

ДОДАТОК А

Перелік посилань відповідно до наукових досліджень кафедри

18. Danylenko, S., Smelyakov, K., Chupryna, A. Methods of Digital-To-Analog Conversion for Reproduction of Sound Waves. 2022 IEEE 9th International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2022 - Proceedings, 2022.

19. Smelyakov, K., Smelyakov, S., Chupryna, A. Adaptive edge detection models and algorithms. Studies in Computational Intelligence, 2020.

ДОДАТОК Б

Звіт результатів Перевірки на унікальність тексту



Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016354997

Дата перевірки:
13.06.2024 06:32:06 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2024 06:34:34 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм-22-6_Слободяник_О_В_скорочений

Кількість сторінок: 37 Кількість слів: 7349 Кількість символів: 56898 Розмір файлу: 762.00 KB ID файлу: 1016157910

0%
Схожість

Збіги відсутні

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Рисунок Б.1 – Результати перевірки

ДОДАТОК В

Апробація результатів роботи

<https://openarchive.nure.ua/entities/publication/392d0519-626f-46d9-a8da-d5e16f5a12e9>

УДК 004.514

DOI: <https://doi.org/10.30837/IYF.IIS.2024.498>

**БАГАТОКРИТЕРІАЛЬНА ОПТИМІЗАЦІЯ МЕХАНІКИ
ВИТРИВАЛОСТІ ДЛЯ ІГР РІЗНИХ ЖАНРІВ**

Слободяник О. В.

Науковий керівник – к.т.н., доц. Чуприна А. С.
Харківський національний університет радіоелектроніки, каф. ПІ,
м. Харків, Україна
e-mail: oleksandr.slobodianyuk@nure.ua

This research highlights the importance of multicriteria optimization in the gaming industry to enhance gameplay and immersion. It examines the mechanics of realistic endurance in games like "Project Zomboid" and "Squad," identifying key criteria such as fatigue, injuries, hunger, and jumps. The authors emphasize the significance of selecting criteria that support immersion without negatively affecting gameplay. They conclude that multicriteria optimization is the key to synergizing all endurance parameters in the gaming environment.

В останні часи в ігровій індустрії тема багатокритеріальної оптимізації стає актуальним і перспективним дослідженням. Сучасна індустрія відеоігор постійно зростає, вимоги до якості продуктів та їх геймплею стають вищими, а ігрові консолі та персональні комп'ютери стають все більш потужними. Через таку зміну розвитку та технологій сучасні розробники повинні використовувати складніші механіки, які, в свою чергу, потребують наявності більшої кількості критеріїв. Саме тут з'являється потреба в багатокритеріальній оптимізації. Ця область досліджень відкриває нові перспективи для створення унікальних ігор, сприяючи покращенню відтворюваності та іммерсії в ігровому середовищі.

Багатокритеріальний характер виникає у випадках, коли потрібно оцінити якість моделюючого процесу за кількома критеріями одночасно [1]. У випадку різних механік в іграх, а саме механіки реалістичної витривалості, можна виділити велику кількість критеріїв: вага головного персонажу, його показники витривалості та сили, вага його інвентарю, частота стрибків, частота бігу, наявність речей в руках тощо. Серед цього треба виокремити потрібні критерії для розробника, базуючись на вимогах жанру, геймплею та складності. Отже, метою роботи є абстракція необхідних критеріїв для реалістичної витривалості, їх багатокритеріальна оптимізація та визначення ступенів впливу кожного критерію на параметр витривалості.

Першочергово виокремлено найпопулярніші критерії витривалості. Для цього було проаналізовано продукти ігрового ринку, які мають авторитет серед гравців, як реалістичні ігри, зокрема: «Project Zomboid» від розробника The Indie Stone та «Squad» від розробника Offworld Industries. Гра «Project Zomboid» має дуже реалістичний геймплей, параметр витривалості впливає безпосередньо на геймплей: на

пересування по карті, можливість використовувати зброю ближнього бою, перетинати паркан та різні перешкоди. Вплив на витривалість поділяється на дві категорії – зменшення запасу витривалості та пряме зменшення очок витривалості. Кількість очок витривалості та її запасу приховано від гравців, а лише сигналізує індикаторами ті критерії, які зараз активні та впливають на витривалість. Основні критерії на вплив параметра витривалості:

-знесилення, сонливість, поранення, голод, спрага, страх, надмірна вага – впливають на максимальний запас очок витривалості, а точніше прискорення їх витрачання.

-біг, долаття перешкод, стрибки, фізичні вправи, взаємодія з навколишнім середовищем – витрачають напряму очки витривалості.

Гра «Squad» має більшу іммерсивність, але меншу реалістичність механіки витривалості. Параметр витривалості впливає лише на пересування по карті, а головними критеріями впливу є: біг – витрачає очки витривалості, поранення – впливає на швидкість витрачання очок витривалості. Отже, основними критеріями параметру витривалості треба виділити такі, що не будуть негативно впливати на геймплей гри та наблизити гравця до максимальної іммерсії [2]. Критерії, які впливають на збільшення або зменшення швидкості витрачання очок витривалості: вага інвентарю, знесилення, голод, спрага, поранення. Вони використовуються, як окремі формули, що впливають на загальний коефіцієнт витрачання витривалості.

Критерії, які витрачають витривалість: біг угору, біг згори, біг по площині, стрибки, присідання. Вони використовуються як реалізація механіки витривалості. Більш складні дії мають більший коефіцієнт, який впливає на витрачання очок витривалості. Наприклад, біг угору буде більш затратним, ніж біг по площині, тобто він матиме більший коефіцієнт витрачання. Критерії, які витрачають витривалість та які зменшують або збільшують швидкість її витрачання, об'єднуються в одну формулу підрахунку витривалості.

Всі ці параметри взаємопов'язані – голод впливає на максимальну вагу інвентарю та знесилення, спрага впливає на знесилення тощо. Можемо зробити висновок, що для синергічної роботи всіх цих параметрів треба використовувати багатокритеріальну оптимізацію. Вона допомагає покращити геймплей та іммерсію шляхом оптимізації визначених критеріїв, що впливають на механіку витривалості в різних жанрах ігор.

Список використаних джерел:

1. Conference Paper Machine Learning Models Efficiency Analysis for Image Classification / Problem Smelyakov, K., Honchar, Y., Bohomolov, O., Chupryna, A. – CEUR Workshop, 3171, pp 942–959, 2022.
2. CORLEY, S. C., Games with Vector Payoff, Journal of Optimization Theory and Applications, Vol. 47, pp. 491–498, 1985.

ДОДАТОК Г

Слайди презентації

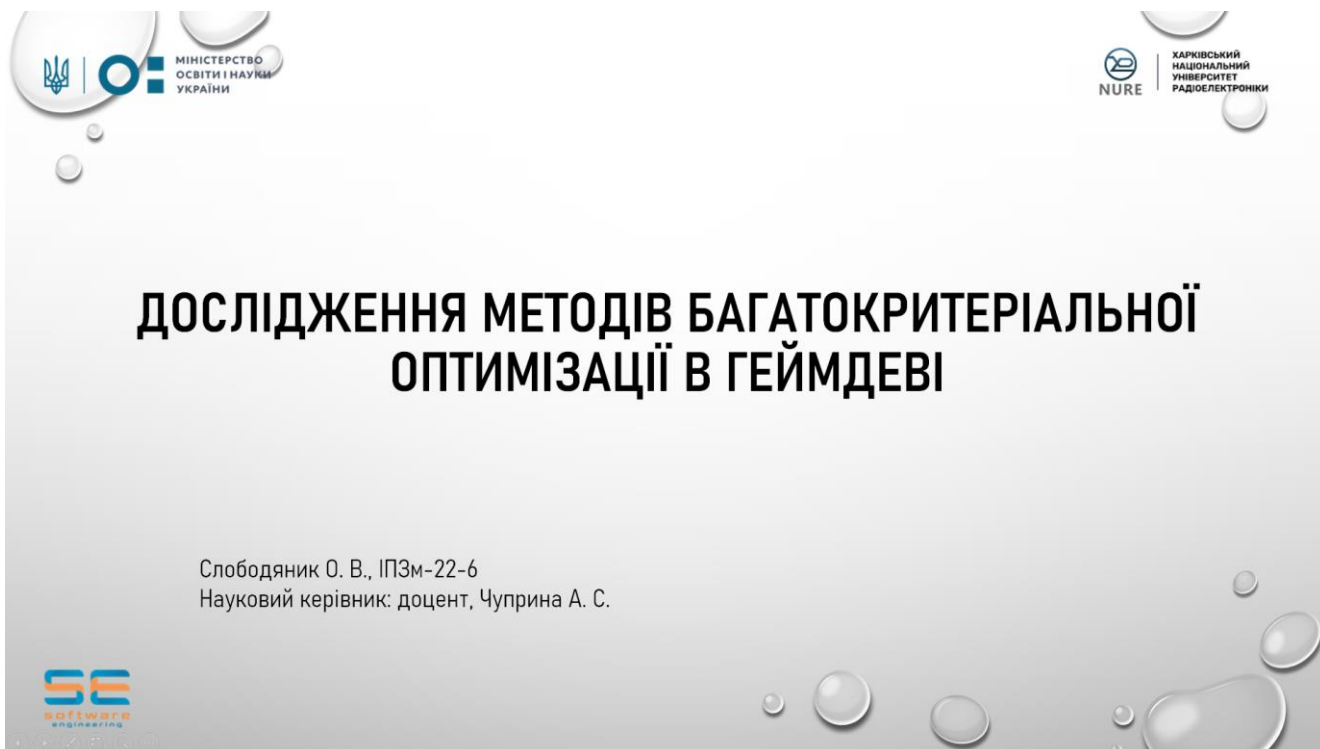


Рисунок Г.1 – Слайд 1

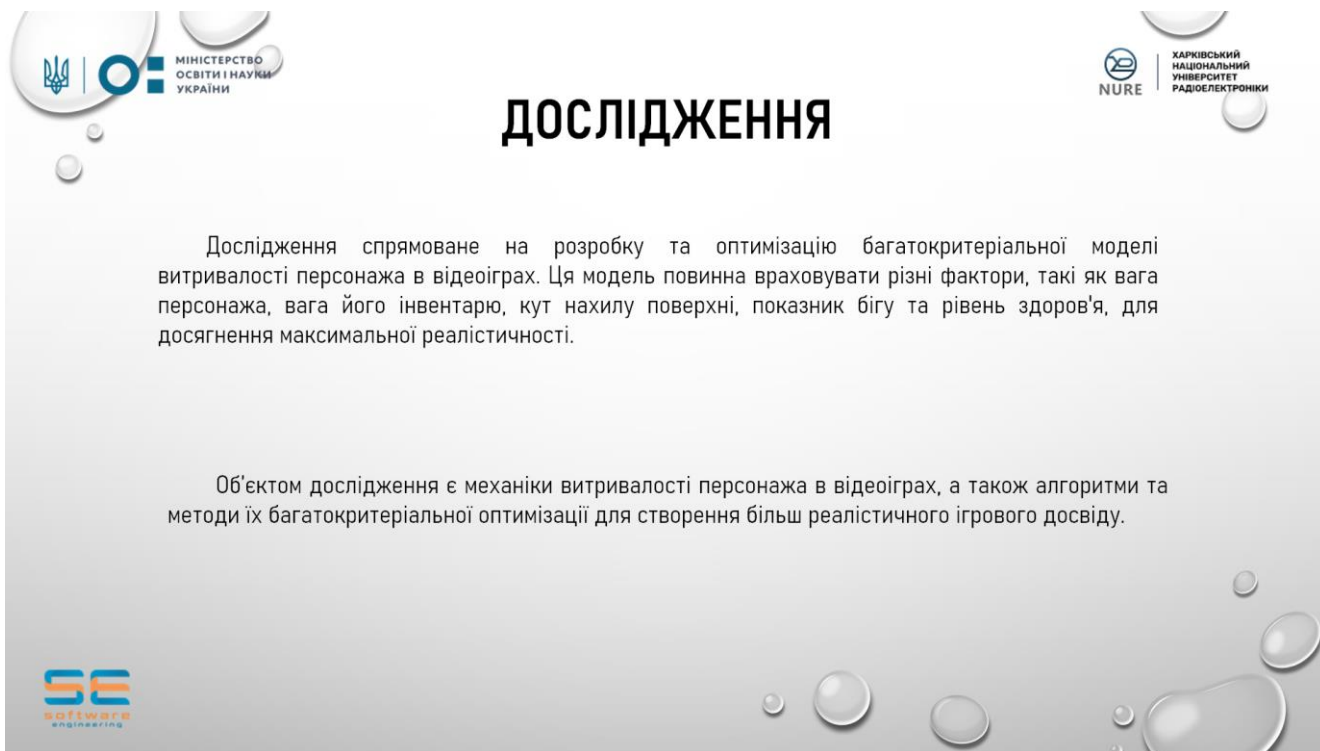


Рисунок Г.2 – Слайд 2

АНАЛІЗ АНАЛОГІВ

Основні джерела включають наукові статті, технічну документацію та практичні посібники з розробки ігор, таких як «Escape from Tarkov» та «Arma 3». Важливими теоретичними основами є праці з фізичної моделі персонажів, оптимізації ігрових механік та алгоритмів симуляції фізичних навантажень.

У більшості наявних моделей витривалості не враховуються всі критичні параметри або їх взаємодія. Існує необхідність у комплексному підході, який би інтегрував усі ключові фактори для забезпечення більш реалістичного ігрового досвіду.



Рисунок Г.3 – Слайд 3

ПОСТАНОВКА ЗАДАЧІ

Задачі:

- розробити технічне завдання щодо магістерського дослідження;
- знайти та проаналізувати теоретичний матеріал щодо теми та напрямку дослідження;
- проаналізувати продукти-аналоги, які використовують схожі алгоритми та механіки;
- визначити основні параметри, які будуть використовуватися для проектування реалістичного алгоритму механіки витривалості;
- спроектувати алгоритм-формулу та оптимізувати її функції та параметри;
- створення прототипу для проведення експерименту;
- проведення експерименту щодо оптимального рішення для створення алгоритму-формули;
- запис результатів експерименту щодо підбору різних значень параметрів та їх відношень (критеріїв);
- зробити висновки щодо проведеного аналізу та дослідження.

Рисунок Г.4 – Слайд 4




МЕТОДОЛОГІЯ



Методологія включала аналіз існуючих моделей витривалості, розробку теоретичної багатокритеріальної моделі, її впровадження у прототипі гри та тестування за допомогою експериментів.

Використані інструменти включають Unity для розробки прототипу, C# для програмування ігрової логіки, а також методи чисельного аналізу для оптимізації моделі.






Рисунок Г.5 – Слайд 5

ОПТИМІЗАЦІЯ ПАРАМЕТРІВ

$$WK_p = \frac{W_p - W_{pmin}}{W_{pmax} - W_{pmin}} + 1$$

$$WK_i = \frac{W_i}{W_{imax}}$$

$$AK_{slope} = \frac{A_{slope} + A_{maxslope}}{A_{maxslope}} - 1$$

$$S_c = 100 * \left((k_1 * WK_p + k_2 * WK_i + k_3 * AK_{slope}) * R_k \right) * \frac{H_c}{H_{max}}$$

Створення загальної формули підрахунку витривалості гравця вимагає інтеграції кількох ключових параметрів, таких як вага персонажа, вага його інвентарю, кут нахилу поверхні, показник бігу та показник здоров'я. Кожен з цих параметрів має свій вплив на витривалість, а їх взаємодія створює комплексну модель поведінки персонажа. Для цього були створені окремі формули для кожного параметру, які враховують специфічні впливи на витривалість, а потім ці формули комбінуються в єдину загальну формулу.


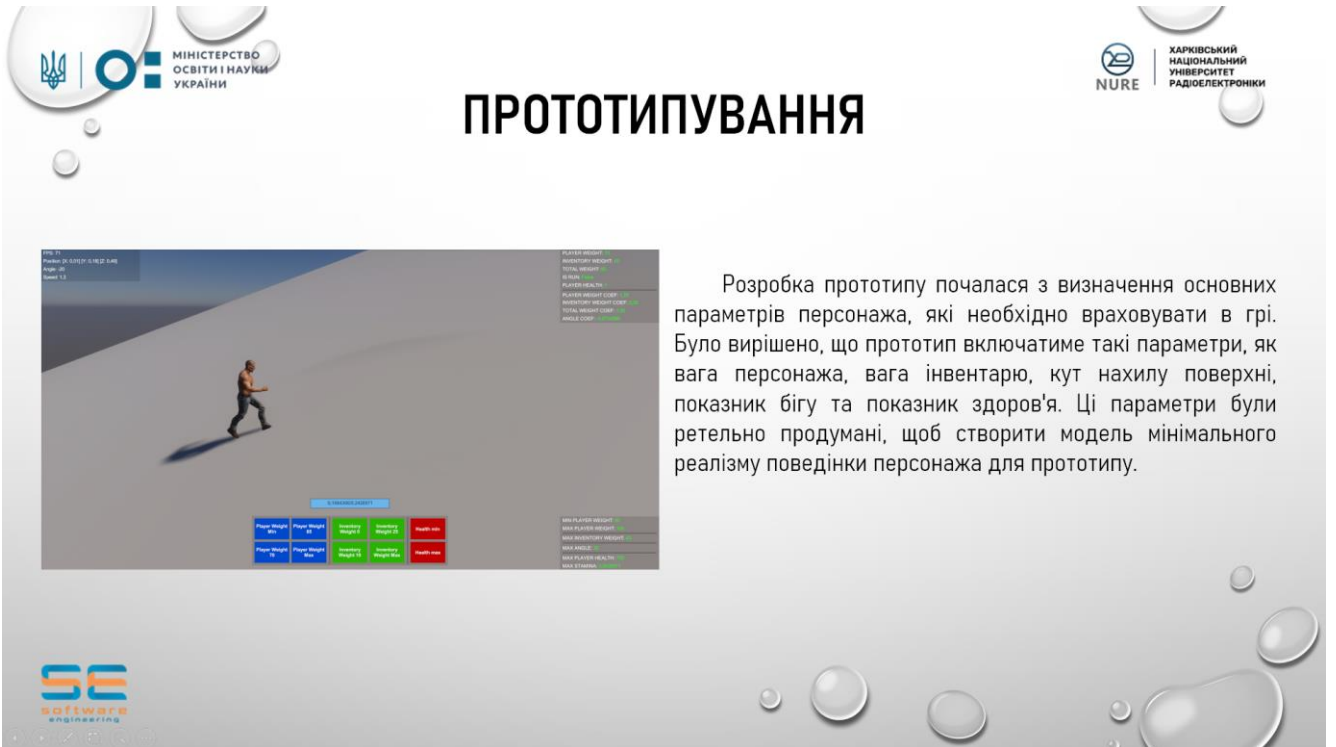


Рисунок Г.6 – Слайд 6



Розробка прототипу почалася з визначення основних параметрів персонажа, які необхідно враховувати в грі. Було вирішено, що прототип включатиме такі параметри, як вага персонажа, вага інвентарю, кут нахилу поверхні, показник бігу та показник здоров'я. Ці параметри були ретельно продумані, щоб створити модель мінімального реалізму поведінки персонажа для прототипу.

Рисунок Г.7 – Слайд 7

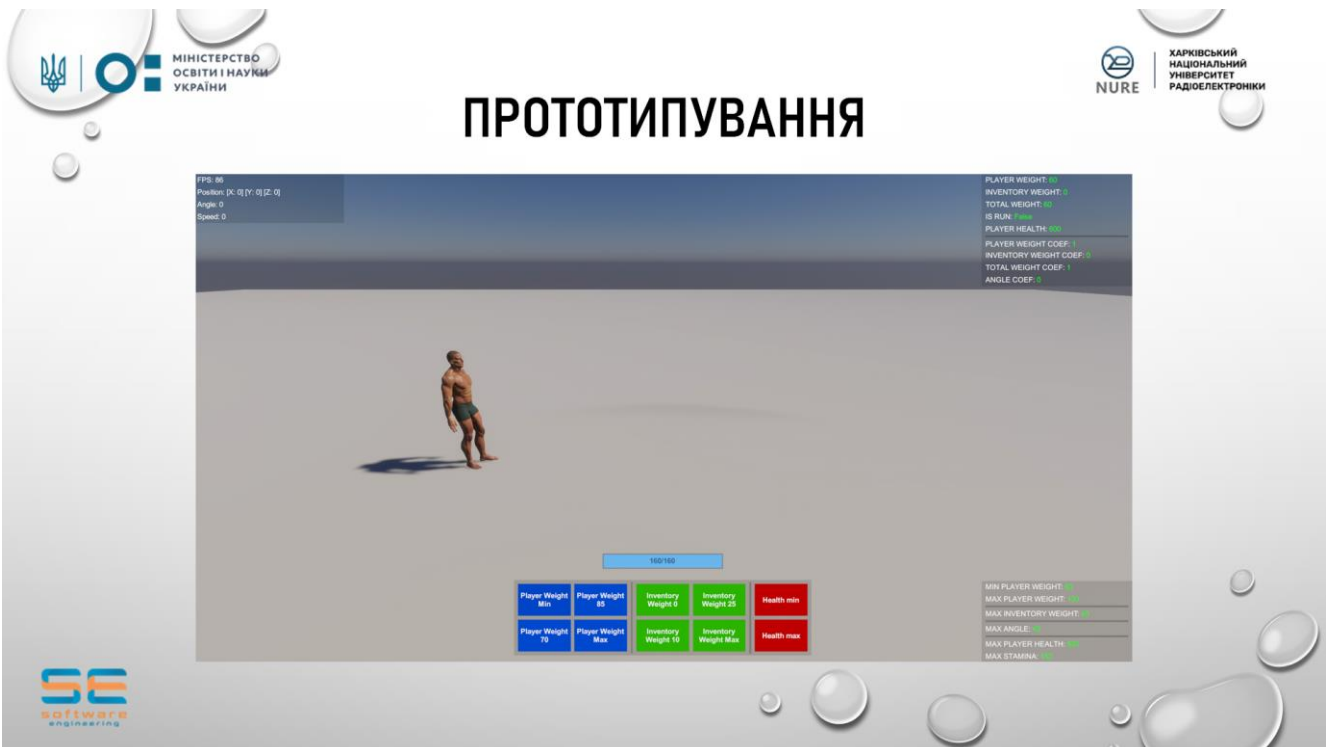


Рисунок Г.8 – Слайд 8

ЕКСПЕРИМЕНТ



Рисунок Г.9 – Слайд 9

ЕКСПЕРИМЕНТ



Рисунок Г.10 – Слайд 10

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ

№	Вага персонажу	Вага інвентарю	Відсоток здоров'я	Біг	Кут нахилу	Результат
1	60	0	1%	+	0	9,71
2	60	0	100%	-	0	13,01
3	60	25	100%	+	-20	16,06
4	60	40	1%	-	15	4,39
5	70	0	100%	+	0	10,78
6	70	10	100%	-	10	12,04
7	70	10	1%	+	10	9,07
8	70	40	100%	-	-20	3,72
9	100	0	100%	+	18	12,74
10	100	25	1%	-	0	5,77
11	100	40	100%	+	-10	6,54
12	100	40	100%	-	-8	6,24

Рисунок Г.11 – Слайд 11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

УДК 004.514 DOI: <https://doi.org/10.38871/1918-2024.498>
БАГАТОКРИТЕРІАЛЬНА ОПТИМІЗАЦІЯ МЕХАНІКИ ІНТЕРВАЛІВ ДІЙ В ГРИВНИХ САНІТІ
 Олександр О. В.
 Науковий керівник – к.т.н., доц. Чурбан А. С.
 Харківський національний університет радіоелектроніки, каф. ПП, м. Харків, Україна
 e-mail: oalexandr.dobrodanuk@nure.ua

This research highlights the importance of multicriteria optimization in the gaming industry to enhance gameplay and immersion. It examines the mechanics of realistic endurance in games like "Project Zomboid" and "Squad," identifying key criteria such as fatigue, hunger, and thirst. The authors emphasize the significance of selecting criteria that support immersion without negatively affecting gameplay. They conclude that multicriteria optimization is the key to overcoming all endurance parameters in the gaming environment.

В останні часи в грівній індустрії тема багатокритеріальної оптимізації стає актуальною і перспективною дослідженням. Сучасні інді-ігри наводять постійно зростаючі вимоги до якості продукції та їх геймплею стають важливішими, а гравці вимагають повноцінної взаємодії з ігровим середовищем. Через таку зміну розвитку та тематичної сфери розробники іноді використовують складні механізми, які, з одного боку, поглиблюють глибину імітації критеріїв. Саме тут і виникає потреба в багатокритеріальній оптимізації. Ця область досліджень відкриває нові перспективи для створення унікальних ігор, створюючи покращення кінцевості та інтересу в грі в ігровому середовищі. Багатокритеріальний характер виникає у випадках, коли потрібно оцінити якість моделювання процесу за кількома критеріями одночасно [1]. У випадку ігрових механізмів в іграх, а саме механізмів реалістичної витривалості, можна виділити якнайменше чотири критерії: вага персонажу, його показники витривалості та сил, вага його інвентарю, частота стрибків, частота бігу, кількість речей в руках тощо. Серед інших треба звернути увагу на критерії для розробників, базуючись на вимогах жанру, геймплею та складності. Однак, метою роботи є абстракція необхідних критеріїв для реалістичної витривалості, їх багатокритеріальна оптимізація та визначення ступеня впливу кожного критерію на параметр витривалості.

Перспективною напрямком найближчих критеріїв витривалості. Для цього було проаналізовано продукти грівної індустрії, які мають відповідні сценарії гри, реалістичні ігри, зокрема: «Project Zomboid» від розробника The Indie Stone та «Squad» від розробника Obsidian Entertainment. Гра «Project Zomboid» має дуже реалістичний геймплей, параметр витривалості впливає безпосередньо на геймплей: на

пересування по карті, можливість використовувати зброю близького бою, переміщення персонажа та інші параметри. Вплив на витривалість поділяється на дві категорії: зменшення запасу витривалості та зростаючий очок витривалості. Кількість очок витривалості та її запасу пропорційно від гри, а значення параметрів критеріїв та критеріїв, які враховують та визначають витривалість. Основні критерії на вплив параметри витривалості: інтелектуальна, сонливість, поранення, голод, спраг, стрес, напруга вага – впливають на максимальний запас очок витривалості, а точніше приросту очок витривалості.


«Squad» дозволяє керувати стрілками, фізичні вправи, взаємодія з навколишнім середовищем – витривалість впливає на очок витривалості. Гра «Squad» має більшу імітаційність, але менше реалістичність механіки витривалості. Параметр витривалості впливає лише на пересування по карті, а головним критерієм впливу є біг – витривалість очок витривалості, поранення – впливає на швидкість витривалості очок витривалості. Однак, основним критерієм параметру витривалості треба відзначити те, що не будуть постійно впливати на геймплей ігри та найбільш грати до максимальної імітації [2]. Критерії, які впливають на більшість або меншість показників витривалості очок витривалості: вага інвентарю, інтелектуальна, голод, спраг, поранення. Вони використовуються, як середні фактори, що впливають на загальний коефіцієнт витривалості витривалості.

Критерії, які витривалість витривалості біг, утримувати, біг, утримувати, біг по складних стрібках, прискорення. Вони використовуються як реалістичні механіки витривалості. Більш складні дії мають більший коефіцієнт, який впливає на витривалість очок витривалості. Наприклад, біг утримувати біг біг утримувати, ніж біг по площині, тобто він має менший коефіцієнт витривалості. Критерії, які витривалість витривалості та які зменшують або збільшують витривалість в ігровому середовищі, об'єднуються в одну формулу підручну витривалості.

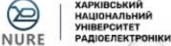
Всі ці параметри взаємозалежні – голод впливає на максимальну вагу інвентарю та інтелектуальну, спраг впливає на інтелектуальну тощо. Можливо профіти висновку, що для спрощеної роботи всіх цих параметрів треба використовувати багатокритеріальну оптимізацію. Вони дозволяють покращити геймплей та імітацію складної оптимізації витривалості критеріїв, що впливають на механіку витривалості в різних жанрах ігор.

Список використаних джерел:
 1. Conference Paper Machine Learning Models Efficiency Analysis for Image Classification / Prof. Dr. İbrahim K. Hossain, Y. Bohemol, O. Shereva, A. – CEUR Workshop, 3171, pp.842-855, 2022.
 2. CORLEY, S. C., Games with Vector Payoff, Journal of Optimization Theory and Applications, Vol. 47, pp. 491–498, 1983.

Рисунок Г.12 – Слайд 12




МІНІСТЕРСТВО
ОСВІТИ І НАУКИ
УКРАЇНИ



ХАРКІВСЬКИЙ
НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНИКИ


ВИСНОВКИ

Результатом дослідження стала формула, яка інтегрує всі зазначені параметри для обчислення витривалості персонажа. Ця формула дозволяє точно визначати рівень витривалості в залежності від змінних умов гри, забезпечуючи більш реалістичну поведінку персонажа. Впровадження цієї формули у ігровий процес дає змогу створювати більш складні та захоплюючі ігрові сценарії, де гравцям необхідно стратегічно планувати свої дії, враховуючи обмеження витривалості.

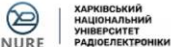


SE
SOFTWARE
ENGINEERING


Рисунок Г.13 – Слайд 13




МІНІСТЕРСТВО
ОСВІТИ І НАУКИ
УКРАЇНИ



ХАРКІВСЬКИЙ
НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНИКИ






Unity



Azure

ДЯКУЮ ЗА УВАГУ!



SE
SOFTWARE
ENGINEERING

Рисунок Г.14 – Слайд 14

ДОДАТОК Д

Експертний висновок результатів перевірки кваліфікаційної роботи на
відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент
(посада)

програмної інженерії
(кафедра)

ПЗМ-22-6
(група)

Слободяник Олександр Валентинович

(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунок	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	

Зауважень немає

Експерт

(підпис)

Олена ОЛІЙНИК

(прізвище, ініціали)

13.06.2024

Рисунок Д.1 – Експертний висновок

ДОДАТОК Е

Скрипти прототипу

Скрипт Player.cs

```

using Game.Items;
using UnityEngine;

namespace Game.Player
{
    public class Player : MonoBehaviour
    {
        [SerializeField] private Inventory inventory;

        [Header("Additional Player`s Parameters")]
        [SerializeField, Range(80f, 120f)] private float maxWeight;
        [SerializeField, Range(45f, 60f)] private float minWeight;
        [SerializeField] private PlayerPlaceholder playerPlaceholder;

        private float _health;
        private float _maxHealth;
        private float _stamina;
        private float _maxStamina;
        private float _weight;
        private bool _isRun;

        private Vector3 _currentPosition;
        private float _currentAngle;
        private float _currentSpeed;
        private float _maxAngleToMove;

        public float Health { get { return _health; } set { _health =
Mathf.Clamp(value, 0f, _maxHealth); } }
        public float MaxHealth
        {
            get
            {
                _maxHealth = 10 * Weight;
                return _maxHealth;
            }
        }
        public float Stamina { get { return _stamina; } set { _stamina =
Mathf.Clamp(value, 0f, _maxStamina); } }
        public float MaxStamina {
            get
            {
                _maxStamina = (100 + 1 * Weight) * (Health / MaxHealth);
                return _maxStamina;
            }
        }
        public float Weight { get { return _weight; } set { _weight =
value; } }
        public float MaxWeight { get { return maxWeight; } set { maxWeight
= value; } }
    }
}

```

```

        public float MinWeight { get { return minWeight; } set { minWeight
= value; } }
        public bool IsRun { get { return _isRun; } set { _isRun = value; }
}

        public Inventory Inventory { get { return inventory; } set {
inventory = value; } }
        public Vector3 CurrentPosition {
            get {
                _currentPosition = transform.position;
                return _currentPosition;
            }
        }
        public float CurrentAngle {
            get {
                Vector3 playerDirection = -transform.up;

                if (Physics.Raycast(transform.position, playerDirection,
out RaycastHit hit))
                {
                    Vector3 surfaceNormal = hit.normal;
                    Vector3 characterForward = transform.forward;
                    float slopeAngle = Vector3.Angle(Vector3.up,
surfaceNormal);

                    Vector3 slopeDirection =
Vector3.ProjectOnPlane(surfaceNormal, Vector3.up).normalized;
                    return _currentAngle = slopeAngle *
Mathf.Sign(Vector3.Dot(slopeDirection, characterForward));
                }
                else
                    return _currentAngle = 0f;
            }
        }
        public float CurrentSpeed { get { return _currentSpeed; } set {
_currentSpeed = value; } }
        public float MaxAngleToMove {
            get {
                _maxAngleToMove = -20 * CalculateOverallWeight(0.4f, 0.6f)
+ 60;

                return _maxAngleToMove;
            }
        }
        public PlayerPlaceholder PlayerPlaceholder { get { return
playerPlaceholder; } set { playerPlaceholder = value; } }

        /// <summary>
        /// Calculates the weight coefficient (if MaxWeight = 100, Weight =
60, K = 0)
        /// </summary>
        /// <returns><b>K</b> - coefficient (from [1] to [2])</returns>
        public float CalculatePlayerWeight() => (Weight - MinWeight) /
(MaxWeight - MinWeight) + 1;

        /// <summary>
        /// Calculates the weight of inventory coefficient
        /// </summary>
        /// <returns><b>K</b> - coefficient (from [0] to [1])</returns>
        public float CalculateInventoryWeight() => Inventory.Weight /
Inventory.MaxWeight;

```

```

    /// <summary>
    /// Calculates the overall weight coefficient
    /// </summary>
    /// <param name="k1">Accounting factor to player weight
coefficient</param>
    /// <param name="k2">Accounting factor to inventory weight
coefficient</param>
    /// <returns><b>K</b> - coefficient (from [1] to [2])</returns>
    public float CalculateOverallWeight(float k1, float k2)
    {
        float sumK = k1 + k2;
        float factK1 = 1 / sumK * k1;
        float factK2 = 1 / sumK * k2;

        float min = factK1;
        float max = 2 * factK1 + factK2;

        return (CalculatePlayerWeight() * factK1 +
CalculateInventoryWeight() * factK2 - min) / (max - min) + 1;
    }

    /// <summary>
    /// Calculates the angle coefficient between _player and surface
    /// </summary>
    /// <returns><b>K</b> - coefficient (from [-1] to [1])</returns>
    public float CalculatePlayersAngle() => (CurrentAngle +
MaxAngleToMove) * 2 / (MaxAngleToMove + MaxAngleToMove) - 1;

    /// <summary>
    /// Calculates the run coefficient
    /// </summary>
    /// <returns><b>K</b> - coefficient (from [1] to [2])</returns>
    public float CalculateRun()
    {
        if (IsRun)
            return 2f;
        else
            return 1f;
    }

    /// <summary>
    /// Calculate spending stamina per second
    /// </summary>
    /// <param name="k1">Accounting factor to player angle
coefficient</param>
    /// <param name="k2">Accounting factor to player weight
coefficient</param>
    /// <param name="k3">Accounting factor to inventory weight
coefficient</param>
    /// <returns></returns>
    private float CalculateStaminaPerSecond(float k1, float k2, float
k3)
    {
        float sumK = k1 + k2 + k3;
        float factK1 = 1 / sumK * k1;
        float factK2 = 1 / sumK * k2;
        float factK3 = 1 / sumK * k3;

```

```

        return 100 * ((factK1 * CalculatePlayerWeight() + factK2 *
CalculateInventoryWeight() + factK3 * CalculatePlayersAngle()) *
CalculateRun()) * (Health / MaxHealth);
    }

    public void SpendStamina() => Stamina -=
CalculateStaminaPerSecond(3f, 3f, 4f) * Time.deltaTime;

    public void RecoverStamina() => Stamina = Mathf.Clamp(Stamina +
CalculatePlayerWeight(), 0f, MaxStamina);
    }
}

```

Скрипт PlayerController.cs

```

using UnityEngine;

namespace Game.Player
{
    public class PlayerController : MonoBehaviour
    {
        [SerializeField] private PlayerParameterStorage playerStorage;
        [SerializeField] private Player playerPrefab;

        private Player _player;

        public Player Player { get { return _player; } }

        public void SelectPerson(int index)
        {
            if (_player != null)
                Destroy(_player.gameObject);

            _player = Instantiate(playerPrefab);

            Instantiate(playerStorage.Players[index].Model,
_player.PlayerPlaceholder.transform);

            _player.Weight = playerStorage.Players[index].Weight;
            _player.Stamina = _player.MaxStamina;
            _player.Health = _player.MaxHealth;
        }

        public void SelectInventory(float weight) =>
_player.Inventory.Weight = weight;

        public void SelectHP(float hp)
        {
            if (hp == -1)
                _player.Health = _player.MaxHealth;
            else
                _player.Health = hp;
        }
    }
}

```