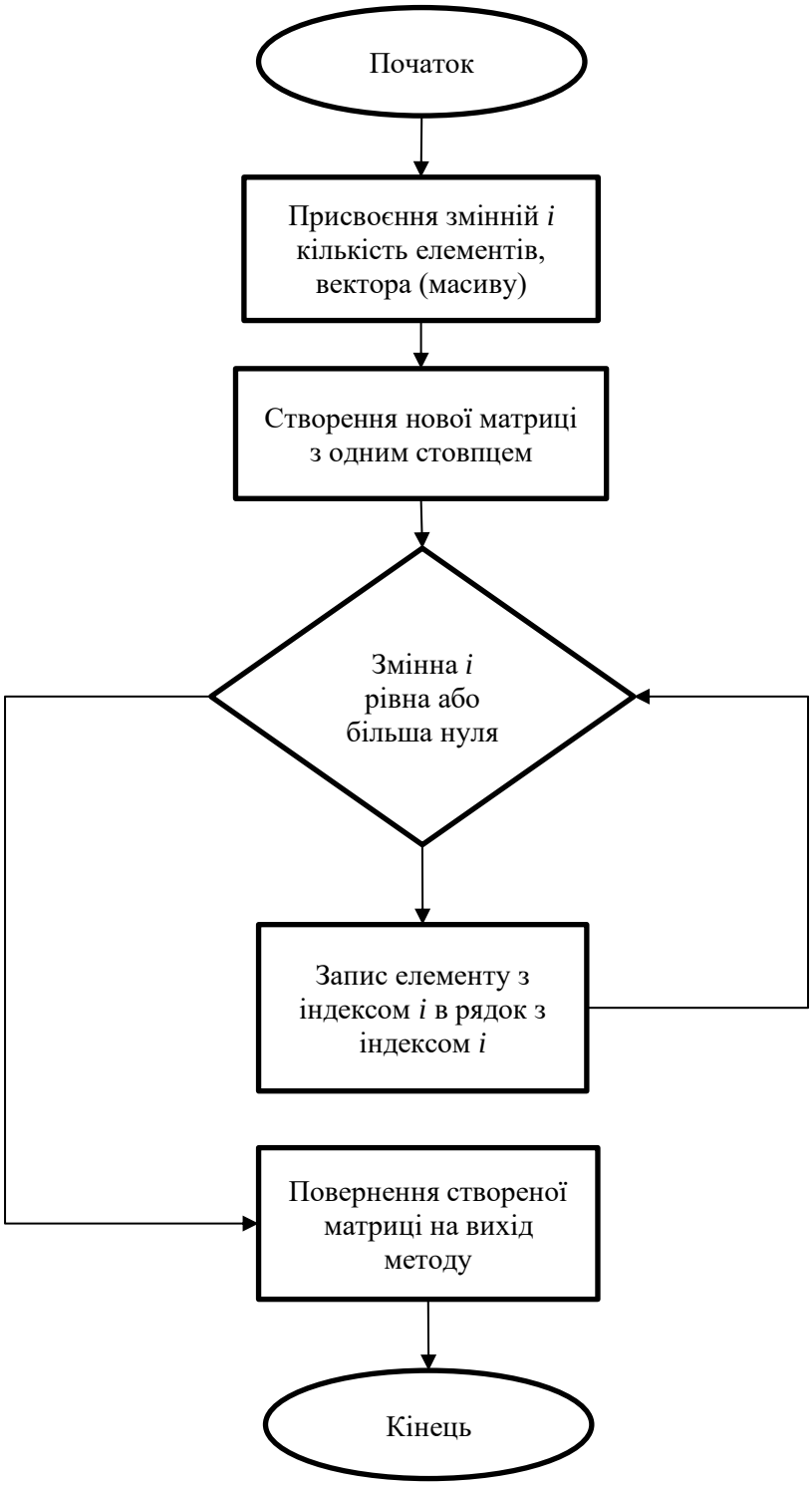



Рис. 2.6. Алгоритм роботи методу приведення вектору до матриці

На рис. 2.7 зображено алгоритм роботи методу застосування сигмоїдної функції для елементів матриць.

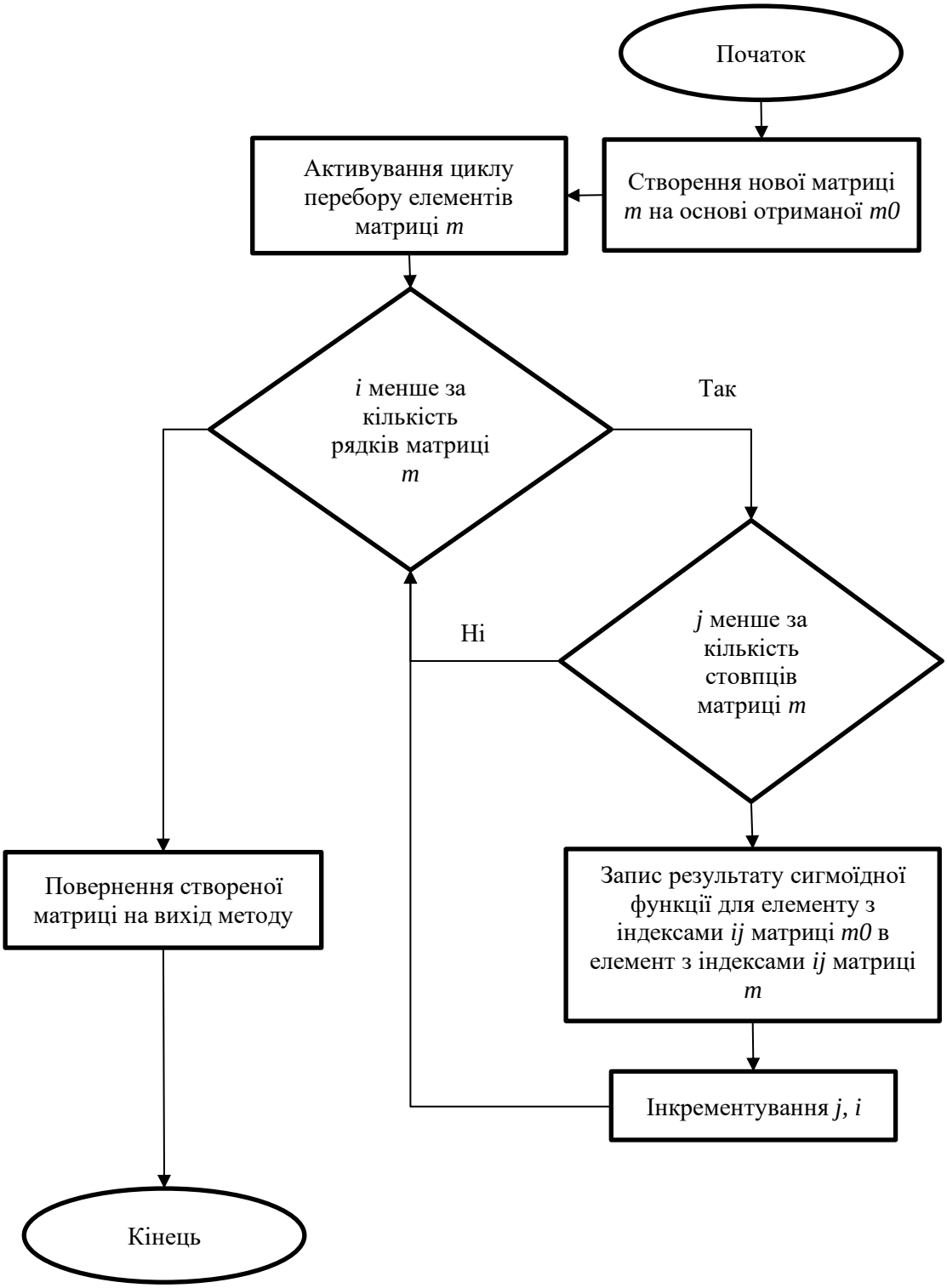



Рис. 2.7. Алгоритм застосування сигмоїдної функції для кожного з елементів нейронної мережі

На рис. 2.8 зображено алгоритм транспонування матриці, за допомогою якої рядки і стовпці міняються місцями.

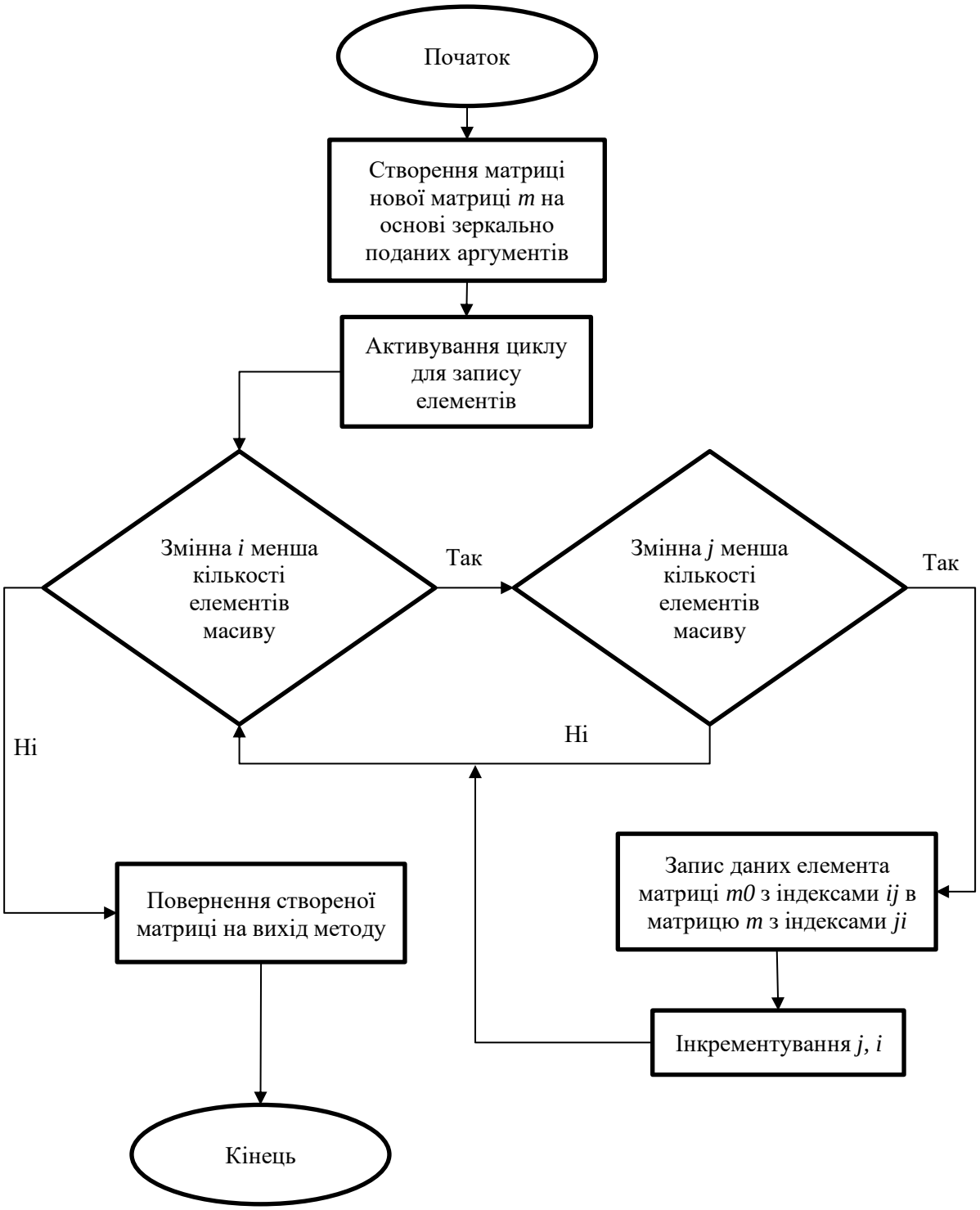



Рис. 2.8. Алгоритм транспонування матриці

На рис. 2.9 зображено алгоритм, який підходить як для додавання і віднімання, так і множиння матриць.


Рис. 2.9. Алгоритм множення, додавання та віднімання матриць

Як показано на рис. 2.4, можна зробити висновок, що точність передбачення нейронної мережі залежить від кількості пройдених ітерацій за час яких було врегульовано шари ваг та нейрони зміщення. В програмній реалізації здійснено 100 000 ітерацій, що дозволило суттєво зменшити похибку при передбаченні кута нахилу корабля до астероїда.

Результатами виконання другого розділу є:

1. Запропоновано алгоритм навчання нейронної мережі на основі зворотного поширення помилки та визначено функції активації нейронів, що дало змогу забезпечити ефективність управління та моніторингу руху об’єктів у комп’ютерній ігровій системі.

2. Спроектовано концепт нейронної мережі для забезпечення виконання математичних операцій над матрицями, що дало змогу визначити її архітектуру за складом шарів і забезпечити оптимальний час навчання нейронної мережі.

3. Запропоновано блок-схеми алгоритмів програмної реалізації ігрової комп’ютерної системи, що дало можливість відтворити етапи виконання функцій та їх умов, а також досягти високої інтерактивності при подальшій реалізації у вигляді веб-орієнтованого додатку.

# РОЗДІЛ 3
# АПРОБАЦІЯ ІГРОВОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ ТА ІМПЛЕМЕНТАЦІЯ МОДУЛЯ ІНТЕЛЕКТУАЛЬНОГО ПЕРЕХОПЛЕННЯ РУХОМИХ ОБ'ЄКТІВ

## 3.1. Реалізація модуля нейронної мережі

Програмне забезпечення розробляється за допомогою скриптової мови JavaScript, яка підтримує об'єктно орієнтоване програмування (ООП). В ООП ПО будується на основі класів їх екземплярів та відношень між ними. Розроблювана нейронна мережа являє собою клас, екземпляр, якого створюється за допомогою спеціального блоку інструкцій, який називається конструктором класу[17]. Кожен клас містить набір функцій, які в ООП називаються методами.

Створення класу нейронної мережі наведено у лістингу на рис. 3.1.

```
class NeuralNetwork {
    constructor(numInputs, numHidden, numOutputs) {
        this._inputs = [];
        this._hidden = [];
        this._numInputs = numInputs;
        this._numHidden = numHidden;
        this._numOutputs = numOutputs;
        this._bias0 = new Matrix(1, this._numHidden);
        this._bias1 = new Matrix(1, this._numOutputs);
        this._weights0 = new Matrix(this._numInputs,
this._numHidden);
        this._weights1 = new Matrix(this._numHidden,
this._numOutputs);
        this._bias0.randomWeights();
        this._bias1.randomWeights();
        this._weights0.randomWeights();
        this._weights1.randomWeights()};
```

Рис. 3.1. Лістингс творення класу нейронної мережі

Даний клас див. рис. 3.1 містить конструктор, який приймає такі аргументи, як кількість входів, кількість прихованих нейронів, кількість виходів.

Дані аргументи присвоюються полям (змінним), які доступні лише для даного класу. Доступ до даних полей здійснюється за допомогою гетерів та сетерів див. лістинг 3.2. Поля «_inputs» та «_hidden», являються масивами, які містять певну кількість елементів, за своєю суттю дані елементи є нейронами мережі. Наступні поля «_numInputs, _numHidden, _numOutput», всього лиш константи, які визначають кількість елементів вище описаних масивів. Наступні поля – це «_bias0, _bias1». Дані нейрони зміщення в даному випадку є об'єктами (екземплярами класу), кожен з яких відповідає за свій шар ваг. Дані об'єкти створюються за допомогою оператора «new» після чого вказується клас до якого вони відносяться та аргументи, які використовуються для створення даного об'єкта. З даного відрізку коду можна зрозуміти, що задається значення «1» відповідає за кількість рядків матриці, а інше за кількість стовпців. Хоч даний об'єкт являється матрицею і розраховується як матриця згідно формул в розіді 2.2, проте в програмній реалізації його краще виразити як вектор (масив) для спрощення роботи. Останіми полями даного класу є зв'язки між нейронами «_weights1, _weights2», які також є матрицями для кожного з шару.

Наступними рядками коду є генерування випадкових значень для раніше створених екземплярів класу «Matrix». Даний метод доступний лише екземплярам класу від якого вони були створені. Забігаючи на перед слід сказати, що дані значення являються початковими оскільки все це відбувається в середині конструктора класу.

Поза конструктором класу реалізовано методи, які називаються гетерами і сетерами. Дані методи реалізують обмеження доступу до полей класу, що позитивно впливає на захищеність. Гетери та сетери[18] наведено в рис. 3.2.

```
get inputs() {
    return this._inputs;
}
set inputs(inputs) {
    this._inputs = inputs;
}
get hidden() {
    return this._hidden;
}
set hidden(hidden) {
    this._hidden = hidden;
}
```

Рис. 3.2. Лістинг гетерів та сетерів класу NeuralNetwork

Наступною є реалізація алгоритму FeedForward. В контексті ПЗ даний алгоритм реалізується як метод класу NeuralNetwork. Метод наведено у лістингу на рис. 3.3.

```
feedForward(inputArray) {
    this.inputs = Matrix.convertFromArray(inputArray);
    this.hidden = Matrix.dot(this.inputs, this.weights0);
    this.hidden = Matrix.add(this.hidden, this.bias0);
    this.hidden = Matrix.map(this.hidden, x =>
sigmoid(x));
    let outputs = Matrix.dot(this.hidden, this.weights1);
    outputs = Matrix.add(outputs, this.bias1);
    outputs = Matrix.map(outputs, x => sigmoid(x));
    return outputs;    }
```

Рис. 3.3. Лістинг метод feedForward

Даний метод здійснює ряд операцій над матрицями, функціонування яких відображено в блок-схемах див. розділ 2.2. Дані операції є статичними методами класу Matrix. Перевагою статичних методів на звичайними є те, що їх можна викликати безпосередньо за допомогою самого класу, без створення його екземпляру, що потребує певних затрат пам’яті ЕОМ. Також дані методи функціонують як звичайні функції, що дозволяє їм повертати результат безпосередньо в місце виклику, а не екземплеру класу, якщо викликати звичайний метод. Перша стрічка передає в поле (змінну) «inputs» результат конвертування вектора в матрицю. Інші результати що наведено нижче виконують наступне:

– dot виконує обчислення добутку двох матриць різної розмірності, як це показано в алгоритмі див. розділ 2.2;

– add виконує додавання двох матриць;

– map передає функцію сигмоїди, яка виконується для кожного елемента як показано в алгоритмі див. розділ 2.2;

В кінці методу відбувається повернення змінної «output», яка є вихідним значенням даного алгоритму згідно формул описаних в розділі 2.2. Метод train (тренування) відповідає за навчання нейронної мережі. Реалізація алгоритму backpropagation у вигляді ПЗ наведено у лістингах на рис. 3.4. та рис. 3.5.

```
    train(inputArray, targetArray) {
        let outputs = this.feedForward(inputArray);
        let targets = Matrix.convertFromArray(targetArray);
        let outputErrors = Matrix.subtract(targets, outputs);
        let outputDerivs = Matrix.map(outputs, x => sigmoid(x,
true));
        let outputDeltas = Matrix.multiply(outputErrors,
outputDerivs);
        let weights1T = Matrix.transpose(this.weights1);
        let hiddenErrors = Matrix.dot(outputDeltas, weights1T);
```

Рис. 3.4. Лістинг backpropagation

```
            let hiddenDerivs = Matrix.map(this.hidden, x =>
sigmoid(x, true));
            let hiddenDeltas = Matrix.multiply(hiddenErrors,
hiddenDerivs);
            let hiddenT = Matrix.transpose(this.hidden);
            this.weights1 = Matrix.add(this.weights1,
Matrix.dot(hiddenT, outputDeltas));
            let inputsT = Matrix.transpose(this.inputs);
            this.weights0 = Matrix.add(this.weights0,
Matrix.dot(inputsT, hiddenDeltas));
            this.bias1 = Matrix.add(this.bias1, outputDeltas);
            this.bias0 = Matrix.add(this.bias0, hiddenDeltas);
        }
    }
```

Рис. 3.5. Лістинг алгоритму backpropagation

З даного методу можна виділити раніше не розглянутий метод. Даний метод змінює рядки на стовпчики, як зображено в блок-схемі розділу 2.2.

Реалізація лістингу функції сигмоїди наведено на рис.3.6.

```
function sigmoid(x, deriv = false) {
    if (deriv) {
        return x * (1 - x); // where x = sigmoid(x)
    }
    return 1 / (1 + Math.exp(-x));
}
```

Рис. 3.6. Лістинг функції сигмоїду

Клас Matrix та його методи наведено в додатку А Лістинг програми, оскільки немає сенсу дублювати блок-схеми з розділу 2.2.

## 3.2. Імплементація модуля інтелектуального перехоплення об’єктів

Гра це об’єкт, який відображає якісь властивості, звук, зображення тощо. Відповідно ці властивості є вихідними даними, а гравець їх обробляє. Відповідно таким чином можна зробити висновок, що нейронна мережа, яка імітує гравця найсправді обробляє вихідні дані гри, тому і являється обробником.

Маніпуляції з вхідними та вихідними даними наведено у лістингу на рис. 3.7

```
    if (AUTOMATION_ON) {
      nn = new NeuralNetwork(NUM_INPUTS, NUM_HIDDEN,
NUM_OUTPUTS);
                let ax, ay, sa, sx, sy;
                for (let i = 0; i < NUM_SAMPLES; i++) {
     ax = Math.random() * (canv.width + ROID_SIZE) -
ROID_SIZE / 2;
    ay = Math.random() * (canv.height + ROID_SIZE) -
ROID_SIZE / 2;
                    sa = Math.random() * Math.PI * 2;
                    sx = ship.x;
                    sy = ship.y;
                    let angle = angleToPoint(sx, sy, sa, ax,
ay);
    let direction = angle > Math.PI ? OUTPUT_LEFT :
OUTPUT_RIGHT;
    nn.train(normaliseInput(ax, ay, angle, sa),
[direction]);
                }
            }
```

Рис. 3.7. Лістинг обробки даних

Обробник див. Лістинг рис. 3.7 організовано на основі умови «if», яку при бажанні можна вимикати замінивши константу «AUTOMATION_ON» на значення false або 0.

В даній програмі створюється екземпляр класу нейронної мережі. Наступні рядки коду відображають дані позиціонування астероїда, корабля та кут нахилу корабля які передаються нейронній мережі для навчання. Все це огорнуто циклом, який виконується визначену кількість разів, що забезпечує навчання нейронної мережі, яка буде отримувати нові дані при кожній новій ітерації циклу. Також слід звернути увагу на код визначення кута до точки (позиції астероїда). Раніше в розділі 2.1 було розглянуто принцип відображення об’єктів на полотні. Завдяки цьому при обчисленні для отримання потрібних результатів необхідно передати аргументи позиціонування астероїда по осі Y в негативну сторону і позитивну по осі X з поправкою на місце знаходження корабля. Врахування даного аспекту наведено в лістингу на рис. 3.8.

```
function angleToPoint(x, y, bearing, targetX, targetY) {
       let angleToTarget = Math.atan2(-targetY + y, targetX - x);
       let diff = bearing - angleToTarget;
            return (diff + Math.PI * 2) % (Math.PI * 2);
        }
```

Рис. 3.8. Лістинг визначення кута до точки (астероїда)

## 3.3. Апробація методів і засобів управління та моніторингу руху об’єктів в ігровій комп’ютерній системі

Апробацією розробленого ПЗ є його тестування в середовищі розробки, а саме браузері GoogleChrome. Тестування нейронної мережі проводиться за допомогою кількості ітерацій витрачених на навчання мережі.

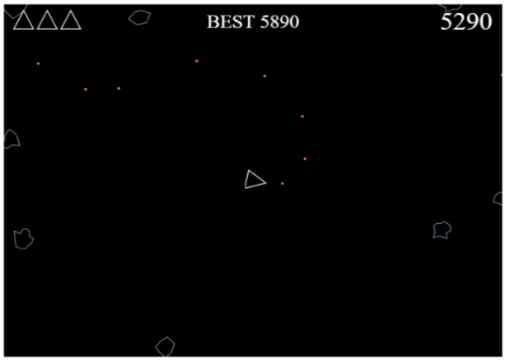
З кількістю ітерацій в 1000 нейронна мережа майже не надає потрібного значення повороту корабля, оскільки вона не знає «куди» повертати. Результат зображено на рис. 3.9.


Рис. 3.9. Результат навчання з 1000 ітерацій

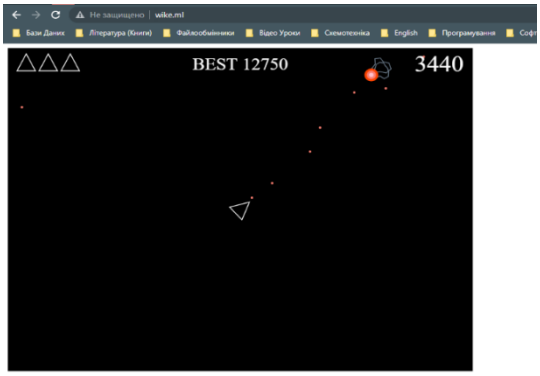
Результат з кількістю ітерацій в 10 000 дещо кращий від попереднього, адже мережа проявляє спроби передбачення місцезнаходження астероїда.

Спроби передбачення з 10 000 ітерацій навчання зображено на рис. 3.10(а).

Вимоги до нейронної мережі вдається задовольнити при 100 000 ітерацій навчання. Дана мережа навчилась точно визначати ціль та уражати її, що також видно з показника найкращого результату в 12750 балів. Найкращий результат навчання нейронної мережі зображено на рис. 3.10 (б).




а)




б)

Рис. 3.10. Результат навчання

а) навчання 10 000 ітерацій, б) навчання 100 000 ітерацій

Повний програмний код наведено в додатках.

Результатами виконання третього розділу є:

1. Спроектовано та реалізовано у вигляді класу «NeuralNetwork» модуль, що забезпечує налаштування параметрів нейронної мережі з врахуванням її архітектури.

2. Програмно реалізовані методи прямого поширення та навчання із зворотнім поширенням помилки з використанням сигмоподібної функції активації, що дало змогу забезпечити навчання нейронної мережі.

3. Забезпечено імплементацію модуля нейронної мережі в ігрову комп’ютерну систему та проведено апробацію розроблюваних методів і засобів в ігровій комп’ютерній системі з предметною областю «гри астероїди».

# РОЗДІЛ 4
# ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

У кваліфікаційній роботі магістра розглянуто ПЗ, що являє собою імплементацію інтелектуального модуля з інтелектуальним перехопленням об’єктів в ігровій комп’ютерній системі на основі предметної області «гри астероїди». Оскільки модуль може бути імплементовано в ігрові комп’ютерні системи з іншими предметними областями, слід передбачити безпеку роботи користувача за ПК.

## 4.1. Охорона праці

Загальні вимоги безпеки з охорони праці для користувачів ЕОМ.

ЕОМ та устаткування повинні бути справними і випробуваними відповідно до чинних нормативних актів.

Після закінчення роботи електронно-обчислювальна техніка відключається від електричної мережі. Це ж саме повинно бути здійснено у випадку виникнення аварійної ситуації під час роботи.

Під час експлуатації ЕОМ забороняється здійснювати ремонт та налагодження їх на робочому місці, працювати на зіпсованій техніці, загромаджувати робочі місця матеріалами, які не використовуються для поточної роботи.

Обслуговування, ремонт та налагодження ЕОМ, інші операції у цьому плані мають здійснюватись тільки при повному відключенні живлення.

У випадку коли ремонтні та інші операції неможливо здійснити при відключеному живленні, необхідно, щоб устаткування, допоміжна апаратура та прилади були заземлені, роботу виконували два або більше працівників з використанням інструментів з ізольованими ручками, а на підлозі були діелектричні килимки.

Ремонт відеотермінала без футляра, а також усі види робіт з відкритим кінескопом повинні проводитися в захисних окулярах або масці.

Режим праці та відпочинку працівників електронно-обчислювальної техніки визначається ДСанНіП 3.3.2-007-98 Вимоги до режимів праці і відпочинку при роботі з ВДТ ЕОМ і ПЕОМ. Через кожні 40-50 хв. роботи необхідно робити 3 – 5 хвилинні перерви для відпочинку. Сумарна тривалість роботи на день не повинна перевищувати 4 год., а на тиждень – 20 год.

До роботи з профілактикою обслуговування, налагодження і ремонту ЕОМ допускаються працівники віком старше 18 років, які пройшли попереднє спеціальне навчання, мають відповідне посвідчення, не мають медичних протипоказань, пройшли інструктаж з охорони праці та пожежної безпеки.

Усі працівники підлягають обов’язковому медичному огляду відповідно до Положення про медичний огляд працівників певних категорій.

Інструкція з охорони праці при роботі з ПК.

1. Загальні положення

1.1 Дана інструкція, розроблена на підставі ДНАОП 5.2.30-1.08-96 і ВСН 4559-88 "Тимчасові санітарні норми і правила для працівників обчислювальних центрів" і рекомендацій учбово - методичного посібника "Охорона праці користувачів комп'ютерних відео дисплейних терміналів".

1.2. До роботи на персональних ЕОМ допускаються особи від 18 років, що не мають протипоказань за результатами попереднього медичного огляду і пройшли інструктаж, навчання й перевірку знань по охороні праці і мають 1 кваліфікаційну групу з електробезпеки.

1.3. Допуск до роботи на персональних ЕОМ осіб до 18 років (практикантів, учнів) здійснюється під керівництвом досвідчених працівників, що мають кваліфікаційну групу з електробезпеці не нижче 3.

1.4. Праця на робочому місці, оснащеному дисплеєм, супроводжується дією наступних небезпечних і шкідливих факторів: напруження зору; гіподинамія;

монотонність праці; підвищений рівень статичної електрики; відбиті бліки екрана дисплея; емоційні перенавантаження; можливість поразки електричним струмом;

1.5. Для зниження й попередження шкідливого впливу вищевказаних факторів необхідно:

– для зниження рівня статичної електрики розташовувати екран дисплея на відстані не ближче 550 - 700 мм. від очей оператора;

– для зниження бліків екран дисплея повинен розташовуватися перпендикулярно світлового потоку від віконних прорізів або від електросвітильників;

– для зниження утоми очей:

а) освітленість робочого місця повинна бути не менш 300 - 500 люкс; яскравість світіння екрана - не менш 100 кл / кв. м.;

б) мінімальний розмір світної точки - не більш 0,6 мм.;

в) контрастність зображення знака - не менш 0,8;

г) частота регенерації - не менш 72 Гц.;

д) для зниження впливу гіподинамії й емоційних перевантажень варто використовувати технологічні перерви і виконувати комплекс фізичних вправ, зазначених у Додатку 1 і 2, п. 10, Тимчасових санітарних норм.

2. Вимоги безпеки перед початком роботи.

2.1. Залишити в гардеробі вуличний одяг, особисті речі.

2.2. Забрати з робочого місця предмети, що не будуть використовуватися в роботі.

Забороняється класти на блоки ЕОМ папір, книги, документи й інші предмети.

2.3. Забороняється, щоб уникнути перевантаження мережі, підключати ЕОМ через трійники разом з іншими електроприладами.

2.4. Уключити, при необхідності, штучне освітлення, настільний світильник.

2.5. Зовнішнім оглядом переконатися в справності сполучних проводів, штепсельних рознімань, шин заземлення й вимикачів, у надійності кріплення захисних кожухів і кришок блоків ЕОМ.

2.6. Перевірити відсутність пилу на екрані дисплея й правильність установки паперу в прийомний лоток принтера. Не допускати забивання пилом і сторонніми предметами вентиляційних отворів для відводу тепла з блоків ЕОМ.

2.7. Відрегулювати висоту сидіння стільця й підставки для ніг. Відрегулювати положення екрана монітора щодо свого поля зору.

2.8. При виявленні несправностей і інших недоліків, що створюють небезпеку або значні незручності в роботі, заявити про це керівникові відділу, ділянки.

3. Вимоги безпеки під час роботи.

3.1. При включенні персональної ЕОМ і освітлення в електромережу, братися тільки за ізольовані частини штепсельних колодок.

3.2. Дотримувати зазначену в інструкції з експлуатації послідовність включення блоків ЕОМ.

3.3. Щоб уникнути розрядів статичної електрики, забороняється доторкатися до екрана дисплея.

3.4. При введенні даних, редагуванні програм, читанні інформації з екрана, безперервна тривалість роботи перед екраном не повинна перевищувати 1 годину з наступними регламентованими перервами по 10 хвилин для відпочинку та виконуванню комплексу фізичних вправ, релаксаційної гімнастики й ауто генного тренування.

3.5. Забороняється при не відключеному електроживленні ЕОМ : розкривати захисні кожухи й кришки блоків ЕОМ, робити регулювання й чищення внутрішніх деталей, змінювати запобіжники; переключати сполучні шнури блоків ЕОМ; змінювати встановлену конфігурацію робочого місця, переставляти блоки ЕОМ; робити вологе прибирання поверхонь комп'ютера; приймати їжу безпосередньо за клавіатурою комп'ютера.

3.6. Категорично забороняється на робочому місці оператора ЕОМ : курити, користатися відкритим вогнем; зберігати легкозаймисті, вибухонебезпечні і хімічно активні, що руйнують ізоляцію, продукти.

4. Дії персоналу в аварійних ситуаціях.

4.1. Ознаками аварійної ситуації на робочому місці оператора ЕОМ є:

– поява збоїв у роботі ЕОМ, заїдання паперу в принтері, зникнення зображення на екрані дисплея,

– коротке замикання, іскріння, появи запаху гару, підвищене нагрівання корпусу, штепсельних рознімань, сполучних проводів, зниження або зникнення напруги в мережі.

4.2. В аварійній ситуації необхідно :

– роботу припинити, ЕОМ відключити від мережі;

– при загорянні використовувати вуглекислотний або порошковий вогнегасники;

– ужити заходів по евакуації людей і наданню першої медичної допомоги постраждалим;

– доповісти про те, що трапилося, керівникові відділу, ділянки;

– при необхідності викликати швидку допомогу, пожежну команду.

5. Вимоги безпеки після закінчення роботи.

5.1. Закінчити працюючі програми, закрити всі каталоги, підготувати комп'ютер до вимикання.

5.2. Відключити ЕОМ і місцеве електроосвітлення від мережі.

5.3. Упорядкувати робоче місце, забрати документи, що використовувалися.

5.4. Переконатися у відсутності пожежної небезпеки.

5.5. Докласти керівникові відділу, ділянки про закінчення роботи і про виявлені під час роботи несправності й інші недоліки.

## 4.2. Безпека життєдіяльності

Під працездатністю людини розуміють можливість її виконувати роботу з необхідною якістю та в установлений час. Працездатність людини залежить як від зовнішніх чинників, так і внутрішнього стану (внутрішні чинники).

До зовнішніх чинників належать: кількість та форма отриманої інформації, зручність робочого місця, характер взаємостосунків в колективі, вплив чинників середовища існування.

До внутрішніх чинників належать: рівень підготовки, тренованість людини та її емоційна стійкість.

У процесі роботи людина переживає різні функціональні стани, які зумовлюють різні рівні її працездатності.

На рис. 4.1 наведено зміни функціонального стану та якості роботи людини у процесі одного трудового циклу (зміни).

Виділяються 4 фази працездатності: пристосування до праці, стійкої працездатності, субкомпенсації, втоми.

Тривалість усіх фаз та усього циклу роботи залежить від рівня підготовки людини до роботи.

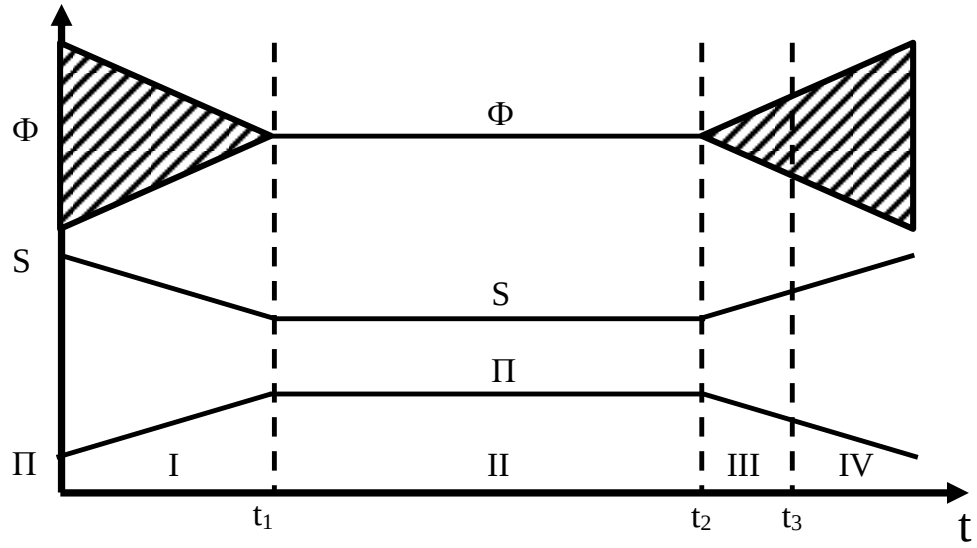



Рис. 4.1 Фази працездатності

Ф – показник функціонального стану;

Б – помилки роботи;

П – продуктивність праці.

Фаза пристосування до праці (0 – 1), - це час протягом якого людина адаптується до майбутніх умов праці. Основний показник поступово досягає свого встановленого значення. Тривалість періоду пристосування організму до умов праці залежить від багатьох чинників, серед яких основними є інтенсивність роботи (чим інтенсивніша робота, тим цей період коротший) та рівень готовності людини до майбутньої роботи.

Значного скорочення фази пристосування до праці можна досягти за рахунок попередньої підготовки людини до роботи (виконання фізичних вправ, адаптації зору, слуху та інші) та шляхом посиленого навчального навантаження. Суть останнього полягає в тому, що оператор перед початком роботи проводить короткочасне тренування щодо розв’язання однієї чи кількох задач підвищеної складності.

Фаза стійкої працездатності ($t_1$ – $t_2$) характеризується найвищою якістю праці при оптимальних рівнях функціонування фізіологічних систем організму. Тривалість цього періоду залежить від інтенсивної роботи. Чим інтенсивніша праця, тим коротний цей період. Найоптимальніша динамічна робота, коли цей період може бути в десятки разів довшим, ніж при статичній діяльності.

На процес стійкої працездатності великий вплив справляють емоції. Негативні (страх, невпевненість, поганий настрій) знижують працездатність. Позитивні (впевненість, спокій, бадьорий настрій) значно продовжують період стійкої працездатності.

Продовження періоду стійкої працездатності можна забезпечити: оптимальним рівнем напруги психофізіологічних функцій; комфортними умовами праці; правильним поєднанням режимів праці та відпочинку; емоційним розвантаженням; використанням тонізуючих напоїв (кава, чай), фармакологічних засобів, зокрема препаратів рослинногопоходження (вітаміни,

препарати, які впливають на енергетичні та метаболічні процеси); інформуванням людини про наслідки її діяльності, наглядом та контролем її роботи.

Практичний досвід свідчить, що вживання легких стимуляторів допомагає знизити сонливість, сприяє підвищенню працездатності на короткий період. Однак активні стимулятори на відповідальних видах робіт здатні викликати негативний ефект – погіршується самопочуття, знижується рухливість та швидкість реакції. Поширене серед населення вживання транквілізаторів, викликаючи заспокоєння та запобігаючи розвитку нервозів, може знизити психічну активність, сповільнити реакції, спричинити апатію та сонливість. [6].

Фаза субкомпенсації ($t_2 – t_3$) розглядається як початок розвитку втоми. В цей період якість праці, ще зберігається на високому рівні, але тільки за рахунок перенапруг і відповідних функцій організму.

Фаза втоми (з моменту у характеризується чітко вираженими зниженнями якості роботи при подальшому погіршенні функціонального стану людини. Об’єктивними показниками втоми є зміна частоти пульсу, дихання, зорової та слухової чутливості.

Наступною фазою життєдіяльності людини повинна бути фаза відновлення працездатності (відпочинку), яка може тривати від 3 до 5 хвилин; 60 – 90 хв. і навіть декілька діб. [20].

# ВИСНОВКИ

Основні результати, отримані у кваліфікаційній роботі магістра полягають в наступному:

1. Проведено аналіз особливостей ігрової комп’ютерної системи за час, якої здійснено узагальнення предметної області;
2. Проаналізовано узагальнення предметної області, що дало змогу здійснити формалізацію об’єкта ігрової комп’ютерної системи;
3. Проведено аналіз технологій реалізації згідно до тенденцій розробки у сучасному світі;
4. Здійснено аналіз структури нейронної мережі, для забезпечення подальшої розробки математичного забезпечення;
5. Здійснено аналіз математичного забезпечення, куди розгляд структури алгоритму навчання нейронної мережі, функції її активації;
6. Побудовано концепт нейронної мережі, яка здійснює математичні операції з матрицями;
7. Розроблено блок-схеми алгоритмів програмної реалізації, які відтворють етапи виконання функцій та їх умов.
8. Реалізовано модуль нейронної мережі, який являє собою клас «NeuralNetwork»;
9. Впроваджено режими доступу до даних класу «NeuralNetwork», за допомогою функцій гетерів та сетерів;
10. Реалізовано методи прямого поширення та навчання зворотнього поширення, окремим з яких є сигмоїдальна функція активації;
11. Здійснено імплементацію модуля нейронної мережі в ігрову комп’ютерну систему;
12. Проведено апробацію розроблюваних методів і засобів в ігровій комп’ютерній системі з предметною областю «гри астероїди».

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Asteroids (video game). URL: https://en.wikipedia.org/wiki/Asteroids_(video_game) (дата звернення 10.10.2022 р.).
2. Modern Web Development: Understanding domains, technologies, and user experience (Developer Reference) 1st Edition, Microsoft Press, 2016, 448p.
3. Probabilistic Machine Learning: An Introduction by Kevin Patrick Murphy. MIT Press, 2012. 864p.
4. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 10.10.2022 р.).
5. Гамільтонів граф. URL: https://uk.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D1%96%D0%BB%D1%8C%D1%82%D0%BE%D0%BD%D1%96%D0%B2_%D0%B3%D1%80%D0%B0%D1%84 (дата звернення 10.10.2022 р.).
6. Artificial Intelligence: A Modern Approach, 4th US edition by Stuart Russell and Peter Norvig. Pearson publisher, 2020, 1136p.
7. Machine Learning Refined: Foundations, Algorithms, and Applications 2nd Edition, by Jeremy Watt. Cambrige University Press, 2020, 594p.
8. Types of Neural Networks and Definition of Neural Network. URL: https://www.mygreatlearning.com/blog/types-of-neural-networks/ (дата звернення 10.10.2022 р.).
9. Feedforward neural network. URL: https://en.wikipedia.org/wiki/Feedforward_neural_network (дата звернення 10.10.2022 р.).
10. JavaScript for Kids: A Playful Introduction to Programming by Nick Morgan, 2014, 336p.
11. Canvas. URL: https://developer.mozilla.org/ru/docs/Web/API/Canvas_API (дата звернення 10.10.2022 р.).

12. JavaScript math function. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/atan2 (дата звернення 10.10.2022 р.).
13. Math behind Artificial Neural Networks. URL: https://medium.com/analytics-vidhya/math-behind-artificial-neural-networks-42f260fc1b25 (дата звернення 10.10.2022 р.).
14. Linear Algebra explained in the context of deep learning. URL: https://towardsdatascience.com/linear-algebra-explained-in-the-context-of-deep-learning-8fcb8fca1494
15. Sigmoid function. URL: https://en.wikipedia.org/wiki/Sigmoid_function (дата звернення 10.10.2022 р.).
16. Estimation of Neurons and Forward Propagation in Neural Net. URL: https://www.analyticsvidhya.com/blog/2021/04/estimation-of-neurons-and-forward-propagation-in-neural-net/#:~:text=There%20are%20three%20steps%20to,loss%20or%20the%20error%20term (дата звернення 10.10.2022 р.).
17. Constructor JavaScript URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/constructor (дата звернення 10.10.2022 р.).
18. Getters JavaScript URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get (дата звернення 10.10.2022 р.).
19. Грибан В.Г., Негодченко О.В. Охорона праці. К.: Центр учбової літератури, 2009. 209 с.
20. Бедрій Я.І. 39 Безпека життєдіяльності: Навчальний посібник. Київ: Кондор, 2009. 286 с.

# ДОДАТКИ

## ДОДАТОК А

Тези конференцій

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ІВАНА ПУЛЮЯ**

**М А Т Е Р І А Л И**

**X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ**

**«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»**

**7–8 грудня 2022 року**

**ТЕРНОПІЛЬ**
**2022**

УДК 004.85
**В. Яцишин, В. Цимбалістий, Вік. Яцишин**
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

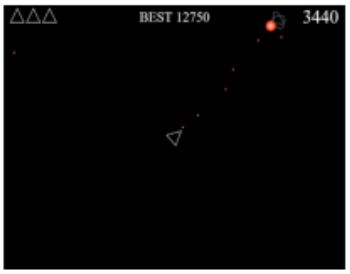
## КОМП'ЮТЕРНІ ІГРИ ЯК СПОСІБ МОДЕЛЮВАННЯ ПОВЕДІНКИ РЕАЛЬНИХ КОМП'ЮТЕРНИХ СИСТЕМ

UDC 004.85
**V. Yatsyshyn, V. Tsymbalistyi, V. Yatsyshyn**

## COMPUTER GAMES AS A WAY OF REAL COMPUTER SYSTEMS BEHAVIOUR MODELLING

Важливу роль при проектуванні комп'ютерних систем відіграє їхнє моделювання. Одним з ефективних способів аналізу поведінки і функціональності складних комп'ютерних систем щодо аналізу траєкторії руху об'єктів є розробка симулятора у вигляді гри. Комп'ютерні ігрові системи можуть бути реалізовані різними технологіями та для різних предметних областей. Як приклад симуляції комп'ютерної системи розглянемо розроблену нами гру «Астероїди».

У даній системі є кілька сутностей, найбільш важливі з яких – корабель та астероїди. Основне завдання, яке потрібно реалізувати у грі полягає у знищенні астероїдів кораблем. Бонуси і рейтингова таблиця користувачів формуються на основі кількості знищених цілей. Для реалізації логіки гри «Астероїди» запропоновано скористатися мовою високого рівня програмування JavaScript, а для забезпечення ефективності поведінки корабля – реалізовано нейронну мережу, в основі якої лежить алгоритм її навчання за допомогою Back Propagation. Нейронна мережа забезпечує ефективність керування кораблем при збитті астероїдів. Функціональність, що покладається на нейронну мережу дозволяє встановити поточне місце об'єкта (астероїда) з певною точністю і здійснює його ураження. На рис. 1 показано один з фрагментів інтерфейсу гри, де корабель виконує потрапляння в астероїд, а також алгоритм навчання нейронної мережі.



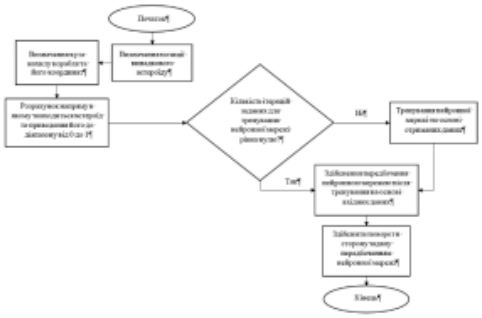



**Рисунок 1.** Фрагмент інтерфейсу гри «Астероїди» та алгоритм навчання нейронної мережі

Запропонована комп'ютерна гра реалізована у середовищі веб-браузера та керується за допомогою нейронної мережі з алгоритмом навчання прямого поширення.

Застосування розробленої моделі гри є доцільним у системах, де необхідна імплементація модуля з інтелектуальним перехопленням об'єктів, що рухаються.

# A Risks management method based on the quality requirements communication method in agile approaches

Vasyl Yatsyshyn[a], Oleh Pastukh[a], Andriy Lutskiv[a] , Viktor Tsymbalistyi[a], Nataliia Martsenko[b]

[a] *Ternopil Ivan Puluj National Technical University, Ruska, 56 street, Ternopil, 46001, Ukraine*
[b] *West Ukrainian National University, Lvivska, 11, Ternopil, 46009, Ukraine*

**Abstract**

In this paper proposed a risks management method based on the quality requirements communication method and SEI risks model. The main value of the method is to increase completeness and traceability of risks during agile software development. This has been achieved by risks identification and its integration to the quality requirements models. Additionally, in the article proposed a formalization of SEI model and the structure of agile software development process, built hierarchical tree with the bipartite vertex at the attributes level (requirement-risk).

**Keywords** [1]

Risk, software, management, agile, quality requirements

## 1. Introduction

Software engineering characterized by using and application of different kinds of software life cycle models. Choosing of the concrete model is defined and dependent from the type of designed system, for instance, software for general market, critical and medical systems, real-time system, etc [1].

Software life cycle models, except for general processes, include a set of activities and sub processes, which are oriented to ensure the quality of the end-user software in the way of taking into account project budget and deadlines.

An important aspects and processes of software engineering are risks identifying, management and monitoring at the different stage of life cycle. Different results would have received at the each life cycle phase in the relation of methods and technologies, which are using to manage risks. In this article, we proposed the method of risks management and monitoring at the stages of software life cycle in case of using agile approach. The main content of this method assumes using and application of requirements communication method [10], which take into account SEI model [2].

In general, a risk of software may defined as a possibility of decreasing quality of final product, increasing in cost of software development, a delay in the completion of the development or a disruption of the project. It has relations to the inefficiency, imperfection, and immaturity of the methods, techniques and technology processes at the different stage of life cycle.

From the point of project development view, the most important risks are technical risks. Considering that, it is very fatefully to identify technical risks in the relation of the requirements, because they form a fundament of the future project development and evolution.

Generally, risks divide by two groups: internal and external [3,4].

Internal risks can be presented like a risk events, which has relations to the internal features of some software project development. Development team wield tools and technologies that help to identify,

ITTAP'2022: 2nd International Workshop on Information Technologies: Theoretical and Applied Problems, November 22–24, 2022, Ternopil, Ukraine
EMAIL: vyatcyshyn@gmail.com (A. 1); oleg.pastuh@gmail.com (A. 2); l.andriy@gmail.com (A. 3); viktortsymbalistuy@gmail.com (A4); nata.martsenko@gmail.com (B)
ORCID: 0000-0002-5517-6359 (A. 1); 0000-0002-0080-7053 (A. 2); 0000-0002-9250-4075 (A. 3); 0000-0002-8651-5616 (A4); 0000-0003-0947-5179 (B)

control and manage these events. For example, risk managers use CASE-tools to estimate a duration of some activities, to define deadlines, to make cost estimation and recruitment.

External risks can be defined as risk events, which appear outside of development team competition and team does not affect to these features (market of software, laws, etc.) [5].

Project Management Institute (PMI) defined system approach to the process of risks management and display it in Project Management Body Of Knowledge (PMBOK).

PMBOK define six main processes of risks management:

- Planning risks management;
- Risks identification;
- Qualitative risks analysis;
- Quantitative risk analysis;
- Planning risks reactions планування реакції на ризики;
- Risks Control and management.

There are many articles and publications devoted to the risks management research [6-8], some of these ideas and concepts are implemented in the standards of software development. The most useful and practical paradigm of risks project management is developed by Software Engineering Institute [7].

## 2. Analysis of risks management approaches

Many formal and non-formal methods are used to identify and evaluate risks of software failures. The choosing one or another method defines by software application (critical software, software for general market) and criticality of consequences when failures are on.

The most famous formal methods of risks failures analysis is Event Tree Analysis (ETA), Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA). Its application is effective in case of complex computer systems designing at the stage of requirements analysis, system architecture design and implementation. These methods help to analyze possible failure modes, develop designs and technical solutions to minimize the consequences of failures.

Event Tree Analysis method provide identification of components in case of failure events which have influence on the security of system functionality. The consequences of each event are traceable until failures of high level will not be define. During creating the chain of events identification, we can calculate probability for each event failure and events combination.

The tree events display a full set of possible consequences for some event of component failure, but do not define reasons of the event.

FMEA and it's edition are generally present as a table methods. Their application can be as additional tools to the previous described methods or can be used as an independent method. FMEA is assigned for the detection of possible kinds of software failures, and conditions of its appearance. These methods are built on the principles of inductive analysis theory, that is effective in the field of identification the conditions of failures appearance in the software components and consequences of defined failures for the whole system.

## 3. SEI model and methodologies

### 3.1 Software risks analysis, identification and management

General taxonomies of risks, which had proposed SEI, are useful and convenient for identification of general sources of software project risks, but they need to be adapted to the concrete situations.

Software Engineering Institute define three methodologies of project risks management:

- Software Risk Evaluation (SRE), which includes formal method of risk identification, analysis, monitoring and elimination; it uses in early stages of software development (before deal) and periodically during software life cycle;
- Continuous Risk Management (CRM) based on some principles of risks management during software life cycle and it is independent from concrete methods and tools of risks estimation and elimination;

- Team Risk Management (TRM) defines additional activities of risks management, which has relation to the collaborative working under the project from development team and stakeholders.

In general, workflow of risks analysis might be present in the next sequence:

- Assigning of an experienced team of experts;
- Preparation of special questions and appointment with experts;
- Choosing of risks analysis technique
- Defining risks factors and its priority
- Creating the mechanism of risks action
- Assigning relation between some risks and the total effect from its action
- Risks distribution between members of the project
- Preparation of risks report and its analysis.

Expert assessment method is usually uses in business projects and it can be implemented in the way of studying opinions of experienced specialists. When this method apply, it is important to define allowable, critical and disastrous factors, which take into account its level and probability. To improve results of expert assessment method, it is convenient to use fuzzy logic during risks level evaluation under the expert assessments.

To reduce the procedure of risks identification uses classification, which was proposed SEI. The classification based with taking into account main processes of software life cycle. It includes the most general areas of risks, which have relations to the software features and characteristics, environment and software development process, project constraints, etc.

## 3.2 Risks classification

The main attention in the article we pay to the software risks at the early stages of life cycle. The most important in this research are internal risks, which has communication with requirements and technical aspects of software development. In the table 1 presented classification of those risks [7].

Many methodologies can be used for gathering information about risks, but the most popular are:

- Expert survey.
- Brain storm.
- Delphi method.
- Crawford cards.

All of the presented before methodologies uses interviewing of experienced specialists, which can make preliminary evaluation of the software project. On our opinion, the most important component in each methodologies is detailed project description. Quality of gathered information define quality of expert conclusions. Because, it is necessary to create rules of software description at different stages of life cycle.

The task of building rules of software description is not trivial, but now there are a few methods, which allow to resolve it with the defined level of acceptability.

Primarily, these are methodologies and methods, which are using at the stage of software specification and design. The most common of them are:

- Diagrams of structured system analysis like ER-diagrams, SADT, IDEF.
- UML diagrams.

Currently, UML is a standard tool of software projects description and support features of object-oriented software development. So far, we can consider that UML is convenient and effective tool of description and gathering input information for risks identification.

The model of complex system, which built using UML notation, describes future software product and includes four main parts:

- logical view;
- implementation view;
- representation of system functionality;
- representation of system structure.

Each part of software description contains a few types of diagrams. These diagrams present some aspect of software model.

**Table 1**
Technical software risks classification

| Class of risks source | Class description | Class element | Attribute of element |
|---|---|---|---|
| Technical aspects of software development | Related to the main processes of life cycle and quality characteristics of software | Requirements features | Stability<br>Completeness<br>Ability to implementation<br>Credibility<br>Reliability<br>Availability<br>Scalability<br>Novelty |
| | | Design and implementation features | Functionality<br>Complexity<br>Interfaces<br>Productivity<br>Testability<br>Hardware constraints<br>Reuse components |
| | | Non-functional and quality characteristics | Maintainability<br>Reliability<br>Safety<br>Security<br>Human factors<br>Usability |

When experts define software risks at the stages of life cycle, they use kinds of diagrams, which are displayed in the table 2.

**Table 2**
UML diagrams application during software project development

| Title of UML diagram | Project description |
|---|---|
| Use Case diagram | Requirements of software project |
| Class diagram | Preliminary software architecture |
| Sequence diagram,<br>Activity diagram | Software algorithms and functions implementation |
| Composite structure diagram | Description of hardware configuration |
| Component diagram | A number of tools, which will be using during software implementation |

UML gives possibility to describe software project and helps experts to receive information in clear and structured view, but it limits their information needs only in technical aspects of software. As a

result, data described with UML is not completeness and does not allow expert to carry out a full risks identification.

According to the SEI risks classification (table 1), UML gives opportunity to evaluate completely or partially only functional requirements and project characteristics. In the table 3, presented attributes of classes (risks), which cannot be identified with UML.

**Table 3**
Risks, which cannot be defined with UML

| Class of risks source | Class description | Class element | Attribute of element |
|---|---|---|---|
| Technical aspects of software development | Related to the main processes of life cycle and quality characteristics of software | Requirements features | Novelty |
| | | Non-functional and quality characteristics | Maintainability<br>Human factors |

During risks identification, experts need to receive as input information, except for detailed project description, reports about previously executed projects.

It is necessary to note, that UML is not a single modelling tool, which allows to describe future software. But when developers uses object-oriented methodology, UML application is more powerful and gives the most advantages.

Except for disadvantages of UML, which have relation to the technical aspects of software production like novelty, human factors and maintainability, there are some another: time, deadlines and project budget. In spite of such UML disadvantages, perspectives of this tool application are very essential to the process of risk identification.

## 3.3 Formal description of SEI model

SEI model defines eighteen the most common risks sources, which can be displayed in the formula 1, as a set of these sources

$$RS = \{rs_1, \dots, rs_{18}\} \quad (1)$$

where $rs_i$ – potential source of risks ($i = 1 \dots 18$).

As a source of risks might be software technical risks, which can be displayed as a set of:

$$TRS = \{rs_1, \dots, rs_7\} \quad (2)$$

where $TRS \subset RS$ – a subset of sources of technical risks, where: $rs_1$ – functional characteristics; $rs_2$ – quality characteristics; $rs_3$ – reliability; $rs_4$ – applicability;$rs_5$ – productivity (time); $rs_6$ – maintainability; $rs_7$ – reuse components.

Except for technical risks, defined a risks subset, which have relation to the project budget:

$$CRS = \{rs_8, \dots, rs_{10}\} \quad (3)$$

where $TRS \subset RS$ – a subset of cost risks, $rs_8$ – a limit of total project budget; rs9 – an unavailable project cost; rs10 – a low degree of realism when estimating project costs.

The success and quality of the project are also affected by the risks, associated with the project planning process. Formally, they can be represented as a subset:

$$PRS = \{rs_{11}, \dots, rs_{13}\} \quad (4)$$

where $PRS \subset RS$ – a subset of planning risks sources, $rs_{11}$ – features and possibilities of plan flexibility changing; rs12 – possibilities of violate defined deadlines at life cycle stages; rs13 – a low level of plan realism at the life cycle phases.

One more risks subset can be defined during project development which have relation to the additional and organizational procedures and processes of life cycle. That subset can be presented in the view:

$$MRS = \{rs_{14}, \dots, rs_{18}\}, \quad (5)$$

where $MRS \subset RS$ – a subset of risks sources for the project management processes and procedures, rs14 – project strategy; rs15 – project planning; rs16 – project evaluation; rs17 – project documentation; rs18 – project forecasting.

In general, we can display SEI model as a hierarchical structure (Fig.1).

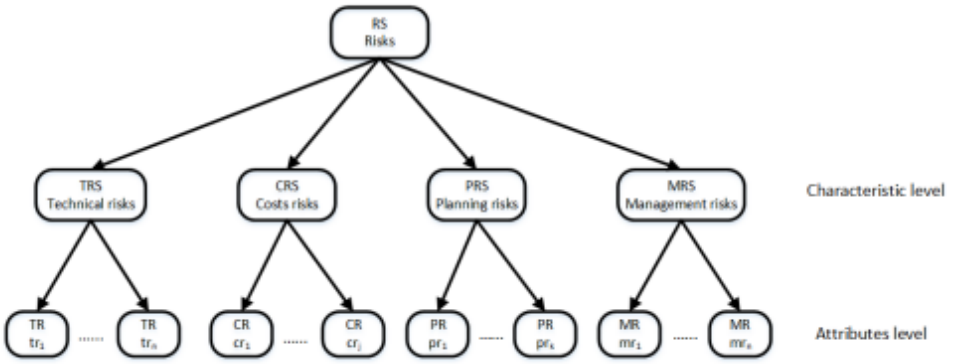



**Figure 1**: SEI model structure

For each subset, presented previously, defined sets of corresponding attributes:

$$\begin{aligned} TR &= \{tr_i\} \in TRS, i = \overline{1 \dots N} \\ CR &= \{cr_j\} \in CRS, i = \overline{1 \dots J} \\ PR &= \{pr_k\} \in PRS, i = \overline{1 \dots K} \\ MR &= \{mr_l\} \in TRS, i = \overline{1 \dots L} \end{aligned} \quad (6)$$

where $TR$ – a set of attributes of technical risks, $CR$ – a set of attributes of costs risks, $PR$ – a set of attributes of planning risks, $MR$ – a set of attributes of additional and organizational risks, $N, J, K, L$ – a number of attributes of corresponding sets of risks.

In the next sections of the article, this model will be implement to the general process of software development using agile approach and integrated with requirements quality model.

## 4. Software quality model and requirements communication method

Software quality model is defined in ISO 25010 standard [9]. It includes three models, which display different aspects of software quality. The structure of the model can be presented as hierarchical model, like SEI model, but it includes more levels of hierarchy. The full structure of ISO 25010 models are showed in the Figure 2.

In the [10,11] had proposed method of requirements definition using ISO 25010 models (Fig. 3). The main value of this method is to provide possibilities of displaying requirements in the structured, standard and unified form.

Generally, the process of software requirements analysis and specification begins with the domain analysis, and after that goes sub processes of gathering, classification and defining priority for each requirement. As we watch in the Fig. 3, requirements presentation executed with supporting of three quality models, which allow to structure and provide communication between them with simultaneous standardization.

The main point of quality model using under the requirements to the software is provide a possibility to display them at the different stages of life cycle. For example, requirements, in the view of quality in use model, are useful at the stage of system testing. External quality model requirements can be used at the stage of software architecture design, internal quality model requirements – at the stage of software development (programming code).

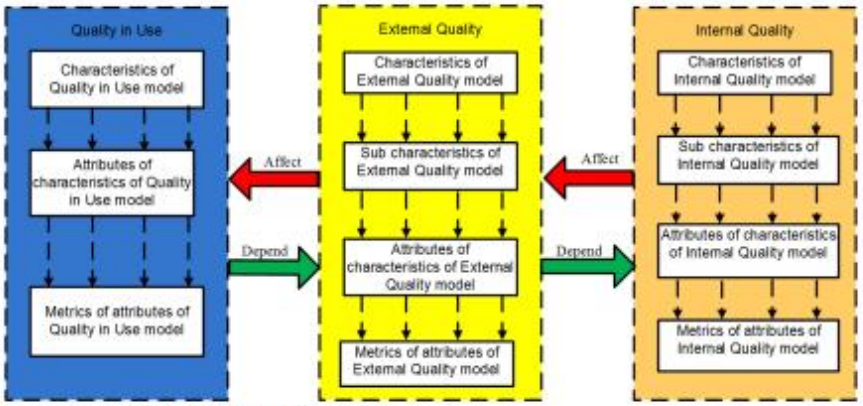



**Figure 2.** Structure of ISO 25010 quality models

Except for this, in the article [10] developed the method of software requirements communication, which helps to provide requirements tracing at the life cycle stages (Figure 4).

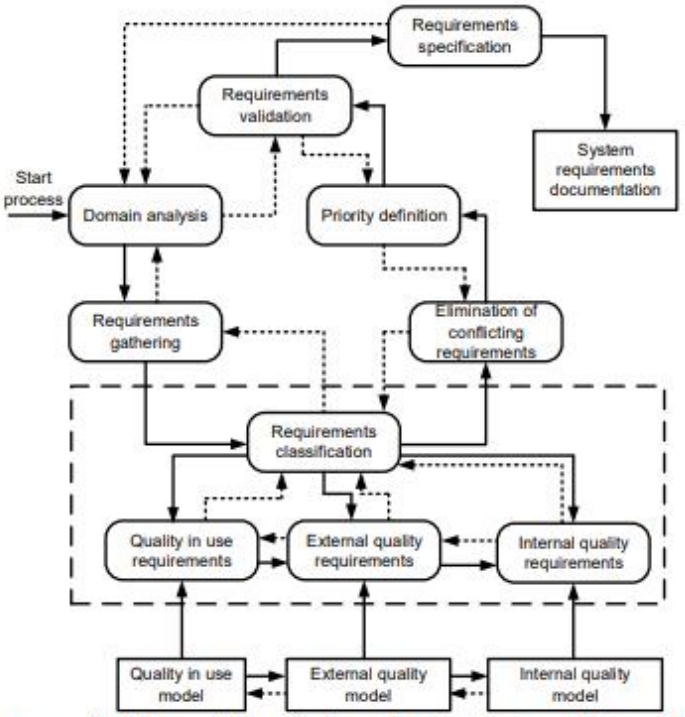



**Figure 3.** The process of requirements identification and analysis using quality models

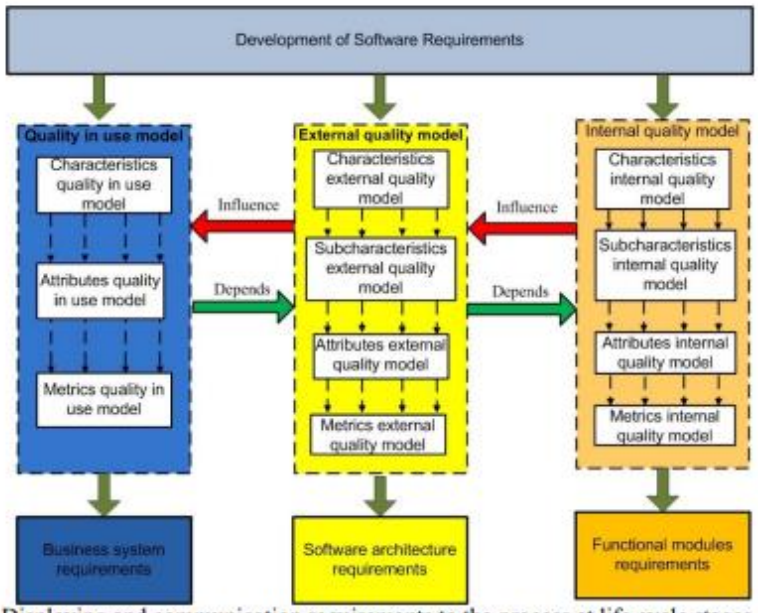



**Figure 4**. Displaying and communication requirements to the process at life cycle stages

After that, it is necessary to identify risks for each requirement [11]. This way, we will increase completeness of technical risks and provide their traceability during software project development. So far, we can also add risks defined before and included to the SEI model using expert classification methods. As a result, we receive hierarchical tree showed in the Figure 5.

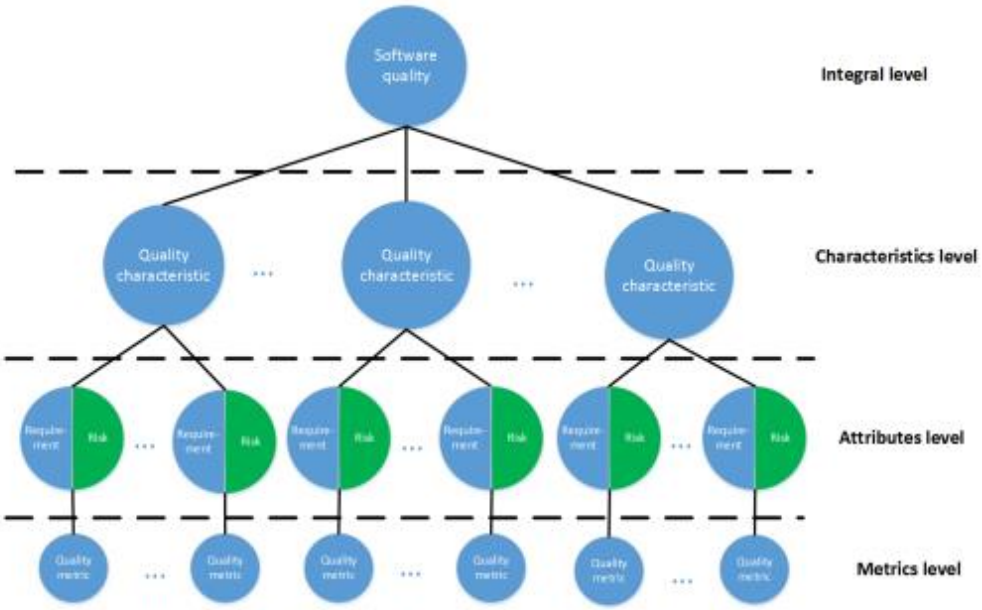



**Figure 5.** Quality model with integrated to the requirements' attributes SEI risks

Quality model with integrated risks we can look as a graph or a tree and we can apply the signature of graph theory to it. This will help us to define another features of risks, calculate their importance. SEI model, quality models with integrated risks, requirements communication method constitute the basis of proposed risks management method. In this article, we consider that relation between requirement and risk is one-to-one. The vertex of the graph is bipartite at the attributes level. But, in more general case, model, presented in the Fig. 5, must be investigate as a hypergraph. Because on each level, except for metrics level, exists connections between elements. In the next section of this article, we define formal description of agile software development process based on the proposed solutions and method.

## 5. Risks management method in agile software development

One of the early processes of software development based on agile approach is customer needs analysis [6]. As a result of this iteration received some necessary data, which can be presented as a set, which include two components: need and time label:

$$Needs = \{n_i, t_j\}, \quad (7)$$

where $n_i$ – a customer need in the concrete time moment, $i = 1, N$ – needs number; $t_j$ – a concrete time moment, $j = 1, T$ – a number of time moments,

At the next step, it is necessary to define detailed requirements to the software, which can be presented as:

$$Req = \{r_i, Needs_{ij}\}, \quad (8)$$

where $r_i$– a requirement to the software, $i = 1, R$ – a number of requirements; $Needs_{ij}$ – customer need, $j \in 1..N$ – a number of needs related to the requirement.

For each requirement, which belong to the $Req$, it is necessary to define it's priority and detailed requirements will be display as:

$$Req = \{r_i, Needs_{ij}, Weight_i\}, \quad (9)$$

where $Weight_i$ – a weight importance coefficient for the *i*-th requirement.

The values of weight importance coefficients may be define by expert methods, for instance, Saaty methods, methods of simple selection algorithms, methods of average weighted, etc.

Application of agile methodologies involves planning of architectural components at the most highest level. At this level, we can describe system architecture as a set of components:

$$ArcHigh = \{comp_i\}, \quad (10)$$

where $comp_i$– a component of software at the conceptual level.

Architecture components implement functional requirements from the set $Req$ (9), so we can note that:

$$\{Req_i\} \rightarrow comp_j, or\ comp_i \rightarrow Req_{ij}, \quad (11)$$

where $\{Req_i\} \in Req, i = 1..H$ – a subset of requirements, which implement some subsystem; $H$ – a number of requirements, which define component at the conceptual level.

To define priority of a software subsystem or a component proposed to calculate average weighted of requirements priorities, which will be implement in them:

$$Weight(comp_i) = \frac{1}{N}\sum_{j=1}^{H} Weight_j \quad (12)$$

where $Weight(comp_i)$– a weight importance coefficient for an architecture component at the conceptual level; $Weight_j$ – a weight importance coefficient for j-th requirement in i-th component. Defined weight importance coefficients for architecture components in future will be include to a sprint planning.

To implement software components we need to plan tasks of their realization. But before it is necessary to make decomposition of large components to the elementary units and after that include them to the sprint. In general, tasks we can present as a set:

$$Task = \{comp_i\{ecomp_{ik}\}, task_{ikj}\}, \quad (13)$$

where $ecomp_{ik}$ – an elementary unit of i-th component, $k = 1..K$, $K$ – a number of elementary units; $task_{ikj}$ – tasks, which need to implement for i-th component development.

Sprint is one of the important part in agile methodologies, which can be displayed as a set:

$$Sprint = \{Task_i, \{Person\}, t_{start}, t_{end}\} \quad (14)$$

where $Task_i$ – a set of tasks during the sprint; $\{Person\}$ – a set of team members who implement tasks; $t_{start}$ – time of sprint starting; $t_{end}$ – time of sprint finishing.

As we proposed before, technical risks are related to the requirements and we can integrate them to the general agile process in the way:

$$Req = \{\{r_i, \{Risk_{iL}\}\}, Needs_{ij}, Weight_i\} \quad , \quad (15)$$

where $Risk_{iL}$ – a set of risks related to the requirement, $i = \overline{1,N}, L = \overline{1,r_i}$

$Risk_{iL}$ belong to the union of sets:

$$Risk_{iL} \in TRS \cup CRS \cup PRS \cup MRS \quad (16)$$

In future, software risks will be taken into account to each sprint and will increase the process of risks management.

## CONCLUSION

In the paper the analysis of software risks management approaches and methods was carried out and defined their main advantages and disadvantages. Application of SEI risks model is very useful during modern software development and it was important to formalize it and integrate to the quality requirements communication method. It provide to improve quality of finish product in the way of effective risks management at the life cycle stages. In this paper we proposed to identify risks according to the requirements and add additional risks defined by SEI model. As a result, we received hierarchical structure, which was integrated with the ISO 25010 quality models. The vertex of the graph is bipartite at the attributes level. But, in more general case, the model must be investigate as a hypergraph and it will be our future task. Also, the paper describe a formal representation of agile software development process structure based on the proposed solutions and method.

## References

[1] Ian Sommerville. Software Engineering, Global Edition. Pearson Higher Ed, 2016, 816 pages.

[2] Fenton N. Risk Assessment and Decision Analysiswith Bayesian Networks. CRC Press. –2012. – P. 33-38.

[3] Soumya Krishnan M. Software Development Risk Aspects and Success Frequency on Spiral and Agile Model /M. Soumya Krishnan // Int. Journal of Innovative Research in Computer and Comm. Eng. – 2015. - Vol. 3. – P. 122-129.

[4] Abdullahi M. Strength and Weakness of Software Risk Assessment Tools. International Journal of Software Engineering and Its Applications. – 2014. – Vol. 8. – № 3. - P. 389-398.

[5] Woody C. Supply-Chain Risk Management: Incorporating Security into Software Development. Software Engineering Institute Carnegie Mellon University. – 2010. – P. 166-178.

[6] Britkin A.I. Model estimate the duration of the iterative software development process. Open education, 2009. – №4. – P. 75.

[7] William R. Project Management Body of Knowledge. PMI Standard Committee. 2006. 182 p.

[8] Hijazi H. A Review of Risk Management in Different Software Development Methodologies. Int. Journal of Computer Appl. – 2013. – Vol. 48, № 3. – P.1441-1453.

[9] ISO/IEC 25010:2011ю Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

[10] Yatsyshyn V, Kharchenko A, Galay I. The Method of Quality Management Software. Proceeding of the VIIth International Conference "Perspective technologies and methods in MEMS design" 11-14 May 2011 - Polyana, Ukraine: Publishing House Vezha&Co. 2011.- p. 228-230.

[11] Kharchenko A., Bodnarchuk I., Yatcysyn V. The Method for Comparative Evaluation of Software Architecture with Accounting of Trade-offs. American Journal of Information Systems, 2014, Vol. 2, No. 1, 20-25/Режим доступу - http://pubs.sciepub.com/ajis/2/1/5/

# ДОДАТОК Б

## Лістинги програми

----------index----------

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Asteroids</title>
    <style></style>
    <script type="text/javascript" src="./neural-network-03.js"></script>
</head>

<body>
    <canvas id="gameCanvas" width="760" height="570"></canvas>
    <script>
        const FPS = 60; // frames per second
        const FRICTION = 0.7; // friction coefficient of space (0 = no friction,
1 = lots of friction)
        const GAME_LIVES = 3; // starting number of lives
        const LASER_DIST = 0.6; // max distance laser can travel as fraction of
screen width
        const LASER_EXPLODE_DUR = 0.1; // duration of the lasers' explosion in
seconds
        const LASER_MAX = 10; // maximum number of lasers on screen at once
        const LASER_SPD = 500; // speed of lasers in pixels per second
        const ROID_JAG = 0.4; // jaggedness of the asteroids (0 = none, 1 = lots)
        const ROID_PTS_LGE = 20; // points scored for a large asteroid
        const ROID_PTS_MED = 50; // points scored for a medium asteroid
        const ROID_PTS_SML = 100; // points scored for a small asteroid
        const ROID_NUM = 3; // starting number of asteroids
        const ROID_SIZE = 100; // starting size of asteroids in pixels
        const ROID_SPD = 50; // max starting speed of asteroids in pixels per
second
        const ROID_VERT = 10; // average number of vertices on each asteroid
        const SAVE_KEY_SCORE = "highscore"; // save key for local storage of high
score
        const SHIP_BLINK_DUR = 0.1; // duration in seconds of a single blink
during ship's invisibility
        const SHIP_EXPLODE_DUR = 0.3; // duration of the ship's explosion in
seconds
        const SHIP_INV_DUR = 3; // duration of the ship's invisibility in seconds
        const SHIP_SIZE = 30; // ship height in pixels
        const SHIP_THRUST = 5; // acceleration of the ship in pixels per second
per second
        const SHIP_TURN_SPD = 360; // turn speed in degrees per second
        const TEXT_FADE_TIME = 2.5; // text fade time in seconds
        const TEXT_SIZE = 40; // text font height in pixels

        // developer flags
        const AUTOMATION_ON = true;
        const MUSIC_ON = true;
        const SOUND_ON = true;
        const SHOW_BOUNDING = false; // show or hide collision bounding
        const SHOW_CENTRE_DOT = false; // show or hide ship's centre dot

        // neural network parameters
```

```
        const NUM_INPUTS = 4;
        const NUM_HIDDEN = 20;
        const NUM_OUTPUTS = 1;
        const NUM_SAMPLES = 100000; // number of training samples
        const OUTPUT_LEFT = 0; // expected neural output for turning left
        const OUTPUT_RIGHT = 1; // expected neural output for turning right
        const OUTPUT_THRESHOLD = 0.05; // how close the prediction must be to
commit to a turn
        const RATE_OF_FIRE = 15; // shots per second

        /** @type {HTMLCanvasElement} */
        var canv = document.getElementById("gameCanvas");
        var ctx = canv.getContext("2d");

        // set up sound effects
        var fxExplode = new Sound("sounds/explode.m4a");
        var fxHit = new Sound("sounds/hit.m4a", 5);
        var fxLaser = new Sound("sounds/laser.m4a", 5, 0.5);
        var fxThrust = new Sound("sounds/thrust.m4a");

        // set up the music
        var music = new Music("sounds/music-low.m4a", "sounds/music-high.m4a");
        var roidsLeft, roidsTotal;

        // set up the game parameters
        var level, lives, roids, score, scoreHigh, ship, text, textAlpha;
        newGame();

        // set up the neural network
        var nn, aiShootTime = 0;
        if (AUTOMATION_ON) {
            nn = new NeuralNetwork(NUM_INPUTS, NUM_HIDDEN, NUM_OUTPUTS);

            // train the network
            let ax, ay, sa, sx, sy;
            for (let i = 0; i < NUM_SAMPLES; i++) {

                // random asteroid location (include off-screen data)
                ax = Math.random() * (canv.width + ROID_SIZE) - ROID_SIZE / 2;
                ay = Math.random() * (canv.height + ROID_SIZE) - ROID_SIZE / 2;

                // ship's angle and position
                sa = Math.random() * Math.PI * 2;
                sx = ship.x;
                sy = ship.y;

                // calculate the angle to the asteroid
                let angle = angleToPoint(sx, sy, sa, ax, ay);

                // determine the direction to turn
                let direction = angle > Math.PI ? OUTPUT_LEFT : OUTPUT_RIGHT;

                // train the network
                nn.train(normaliseInput(ax, ay, angle, sa), [direction]);
            }
        }

        // set up event handlers
        document.addEventListener("keydown", keyDown);
        document.addEventListener("keyup", keyUp);

        // set up the game loop
        setInterval(update, 1000 / FPS);
```

```
        function angleToPoint(x, y, bearing, targetX, targetY) {
            let angleToTarget = Math.atan2(-targetY + y, targetX - x);
            let diff = bearing - angleToTarget;
            return (diff + Math.PI * 2) % (Math.PI * 2);
        }

        function createAsteroidBelt() {
            roids = [];
            roidsTotal = (ROID_NUM + level) * 7;
            roidsLeft = roidsTotal;
            var x, y;
            for (var i = 0; i < ROID_NUM + level; i++) {
                // random asteroid location (not touching spaceship)
                do {
                    x = Math.floor(Math.random() * canv.width);
                    y = Math.floor(Math.random() * canv.height);
                } while (distBetweenPoints(ship.x, ship.y, x, y) < ROID_SIZE * 2
+ ship.r);
                roids.push(newAsteroid(x, y, Math.ceil(ROID_SIZE / 2)));
            }
        }

        function destroyAsteroid(index) {
            var x = roids[index].x;
            var y = roids[index].y;
            var r = roids[index].r;

            // split the asteroid in two if necessary
            if (r == Math.ceil(ROID_SIZE / 2)) { // large asteroid
                roids.push(newAsteroid(x, y, Math.ceil(ROID_SIZE / 4)));
                roids.push(newAsteroid(x, y, Math.ceil(ROID_SIZE / 4)));
                score += ROID_PTS_LGE;
            } else if (r == Math.ceil(ROID_SIZE / 4)) { // medium asteroid
                roids.push(newAsteroid(x, y, Math.ceil(ROID_SIZE / 8)));
                roids.push(newAsteroid(x, y, Math.ceil(ROID_SIZE / 8)));
                score += ROID_PTS_MED;
            } else {
                score += ROID_PTS_SML;
            }

            // check high score
            if (score > scoreHigh) {
                scoreHigh = score;
                localStorage.setItem(SAVE_KEY_SCORE, scoreHigh);
            }

            // destroy the asteroid
            roids.splice(index, 1);
            fxHit.play();

            // calculate the ratio of remaining asteroids to determine music tempo
            roidsLeft--;
            music.setAsteroidRatio(roidsLeft / roidsTotal);

            // new level when no more asteroids
            if (roids.length == 0) {
                level++;
                newLevel();
            }
        }

        function distBetweenPoints(x1, y1, x2, y2) {
```

```
    return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
}

function drawShip(x, y, a, colour = "white") {
    ctx.strokeStyle = colour;
    ctx.lineWidth = SHIP_SIZE / 20;
    ctx.beginPath();
    ctx.moveTo( // nose of the ship
        x + 4 / 3 * ship.r * Math.cos(a),
        y - 4 / 3 * ship.r * Math.sin(a)
    );
    ctx.lineTo( // rear left
        x - ship.r * (2 / 3 * Math.cos(a) + Math.sin(a)),
        y + ship.r * (2 / 3 * Math.sin(a) - Math.cos(a))
    );
    ctx.lineTo( // rear right
        x - ship.r * (2 / 3 * Math.cos(a) - Math.sin(a)),
        y + ship.r * (2 / 3 * Math.sin(a) + Math.cos(a))
    );
    ctx.closePath();
    ctx.stroke();
}

function explodeShip() {
    ship.explodeTime = Math.ceil(SHIP_EXPLODE_DUR * FPS);
    fxExplode.play();
}

function gameOver() {
    ship.dead = true;
    text = "Game Over";
    textAlpha = 1.0;
}

function keyDown(/** @type {KeyboardEvent} */ ev) {

    if (ship.dead || AUTOMATION_ON) {
        return;
    }

    switch(ev.keyCode) {
        case 32: // space bar (shoot laser)
            shootLaser();
            break;
        case 37: // left arrow (rotate ship left)
            rotateShip(false);
            break;
        case 38: // up arrow (thrust the ship forward)
            ship.thrusting = true;
            break;
        case 39: // right arrow (rotate ship right)
            rotateShip(true);
            break;
    }
}

function keyUp(/** @type {KeyboardEvent} */ ev) {

    if (ship.dead || AUTOMATION_ON) {
        return;
    }

    switch(ev.keyCode) {
```

```
                case 32: // space bar (allow shooting again)
                    ship.canShoot = true;
                    break;
                case 37: // left arrow (stop rotating left)
                    ship.rot = 0;
                    break;
                case 38: // up arrow (stop thrusting)
                    ship.thrusting = false;
                    break;
                case 39: // right arrow (stop rotating right)
                    ship.rot = 0;
                    break;
            }
        }

        function newAsteroid(x, y, r) {
            var lvlMult = 1 + 0.1 * level;
            var roid = {
                x: x,
                y: y,
                xv: Math.random() * ROID_SPD * lvlMult / FPS * (Math.random() <
0.5 ? 1 : -1),
                yv: Math.random() * ROID_SPD * lvlMult / FPS * (Math.random() <
0.5 ? 1 : -1),
                a: Math.random() * Math.PI * 2, // in radians
                r: r,
                offs: [],
                vert: Math.floor(Math.random() * (ROID_VERT + 1) + ROID_VERT / 2)
            };

            // populate the offsets array
            for (var i = 0; i < roid.vert; i++) {
                roid.offs.push(Math.random() * ROID_JAG * 2 + 1 - ROID_JAG);
            }

            return roid;
        }

        function newGame() {
            level = 0;
            lives = GAME_LIVES;
            score = 0;
            ship = newShip();

            // get the high score from local storage
            var scoreStr = localStorage.getItem(SAVE_KEY_SCORE);
            if (scoreStr == null) {
                scoreHigh = 0;
            } else {
                scoreHigh = parseInt(scoreStr);
            }

            newLevel();
        }

        function newLevel() {
            music.setAsteroidRatio(1);
            text = "Level " + (level + 1);
            textAlpha = 1.0;
            createAsteroidBelt();
        }

        function newShip() {
```

```
    return {
        x: canv.width / 2,
        y: canv.height / 2,
        a: 90 / 180 * Math.PI, // convert to radians
        r: SHIP_SIZE / 2,
        blinkNum: Math.ceil(SHIP_INV_DUR / SHIP_BLINK_DUR),
        blinkTime: Math.ceil(SHIP_BLINK_DUR * FPS),
        canShoot: true,
        dead: false,
        explodeTime: 0,
        lasers: [],
        rot: 0,
        thrusting: false,
        thrust: {
            x: 0,
            y: 0
        }
    }
}

function normaliseInput(roidX, roidY, roidA, shipA) {
    // normalise the values to between 0 and 1
    let input = [];
    input[0] = (roidX + ROID_SIZE / 2) / (canv.width + ROID_SIZE);
    input[1] = (roidY + ROID_SIZE / 2) / (canv.height + ROID_SIZE);
    input[2] = roidA / (Math.PI * 2);
    input[3] = shipA / (Math.PI * 2);
    return input;
}

function rotateShip(right) {
    let sign = right ? -1 : 1;
    ship.rot = SHIP_TURN_SPD / 180 * Math.PI / FPS * sign;
}

function shootLaser() {
    // create the laser object
    if (ship.canShoot && ship.lasers.length < LASER_MAX) {
        ship.lasers.push({ // from the nose of the ship
            x: ship.x + 4 / 3 * ship.r * Math.cos(ship.a),
            y: ship.y - 4 / 3 * ship.r * Math.sin(ship.a),
            xv: LASER_SPD * Math.cos(ship.a) / FPS,
            yv: -LASER_SPD * Math.sin(ship.a) / FPS,
            dist: 0,
            explodeTime: 0
        });
        fxLaser.play();
    }

    // prevent further shooting
    ship.canShoot = false;
}

function Music(srcLow, srcHigh) {
    this.soundLow = new Audio(srcLow);
    this.soundHigh = new Audio(srcHigh);
    this.low = true;
    this.tempo = 1.0; // seconds per beat
    this.beatTime = 0; // frames left until next beat

    this.play = function() {
        if (MUSIC_ON) {
            if (this.low) {
```

```
                    this.soundLow.play();
                } else {
                    this.soundHigh.play();
                }
                this.low = !this.low;
            }
        }

        this.setAsteroidRatio = function(ratio) {
            this.tempo = 1.0 - 0.75 * (1.0 - ratio);
        }

        this.tick = function() {
            if (this.beatTime == 0) {
                this.play();
                this.beatTime = Math.ceil(this.tempo * FPS);
            } else {
                this.beatTime--;
            }
        }
    }

    function Sound(src, maxStreams = 1, vol = 1.0) {
        this.streamNum = 0;
        this.streams = [];
        for (var i = 0; i < maxStreams; i++) {
            this.streams.push(new Audio(src));
            this.streams[i].volume = vol;
        }

        this.play = function() {
            if (SOUND_ON) {
                this.streamNum = (this.streamNum + 1) % maxStreams;
                this.streams[this.streamNum].play();
            }
        }

        this.stop = function() {
            this.streams[this.streamNum].pause();
            this.streams[this.streamNum].currentTime = 0;
        }
    }

    function update() {
        var blinkOn = ship.blinkNum % 2 == 0;
        var exploding = ship.explodeTime > 0;

        // use the neural network to rotate the ship and shoot
        if (AUTOMATION_ON && !ship.dead) {

            // compute the closest asteroid
            let c = 0; // closest index
            let  dist0  =  distBetweenPoints(ship.x,  ship.y,  roids[0].x,
roids[0].y);
            for (let i = 1; i < roids.length; i++) {
                let  dist1  =  distBetweenPoints(ship.x,  ship.y,  roids[i].x,
roids[i].y);
                if (dist1 < dist0) {
                    dist0 = dist1;
                    c = i;
                }
            }
```

```
                // make a prediction based on current data
                let ax = roids[c].x;
                let ay = roids[c].y;
                let sa = ship.a;
                let sx = ship.x;
                let sy = ship.y;
                let angle = angleToPoint(sx, sy, sa, ax, ay);
                let  predict  =  nn.feedForward(normaliseInput(ax,  ay,  angle,
sa)).data[0][0];

                // make a turn
                let dLeft = Math.abs(predict - OUTPUT_LEFT);
                let dRight = Math.abs(predict - OUTPUT_RIGHT);
                if (dLeft < OUTPUT_THRESHOLD) {
                    rotateShip(false);
                } else if (dRight < OUTPUT_THRESHOLD) {
                    rotateShip(true);
                } else {
                    ship.rot = 0; // stop rotating
                }

                // shoot the laser
                if (aiShootTime == 0) {
                    aiShootTime = Math.ceil(FPS / RATE_OF_FIRE);
                    ship.canShoot = true;
                    shootLaser();
                } else {
                    aiShootTime--;
                }
            }

            // tick the music
            music.tick();

            // draw space
            ctx.fillStyle = "black";
            ctx.fillRect(0, 0, canv.width, canv.height);

            // draw the asteroids
            var a, r, x, y, offs, vert;
            for (var i = 0; i < roids.length; i++) {
                ctx.strokeStyle = "slategrey";
                ctx.lineWidth = SHIP_SIZE / 20;

                // get the asteroid properties
                a = roids[i].a;
                r = roids[i].r;
                x = roids[i].x;
                y = roids[i].y;
                offs = roids[i].offs;
                vert = roids[i].vert;

                // draw the path
                ctx.beginPath();
                ctx.moveTo(
                    x + r * offs[0] * Math.cos(a),
                    y + r * offs[0] * Math.sin(a)
                );

                // draw the polygon
                for (var j = 1; j < vert; j++) {
                    ctx.lineTo(
                        x + r * offs[j] * Math.cos(a + j * Math.PI * 2 / vert),
```

```
                        y + r * offs[j] * Math.sin(a + j * Math.PI * 2 / vert)
                    );
                }
                ctx.closePath();
                ctx.stroke();

                // show asteroid's collision circle
                if (SHOW_BOUNDING) {
                    ctx.strokeStyle = "lime";
                    ctx.beginPath();
                    ctx.arc(x, y, r, 0, Math.PI * 2, false);
                    ctx.stroke();
                }
            }

            // thrust the ship
            if (ship.thrusting && !ship.dead) {
                ship.thrust.x += SHIP_THRUST * Math.cos(ship.a) / FPS;
                ship.thrust.y -= SHIP_THRUST * Math.sin(ship.a) / FPS;
                fxThrust.play();

                // draw the thruster
                if (!exploding && blinkOn) {
                    ctx.fillStyle = "red";
                    ctx.strokeStyle = "yellow";
                    ctx.lineWidth = SHIP_SIZE / 10;
                    ctx.beginPath();
                    ctx.moveTo( // rear left
                        ship.x - ship.r * (2 / 3 * Math.cos(ship.a) + 0.5 *
Math.sin(ship.a)),
                        ship.y + ship.r * (2 / 3 * Math.sin(ship.a) - 0.5 *
Math.cos(ship.a))
                    );
                    ctx.lineTo( // rear centre (behind the ship)
                        ship.x - ship.r * 5 / 3 * Math.cos(ship.a),
                        ship.y + ship.r * 5 / 3 * Math.sin(ship.a)
                    );
                    ctx.lineTo( // rear right
                        ship.x - ship.r * (2 / 3 * Math.cos(ship.a) - 0.5 *
Math.sin(ship.a)),
                        ship.y + ship.r * (2 / 3 * Math.sin(ship.a) + 0.5 *
Math.cos(ship.a))
                    );
                    ctx.closePath();
                    ctx.fill();
                    ctx.stroke();
                }
            } else {
                // apply friction (slow the ship down when not thrusting)
                ship.thrust.x -= FRICTION * ship.thrust.x / FPS;
                ship.thrust.y -= FRICTION * ship.thrust.y / FPS;
                fxThrust.stop();
            }

            // draw the triangular ship
            if (!exploding) {
                if (blinkOn && !ship.dead) {
                    drawShip(ship.x, ship.y, ship.a);
                }

                // handle blinking
                if (ship.blinkNum > 0) {
```

```
                    // reduce the blink time
                    ship.blinkTime--;

                    // reduce the blink num
                    if (ship.blinkTime == 0) {
                        ship.blinkTime = Math.ceil(SHIP_BLINK_DUR * FPS);
                        ship.blinkNum--;
                    }
                }
            } else {
                // draw the explosion (concentric circles of different colours)
                ctx.fillStyle = "darkred";
                ctx.beginPath();
                ctx.arc(ship.x, ship.y, ship.r * 1.7, 0, Math.PI * 2, false);
                ctx.fill();
                ctx.fillStyle = "red";
                ctx.beginPath();
                ctx.arc(ship.x, ship.y, ship.r * 1.4, 0, Math.PI * 2, false);
                ctx.fill();
                ctx.fillStyle = "orange";
                ctx.beginPath();
                ctx.arc(ship.x, ship.y, ship.r * 1.1, 0, Math.PI * 2, false);
                ctx.fill();
                ctx.fillStyle = "yellow";
                ctx.beginPath();
                ctx.arc(ship.x, ship.y, ship.r * 0.8, 0, Math.PI * 2, false);
                ctx.fill();
                ctx.fillStyle = "white";
                ctx.beginPath();
                ctx.arc(ship.x, ship.y, ship.r * 0.5, 0, Math.PI * 2, false);
                ctx.fill();
            }

            // show ship's collision circle
            if (SHOW_BOUNDING) {
                ctx.strokeStyle = "lime";
                ctx.beginPath();
                ctx.arc(ship.x, ship.y, ship.r, 0, Math.PI * 2, false);
                ctx.stroke();
            }

            // show ship's centre dot
            if (SHOW_CENTRE_DOT) {
                ctx.fillStyle = "red";
                ctx.fillRect(ship.x - 1, ship.y - 1, 2, 2);
            }

            // draw the lasers
            for (var i = 0; i < ship.lasers.length; i++) {
                if (ship.lasers[i].explodeTime == 0) {
                    ctx.fillStyle = "salmon";
                    ctx.beginPath();
                    ctx.arc(ship.lasers[i].x, ship.lasers[i].y, SHIP_SIZE / 15,
0, Math.PI * 2, false);
                    ctx.fill();
                } else {
                    // draw the eplosion
                    ctx.fillStyle = "orangered";
                    ctx.beginPath();
                    ctx.arc(ship.lasers[i].x, ship.lasers[i].y, ship.r * 0.75, 0,
Math.PI * 2, false);
                    ctx.fill();
                    ctx.fillStyle = "salmon";
```

```
                    ctx.beginPath();
                    ctx.arc(ship.lasers[i].x, ship.lasers[i].y, ship.r * 0.5, 0,
Math.PI * 2, false);
                    ctx.fill();
                    ctx.fillStyle = "pink";
                    ctx.beginPath();
                    ctx.arc(ship.lasers[i].x, ship.lasers[i].y, ship.r * 0.25, 0,
Math.PI * 2, false);
                    ctx.fill();
                }
            }

            // draw the game text
            if (textAlpha >= 0) {
                ctx.textAlign = "center";
                ctx.textBaseline = "middle";
                ctx.fillStyle = "rgba(255, 255, 255, " + textAlpha + ")";
                ctx.font = "small-caps " + TEXT_SIZE + "px dejavu sans mono";
                ctx.fillText(text, canv.width / 2, canv.height * 0.75);
                textAlpha -= (1.0 / TEXT_FADE_TIME / FPS);
            } else if (ship.dead) {
                // after "game over" fades, start a new game
                newGame();
            }

            // draw the lives
            var lifeColour;
            for (var i = 0; i < lives; i++) {
                lifeColour = exploding && i == lives - 1 ? "red" : "white";
                drawShip(SHIP_SIZE + i * SHIP_SIZE * 1.2, SHIP_SIZE, 0.5 *
Math.PI, lifeColour);
            }

            // draw the score
            ctx.textAlign = "right";
            ctx.textBaseline = "middle";
            ctx.fillStyle = "white";
            ctx.font = TEXT_SIZE + "px dejavu sans mono";
            ctx.fillText(score, canv.width - SHIP_SIZE / 2, SHIP_SIZE);

            // draw the high score
            ctx.textAlign = "center";
            ctx.textBaseline = "middle";
            ctx.fillStyle = "white";
            ctx.font = (TEXT_SIZE * 0.75) + "px dejavu sans mono";
            ctx.fillText("BEST " + scoreHigh, canv.width / 2, SHIP_SIZE);

            // detect laser hits on asteroids
            var ax, ay, ar, lx, ly;
            for (var i = roids.length - 1; i >= 0; i--) {

                // grab the asteroid properties
                ax = roids[i].x;
                ay = roids[i].y;
                ar = roids[i].r;

                // loop over the lasers
                for (var j = ship.lasers.length - 1; j >= 0; j--) {

                    // grab the laser properties
                    lx = ship.lasers[j].x;
                    ly = ship.lasers[j].y;
```

```
                    // detect hits
                    if (ship.lasers[j].explodeTime == 0 && distBetweenPoints(ax,
ay, lx, ly) < ar) {

                        // destroy the asteroid and activate the laser explosion
                        destroyAsteroid(i);
                        ship.lasers[j].explodeTime = Math.ceil(LASER_EXPLODE_DUR
* FPS);
                        break;
                    }
                }
            }

            // check for asteroid collisions (when not exploding)
            if (!exploding) {

                // only check when not blinking
                if (ship.blinkNum == 0 && !ship.dead) {
                    for (var i = 0; i < roids.length; i++) {
                        if   (distBetweenPoints(ship.x,   ship.y,   roids[i].x,
roids[i].y) < ship.r + roids[i].r) {
                            explodeShip();
                            destroyAsteroid(i);
                            break;
                        }
                    }
                }

                // rotate the ship
                ship.a += ship.rot;

                // keep the angle between 0 and 360 (two pi)
                if (ship.a < 0) {
                    ship.a += (Math.PI * 2);
                } else if (ship.a >= (Math.PI * 2)) {
                    ship.a -= (Math.PI * 2);
                }

                // move the ship
                ship.x += ship.thrust.x;
                ship.y += ship.thrust.y;
            } else {
                // reduce the explode time
                ship.explodeTime--;

                // reset the ship after the explosion has finished
                if (ship.explodeTime == 0) {
                    lives--;
                    if (lives == 0) {
                        gameOver();
                    } else {
                        ship = newShip();
                    }
                }
            }

            // handle edge of screen
            if (ship.x < 0 - ship.r) {
                ship.x = canv.width + ship.r;
            } else if (ship.x > canv.width + ship.r) {
                ship.x = 0 - ship.r;
            }
            if (ship.y < 0 - ship.r) {
```

```
            ship.y = canv.height + ship.r;
        } else if (ship.y > canv.height + ship.r) {
            ship.y = 0 - ship.r;
        }

        // move the lasers
        for (var i = ship.lasers.length - 1; i >= 0; i--) {

            // check distance travelled
            if (ship.lasers[i].dist > LASER_DIST * canv.width) {
                ship.lasers.splice(i, 1);
                continue;
            }

            // handle the explosion
            if (ship.lasers[i].explodeTime > 0) {
                ship.lasers[i].explodeTime--;

                // destroy the laser after the duration is up
                if (ship.lasers[i].explodeTime == 0) {
                    ship.lasers.splice(i, 1);
                    continue;
                }
            } else {
                // move the laser
                ship.lasers[i].x += ship.lasers[i].xv;
                ship.lasers[i].y += ship.lasers[i].yv;

                // calculate the distance travelled
                ship.lasers[i].dist += Math.sqrt(Math.pow(ship.lasers[i].xv,
2) + Math.pow(ship.lasers[i].yv, 2));
            }

            // handle edge of screen
            if (ship.lasers[i].x < 0) {
                ship.lasers[i].x = canv.width;
            } else if (ship.lasers[i].x > canv.width) {
                ship.lasers[i].x = 0;
            }
            if (ship.lasers[i].y < 0) {
                ship.lasers[i].y = canv.height;
            } else if (ship.lasers[i].y > canv.height) {
                ship.lasers[i].y = 0;
            }
        }

        // move the asteroids
        for (var i = 0; i < roids.length; i++) {
            roids[i].x += roids[i].xv;
            roids[i].y += roids[i].yv;

            // handle asteroid edge of screen
            if (roids[i].x < 0 - roids[i].r) {
                roids[i].x = canv.width + roids[i].r;
            } else if (roids[i].x > canv.width + roids[i].r) {
                roids[i].x = 0 - roids[i].r
            }
            if (roids[i].y < 0 - roids[i].r) {
                roids[i].y = canv.height + roids[i].r;
            } else if (roids[i].y > canv.height + roids[i].r) {
                roids[i].y = 0 - roids[i].r
            }
        }
```

```
        }
    </script>
</body>

</html>

-----------------------------Neural network-----------------------

"use strict";

const LOG_ON = true; // whether or not to show error logging
const LOG_FREQ = 20000; // how often to show error logs (in iterations)

class NeuralNetwork {
    constructor(numInputs, numHidden, numOutputs) {
        this._inputs = [];
        this._hidden = [];
        this._numInputs = numInputs;
        this._numHidden = numHidden;
        this._numOutputs = numOutputs;
        this._bias0 = new Matrix(1, this._numHidden);
        this._bias1 = new Matrix(1, this._numOutputs);
        this._weights0 = new Matrix(this._numInputs, this._numHidden);
        this._weights1 = new Matrix(this._numHidden, this._numOutputs);

        // error logging
        this._logCount = LOG_FREQ;

        // randomise the initial weights
        this._bias0.randomWeights();
        this._bias1.randomWeights();
        this._weights0.randomWeights();
        this._weights1.randomWeights();
    }

    get inputs() {
        return this._inputs;
    }

    set inputs(inputs) {
        this._inputs = inputs;
    }

    get hidden() {
        return this._hidden;
    }

    set hidden(hidden) {
        this._hidden = hidden;
    }

    get bias0() {
        return this._bias0;
    }

    set bias0(bias) {
        this._bias0 = bias;
    }

    get bias1() {
        return this._bias1;
    }
```

```
set bias1(bias) {
    this._bias1 = bias;
}

get weights0() {
    return this._weights0;
}

set weights0(weights) {
    this._weights0 = weights;
}

get weights1() {
    return this._weights1;
}

set weights1(weights) {
    this._weights1 = weights;
}

get logCount() {
    return this._logCount;
}

set logCount(count) {
    this._logCount = count;
}

feedForward(inputArray) {
    // convert input array to a matrix
    this.inputs = Matrix.convertFromArray(inputArray);

    // find the hidden values and apply the activation function
    this.hidden = Matrix.dot(this.inputs, this.weights0);
    this.hidden = Matrix.add(this.hidden, this.bias0); // apply bias
    this.hidden = Matrix.map(this.hidden, x => sigmoid(x));

    // find the output values and apply the activation function
    let outputs = Matrix.dot(this.hidden, this.weights1);
    outputs = Matrix.add(outputs, this.bias1); // apply bias
    outputs = Matrix.map(outputs, x => sigmoid(x));

    return outputs;
}

train(inputArray, targetArray) {
    // feed the input data through the network
    let outputs = this.feedForward(inputArray);

    // calculate the output errors (target - output)
    let targets = Matrix.convertFromArray(targetArray);
    let outputErrors = Matrix.subtract(targets, outputs);

    // error logging
    if (LOG_ON) {
        if (this.logCount == LOG_FREQ) {
            console.log("error = " + outputErrors.data[0][0]);
        }
        this.logCount--;
        if (this.logCount == 0) {
            this.logCount = LOG_FREQ;
        }
    }
```

```
        // calculate the deltas (errors * derivitive of the output)
        let outputDerivs = Matrix.map(outputs, x => sigmoid(x, true));
        let outputDeltas = Matrix.multiply(outputErrors, outputDerivs);

        // calculate hidden layer errors (deltas "dot" transpose of weights1)
        let weights1T = Matrix.transpose(this.weights1);
        let hiddenErrors = Matrix.dot(outputDeltas, weights1T);

        // calculate the hidden deltas (errors * derivitive of hidden)
        let hiddenDerivs = Matrix.map(this.hidden, x => sigmoid(x, true));
        let hiddenDeltas = Matrix.multiply(hiddenErrors, hiddenDerivs);

        // update the weights (add transpose of layers "dot" deltas)
        let hiddenT = Matrix.transpose(this.hidden);
        this.weights1     =     Matrix.add(this.weights1,     Matrix.dot(hiddenT,
outputDeltas));
        let inputsT = Matrix.transpose(this.inputs);
        this.weights0     =     Matrix.add(this.weights0,     Matrix.dot(inputsT,
hiddenDeltas));

        // update bias
        this.bias1 = Matrix.add(this.bias1, outputDeltas);
        this.bias0 = Matrix.add(this.bias0, hiddenDeltas);
    }
}

function sigmoid(x, deriv = false) {
    if (deriv) {
        return x * (1 - x); // where x = sigmoid(x)
    }
    return 1 / (1 + Math.exp(-x));
}

/**********************
    MATRIX FUNCTIONS
**********************/

class Matrix {
    constructor(rows, cols, data = []) {
        this._rows = rows;
        this._cols = cols;
        this._data = data;

        // initialise with zeroes if no data provided
        if (data == null || data.length == 0) {
            this._data = [];
            for (let i = 0; i < this._rows; i++) {
                this._data[i] = [];
                for (let j = 0; j < this._cols; j++) {
                    this._data[i][j] = 0;
                }
            }
        } else {
            // check data integrity
            if (data.length != rows || data[0].length != cols) {
                throw new Error("Incorrect data dimensions!");
            }
        }
    }

    get rows() {
        return this._rows;
```

```
}

get cols() {
    return this._cols;
}

get data() {
    return this._data;
}

// add two matrices
static add(m0, m1) {
    Matrix.checkDimensions(m0, m1);
    let m = new Matrix(m0.rows, m0.cols);
    for (let i = 0; i < m.rows; i++) {
        for (let j = 0; j < m.cols; j++) {
            m.data[i][j] = m0.data[i][j] + m1.data[i][j];
        }
    }
    return m;
}

// check matrices have the same dimensions
static checkDimensions(m0, m1) {
    if (m0.rows != m1.rows || m0.cols != m1.cols) {
        throw new Error("Matrices are of different dimensions!");
    }
}

// convert array to a one-rowed matrix
static convertFromArray(arr) {
    return new Matrix(1, arr.length, [arr]);
}

// dot product of two matrices
static dot(m0, m1) {
    if (m0.cols != m1.rows) {
        throw new Error("Matrices are not \"dot\" compatible!");
    }
    let m = new Matrix(m0.rows, m1.cols);
    for (let i = 0; i < m.rows; i++) {
        for (let j = 0; j < m.cols; j++) {
            let sum = 0;
            for (let k = 0; k < m0.cols; k++) {
                sum += m0.data[i][k] * m1.data[k][j];
            }
            m.data[i][j] = sum;
        }
    }
    return m;
}

// apply a function to each cell of the given matrix
static map(m0, mFunction) {
    let m = new Matrix(m0.rows, m0.cols);
    for (let i = 0; i < m.rows; i++) {
        for (let j = 0; j < m.cols; j++) {
            m.data[i][j] = mFunction(m0.data[i][j]);
        }
    }
    return m;
}
```

```
    // multiply two matrices (not the dot product)
    static multiply(m0, m1) {
        Matrix.checkDimensions(m0, m1);
        let m = new Matrix(m0.rows, m0.cols);
        for (let i = 0; i < m.rows; i++) {
            for (let j = 0; j < m.cols; j++) {
                m.data[i][j] = m0.data[i][j] * m1.data[i][j];
            }
        }
        return m;
    }

    // subtract two matrices
    static subtract(m0, m1) {
        Matrix.checkDimensions(m0, m1);
        let m = new Matrix(m0.rows, m0.cols);
        for (let i = 0; i < m.rows; i++) {
            for (let j = 0; j < m.cols; j++) {
                m.data[i][j] = m0.data[i][j] - m1.data[i][j];
            }
        }
        return m;
    }

    // find the transpose of the given matrix
    static transpose(m0) {
        let m = new Matrix(m0.cols, m0.rows);
        for (let i = 0; i < m0.rows; i++) {
            for (let j = 0; j < m0.cols; j++) {
                m.data[j][i] = m0.data[i][j];
            }
        }
        return m;
    }

    // apply random weights between -1 and 1
    randomWeights() {
        for (let i = 0; i < this.rows; i++) {
            for (let j = 0; j < this.cols; j++) {
                this.data[i][j] = Math.random() * 2 - 1;
            }
        }
    }
}
```