МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра радіоелектронних та біомедичних комп'ютеризованих засобів і технологій

Пояснювальна записка до кваліфікаційної роботи

магістра

(освітньо-кваліфікаційний рівень)

на тему: «Розробка алгоритму автоматичного формування харчового раціону людини»

ХАІ.502.564М/2.22в.163.2005288 ПЗ

Виконав: студент 2 курсу групи №564М/2
Галузь знань 16 Хімічна та біоінженерія
Спеціальність 163 Біомедична інженерія
Освітня програма «Біомедична інформатика та радіоелектроніка»
(код і найменування напряму підготовки)
Білодід В. Г.
(прізвище й ініціали студента)
Керівник: проф. Висоцька О.В.
(прізвище й ініціали)
Рецензент: проф. Лисиченко М. Л.
(прізвище й ініціали)

Харків – 2022

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій
(повне найменування)
Кафедра радіоелектронних та біомедичних комп'ютеризованих засобів і технологій
(повне найменування)
Рівень вищої освіти другий (магістерський)
Галузь знань 16 Хімічна та біоінженерія
Спеціальність 163 Біомедична інженерія
(код та найменування)
Освітня програма Біомедична інформатика та радіоелектроніка
(найменування)

ЗАТВЕРДЖУЮ
Завідувачка кафедри
_____ О.В. Висоцька
(підпис) (ініціали та прізвище)
«___» _____ 2022 р.

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

Білодіду Володимиру Григоровичу
(прізвище, ім'я та по батькові)
1. Тема кваліфікаційної роботи Розробка алгоритму автоматичного формування харчового раціону людини
керівник кваліфікаційної роботи Висоцька О. В., д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № 1815 -уч від «29» листопада 2022 року.
2. Термін подання здобувачем вищої освіти кваліфікаційної роботи 12 грудня 2022 р.
3. Вихідні дані до роботи: Метод блочно альтернативних мереж. Програмні засоби - Visual Studio, Unity 3D, Adobe Photoshop, Adobe After Effects
4. Зміст пояснювальної записки (перелік завдань, які потрібно розв'язати):
4.1 Аналітичний огляд методів та засобів формування харчового раціону людини.
4.2 Розробка алгоритму формування харчового раціону людини.
4.3 Формування харчового раціону людини при паталогії.
4.4 Реалізація алгоритму автоматичного формування харчового раціону людини

5. Перелік графічного матеріалу
5.1 Структурна схема блоку альтернатив (плакат, арк. А4).
5.2 Блочна мережа альтернатив вибору страв (плакат, арк. А4).
5.3 Асети до програмного продукту (плакат, арк. А4).

5.4 Блочна альтернативна мережа обіду (плакат, арк. А4)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Усі розділи	Висоцька О. В., зав.каф.	05.09.22	

Нормоконтроль _____ В.М. Олійник «12» грудня 2022 р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «05» вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання.	18.10.22	
2	Аналітичний огляд методів та засобів формування харчового раціону людини.	19.10.22-1.11.22	
3	Розробка алгоритму формування харчового раціону людини.	2.11.22 – 13.11.22	
4	Формування харчового раціону людини при виразковій хворобі шлунка.	14.11.22 – 25.11.22	
5	Реалізація алгоритму автоматичного формування харчового раціону людини	26.11.22 – 10.12.22	
7	Передзахист та усунення недоліків.	12.12.22	
8	Захист роботи.	14.12.2022	

Здобувач вищої освіти _____ В. Г. Білодід
(підпис) (ініціали та прізвище)

Керівник кваліфікаційної роботи _____ О.В. Висоцька
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 96 с., 38 рис., 12 табл., 7 дод., 30 джерел.

АВТОМАТИЗАЦІЯ, БЛОЧНІ АЛЬТЕРНАТИВНІ МЕРЕЖІ, ПРОГРАМУВАННЯ, ТЕХНІЧНА РЕАЛІЗАЦІЯ, РАЦІОН, ХАРЧУВАННЯ, РАЦІОН ЛЮДИНИ.

Об’єкт дослідження – процес автоматичного формування харчового раціону людини.

Предмет дослідження – методи та засоби автоматичного формування харчового раціону людини.

Мета роботи – розробка алгоритму автоматичного формування харчового раціону людини.

Методи розробки – методи блочно альтернативних мереж.

Був проведений аналітичний огляд методів та засобів формування харчового раціону людини. Проведене порівняння власного алгоритму з аналогами, були виявлені та враховані недоліки інших алгоритмів та систем.

Для реалізації алгоритму було використано програмне забезпечення Visual Studio, Unity 3D, Adobe Photoshop, Adobe After Effects.

ABSTRACT

Explanatory note to the qualification work of the master: 96 p., 38 fig., 12 tables, 7 appendix, 30 sources.

RATION, AUTOMATION, NUTRITION, HUMAN DIET, BLOCK ALTERNATIVE NETWORKS, PROGRAMMING, TECHNICAL IMPLEMENTATION

Object of research - the process of automatic formation of human diet.

Subject of research - methods and means of automatic formation of human diet.

Purpose - to develop an algorithm for the automatic formation of the human diet.

The algorithm of automatic formation of human diet is developed, the algorithm is implemented in the software environment, developed a digital product based on the implemented algorithm in the software environment

Comparison of own development with analogues, identified and taken into account the shortcomings of other developments.

For the development and implementation of the algorithm was used software Visual Studio, Unity 3D, Adobe Photoshop, Adobe After Effects.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БАМ – блочна альтернативна мережа

ШІ – штучний інтелект

РМ – робочі місця

ТЦ - технологічний цикл

ТФ- таблиця функціонування

ВСТУП

Актуальність роботи. Життя кожної людини, групи людей і нації залежить від географічних, економічних, політичних, культурних і релігійних умов. Спосіб життя формується у результаті щоденного повторення і складається із таких факторів: харчування, фізичні навантаження, наявність шкідливих звичок, моральний і духовний розвиток тощо. В останні десятиліття спосіб життя вважають невід'ємною частиною добробуту, що сприяло збільшенню кількості досліджень. Медики стверджують, що більш ніж половина проблем зі здоров'ям пов'язані із дієтою. Мільйони людей харчуються неправильно, навіть не підозрюючи про це. Актуальність теми зумовлена численністю підходів до вирішення проблеми контролю дієти, проте різні аналоги пропонують можливості, які відрізняються і не завжди зрозумілі та зручні, оскільки існує кілька способів досягнення однієї мети. Дослідження стосовно здорового харчування в сучасних умовах є одним із пріоритетних завдань задля покращення фізичного стану різних вікових груп. На данний момент у світі є дуже багато варіантів щодо їжі, і багато з неї – дуже шкідлива. Більшість людей не розуміє плюси чи мінуси свого поточного раціону харчування. Так як питання їжі завжди буде актуально, було вирішено розробити алгоритм, який буде автоматично формувати харчовий раціон людини спрямований на те, щоб допомогти кінцевому споживачеві дотримуватися здорового харчування, визначаючи склад та калорійність продукту, сформувати рекомендації відповідно до ритму життя.

Мета і завдання дослідження. Метою даної роботи є розробка Розробка алгоритму автоматичного формування харчового раціону людини

Для досягнення поставленої мети в магістерській роботі необхідно вирішити наступні завдання:

− провести аналіз існуючих математичних методів і засобів формування харчового раціону людини;

- розробити мережну модель формування збалансованошо раціону людини на базі блочно-альтернативних мереж

− розробити алгоритм автоматичного формування харчового раціону людини;

− провести реалізацію алгоритма автоматичного формування харчового раціону людини.

Об'єктом дослідження є процес автоматичного формування харчового раціону людини.

Предметом дослідження є методи та засоби автоматичного формування харчового раціону людини.

Методи дослідження. Методами дослідження є метод блочно-альтернативних мереж, методичні основи побудови інформаційних та біотехнічних систем.

Наукова новизна отриманих результатів. Розроблено мережну модель, що базується на методі блочно-альтернативних мереж, враховує раніше не досліджені в комплексі особливості формування збалансованого раціону харчування, що дозволяє знизити ризики розвитку та прогресування паталогій людини.

Практичне значення отриманих результатів. На базі розробленої мережної моделі, розроблено алгоритм автоматичного формування харчового раціону людини і створено програмний продукт, який дозволяє в ігровій формі підібрати раціональне харчування.

1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ ФОРМУВАННЯ ХАРЧОВОГО РАЦІОНУ ЛЮДИНИ.

1.1. Основні принципи раціону харчування людини

Раціон сучасної людини відіграє важливу роль у формуванні її здорового способу життя, постачаючи в організм головні й незамінні поживні речовини, мінеральні компоненти та енергію.

На сучасному етапі розвитку суспільства кожна людина розуміє, що її здоров'я та довголіття безпосередньо залежить від їжі, яку вона щоденно споживає. Різке погіршення екологічного стану в усьому світі в другій половині 20ст., що пов'язано з технічним прогресом, помітно вплинуло на якісний склад їжі, що, в свою чергу, привело до появи нових і різкого зростання кількості «старих», відомих захворювань, обумовлених неправильним харчуванням населення. З'явився термін «хвороби цивілізації». До таких хвороб належать: перевтома, високий кров'яний тиск, атеросклероз, запори, геморой, ожиріння, діабет, жовчнокам'яна хвороба та ін.

Причиною цих захворювань є наявність у харчових продуктах інгредієнтів, які належать до факторів ризику. Для серцево-судинних захворювань таким харчовим інгредієнтом е холестерин для онкологічних - нітроза міни та полі циклічні вуглеводні, що містяться в копченостях, для діабету - глюкоза, для інсульту - кухонна сіль і насичені жирні кислоти і т. ін. Харчування є одним із основних важелів, який, гармонізуючи організм людини з оточенням, сприяє певним чином здоров'ю та здатності організму протидіяти впливу несприятливих факторів навколишнього середовища.

Розуміння значення харчування в життєдіяльності людини, біохімічних процесів перетворень окремих компонентів їжі у структурі тіла, їх вплив на діяльність фізіологічних систем організму є надзвичайно важливим і одним із факторів впливу на здоров'я та працездатність людини. Порушення основних принципів раціонального харчування спричиняють цілу низку аліментарно

обумовлених станів та хвороб - від неспецифічних проявів зниження імунного статусу організму до аліментарних захворювань.

Раціональна корекція харчування населення е актуальною проблемою, оскільки є гарантією забезпечення та зміцнення здоров'я на оптимальному рівні профілактики аліментарних захворювань, зниженню інфекційних захворювань серед населення.

У нашій країні цим проблемам приділяється значна увага, розробляються наукові основи харчування у зв'язку з екологічно несприятливими умовами проживання. Затверджено «Норми фізіологічних потреб населення України в основних харчових речовинах та енергії» для різних груп населення з урахуванням їх віку, статі, професійної діяльності.

Гальмом для максимального використання харчових продуктів у різних раціонах харчування є недостатнє вивчення їх складу і функціональних властивостей, а також недостатня освіченість населення при вільному виборі окремих інгредієнтів їжі та впливу їх комплексів на механізми метаболізму і фізіологічні процеси в організмі здорової та хворої людини. У зв'язку з цим потребують удосконалення традиційні методи обробки харчових продуктів з метою розробки способів, які б мали найбільший оберігаючий вплив на склад сировини, а також удосконалення рецептур страв. Це буде сприяти кращому перетравленню їжі та засвоєнню нутрієнтів.

Було доведено[1], що біологічна роль окремих компонентів харчових продуктів не обмежується їх значенням для організму людини як пластичних і енергетичних ресурсів. Їжа є джерелом інгредієнтів, які виконують регуляторну й захисну функцію, необхідну для узгодженої діяльності всіх органів і систем організму, пристосування його до різних умов зовнішнього середовища, підвищення стійкості організму до дії хвороботворних факторів.

Велике значення для травлення та засвоювання їжі мають її зовнішній вигляд, смак, аромат, час вживання та ін.

Академік Павлов розглядав організм людини як вищий ресурс природи, який завдяки властивостям центральної нервової системи здатний до самооновлення та самовдосконалення[1]. Він підкреслював необхідність системного підходу до цілісного вивчення організму людини, сталості його взаємозв'язків із середовищем перебування.

Їжа та її склад є найважливішим чинником життєдіяльності, росту, розвитку, регулювання функцій та відновлення організму.

Практично всі необхідні для життєдіяльності речовини надходять в організм людини з їжею. Було зазначено[2], що близько шістдесяти поживних речовин вважаються необхідними для нормального функціонування людського організму. Двадцять з них можуть синтезуватися в організмі, а інші повинні надходити з їжею.

Їжа в організмі виконує декілька життєво важливих функцій, основними з яких є:

-постачання енергетичних субстратів, що забезпечують організм енергією і підтримують температуру тіла, роботу м'язів і залоз, здійснюють біосинтез власних білків, нуклеїнових кислот, ферментів, гормонів та інших метаболітів;

-надходження «будівельних» матеріалів, які необхідні для росту і відтворення тіла, для відновлення клітин, а також для біосинтезу нових і заміни старих клітинних структур.

Людина споживає різноманітні продукти, і всі їх поживні речовини мають своє функціональне призначення. Найбільше значення для організму людини мають органічні речовини (азотисті сполуки, білки, жири, вуглеводи), мінеральні елементи та вітаміни. Останнім часом до важливих елементів харчування та речовин, що відіграють значну роль у фізіологічних процесах організму та забезпечують його нормальну життєдіяльність, відносять органічні кислоти, фенольні поєднання, харчові волокна, та велике значення для життєдіяльності організму має вода.

Під раціональним харчуванням мають на увазі науково обґрунтоване харчування здорових груп населення, яке забезпечує сталість внутрішнього середовища організму (гомеостаз) і підтримує його життєві прояви (зростання, розвиток, діяльність різних органів і систем) на високому рівні за різноманітних умов праці і побуту. При цьому науковою основою організації раціонального харчування людини, незалежно від її віку, статі, стану здоров'я і професійної приналежності, е загальні фізіолого-гігієнічні вимоги до харчового раціону, режиму і умов харчування.

У літературі, крім терміну «раціональне харчування» (від латинського слова rationalis — розумний), можна зустріти й інші терміни, такі як «правильне харчування», «науково обґрунтоване харчування» «оптимальне харчування», «збалансоване харчування», «адекватне харчування» та ін. Всі ці терміни рівнозначні.

Раціональне харчування різних груп населення передбачає врахування віку і статі людини, характеру праці, кліматичних умов, функціонального стану організму та інше.

Харчовий раціон повинен складатися, виходячи з потреб організму конкретної особи. При складанні біологічно повноцінного раціону враховують санітарно- епідемічну бездоганність раціону, його енергетичну цінність і якісний склад, збалансованість живильних речовин, поєднання харчових продуктів, засвоюваність, органолептичні властивості (зовнішній вигляд, колір, смак, температура та ін.) і різноманітність, насиченість структури.

Можна виділити три основні фізіологічні постулати, яких необхідно дотримуватися при складанні раціону:

-калорійність їжі, що вживається, повинна відповідати енерговитратам організму;

-у добовому раціоні необхідно враховувати потреби організму в належній кількості білків, жирів і вуглеводів;

-необхідно враховувати відповідну потребу у вітамінах, солях і мікроелементах. Враховуючи можливість токсичного впливу на організм надмірно великих доз вітамінів, солей і мікроелементів, їх кількість не повинна бути вищою за оптимальний рівень.

При цьому до розробки раціонального харчування висуваються певні фізіолого - гігієнічні вимоги (табл.1.1).

Таблиця 1.1 - Фізіолого гігієнічні вимоги для розробки раціонального харчування

Вимоги до раціонального харчування	
Харчовий раціон	-безпечність -енергетична цінність -нутрієнтний склад -збалансованість -засвоюваність, легкотравність -органолептичні властивості страв
Режим харчування	-час харчування -тривалість харчування -кратність харчування -інтервали між харчуванням -черговість прийому страв
Гігієна харчування (умови харчування)	-інтер'єр зали для обіду -сервірування столу

Науково обґрунтовані норми харчування людини базуються на результатах фундаментальних досліджень, що розкрили роль у харчуванні і механізмі асиміляції білків, ліпідів, вуглеводів, вітамінів, мінеральних речовин. Норми харчування людини мають враховувати стать, вік, енерговитрати організму. Ці норми періодично переглядаються з врахуванням новітніх досягнень науки і відповідно до змін, що відбуваються в умовах праці

і побуту населення, зокрема з врахуванням всезростаючої механізації праці в промисловості, в сільському господарстві, в побуті, розвитку міського транспорту, тобто з урахуванням змін, що призводить з одного боку до зниження енерговитрат організму людини, а з іншого - до створення можливих негативних впливів специфічних умов праці.

Важливою вимогою до харчування є санітарно-епідемічна бездоганність споживаної їжі. Їжа повинна бути здоровою в прямому розумінні слова, тобто не містити патогенних мікробів та їх токсинів, токсичних видів мікроскопічних грибів, личинок гельмінтів, отруйних речовин та ін. Однією з головних гарантій цього е використання для харчування доброякісних свіжих продуктів, суворе дотримання правил зберігання, обробки і термінів реалізації готової кулінарії продукції.

При організації раціонального харчування окремих контингентів населення слід керуватися загальними фізіолого-гігієнічними вимогами, що висуваються перед організацією раціонального харчування. Важливе фізіолого-гігієнічне значення мають віково-статеві особливості протікання обмінних процесів в організмі людини, а також умови трудової діяльності контингенту населення при нормуванні потреби в енергії та основних харчових речовинах різних груп населення. Згідно з нормами залежно від рівня середньодобового коефіцієнта фізичної активності (КФА) працездатне населення розподілене на 4 групи (табл.1.2).

Харчування людини повинне задовольняти всі потреби організму, забезпечувати діяльність всіх його органів і систем, тобто виконувати різні функції для нормальної діяльності організму, включаючи можливий захист від дій несприятливих чинників навколишнього середовища. Це можливо тільки в тому випадку, коли харчові речовини з їжею надходитимуть в організм людини регулярно в певній кількості і в певному співвідношенні.

Таблиця 1.2- Групи працездатного населення залежно від фізичної активності

Групи	КФА	Орієнтовний перелік спеціальностей
1. Працівники переважно розумової праці, дуже легка фізична активність	1,4	Науковці, студенти гуманітарного фаху, оператори ЕОМ, контролери, педагоги, диспетчери, працівники пультів управління тощо
2. Працівники, зайняті легкою працею, легка фізична активність	1,6	Водії трамваїв, тролейбусів, робітники конвеєрів, швейники, пакувальники, робітники радіоелектронної промисловості, агрономи, медсестри,
3. Робітники праці середньої важкості, середня фізична активність	1,9	Сподарі, наладчики, верстатники, водії екскаваторів, бульдозерів, автобусів, лікарі-хірурги, текстильники, взуттевики, залізничники, продавці продтоварів, апаратники, металурги-доменщики, робітники хімічних заводів
4. Робітники важкої і особливо важкої фізичної праці, висока і дуже висока фізична активність	2,3 (чоловіки) 2.2 (жінки)	Будівельники, помічники буровиків, прохідники основна маса робітників сільського господарства, механізатори, доярки, овочівники, деревообробники, металурги, ливарники, доменщики, вальники лісу, каменярі, землекопи тощо

1.2. Аналіз методів та засобів формування харчового раціону людини

Системи автоматичного формування раціону є предметом досліджень з 60-х років. Першими були представлені рішення, засновані на лінійному програмуванні. У 1967 році була запропонована система[1], заснована на випадковій генерації страв, що відповідають деяким шаблонам. Критерії

оцінки враховують кількість калорій, вартість страви, її колір і різноманітність. Наступні системи застосовували інтелектуальні методи. Було описано[3] приклад типової експертної системи для планування меню. Вона поєднує в собі систему міркувань, засновану на конкретних випадках, з системою, заснованою на правилах, створеною на основі систем PRISM і CAMPER. Основна ідея полягала в тому, щоб на основі аналізу 84 меню згенерувати різноманітні меню, адаптовані до індивідуальних потреб пацієнтів.

Відомий алгоритм [4], який дозволяє оптимізувати параметри щодо визначення оптимального раціону харчування людини.

За неподільний елемент системи було прийнято робоче місце, позначивши його через r з індексами, а сукупність робочих місць (РМ) - через R. Кожне r представляється у вигляді робітників, працівника плюс верстати і машини. Кожне р має входи x і виходи y, внутрішній стан Z. На входи подаються сигнали (інформація) або матеріали у вигляді продукції, речовин (рідких або газоподібних) і т.д. На виходи подаються сигнали (інформація) або матеріали у вигляді продукції, речовин (рідких або газоподібних) і т.д. Частина вхідних сигнальних впливів може бути керуючими (g). В якості машин використовуються верстати і хімічні апарати, комп'ютери.

Машини виступають як знаряддя праці, а інформація, матеріали - як предмети праці. За кожним r закріплюється певна кількість операцій Р.

Позначимо множину операцій через Р. Крім того, вони функціонують у часі і мають просторові координати.

Множина r з'єднана між собою дугами і утворює комунікаційну мережу з потоками. Маються на увазі потоки інформації, речовин, а також транспортні, людські потоки тощо. Отже, система представляється у вигляді комунікаційної мережі, вершини якої зображують робочі місця, здатні виконувати певну кількість операцій (розв'язувати задачі, обробляти матеріали тощо), а дуги відповідають потокам між цими місцями. Таку мережу

було названо R, мережу, яка допускає рух у часі різних потоків у мережі з можливими спотвореннями і не вводить додаткових вершин замість дуг. У процесі функціонування системи структура мережі може змінюватися з часом - старі дуги та вершини анулюються, а нові додаються. Такі мережі було названо ситуаційними або - РК - мережею. При вирішенні певного класу задач за час tt над кожною ЗП виконується одна з закріплених за нею операцій. Тому побудова самої мережі та визначення приписаної їй операції є основним завданням системних досліджень. У певний проміжок часу мережу можна зобразити у вигляді орієнтованого графа постійної структури (рис. 1.1):

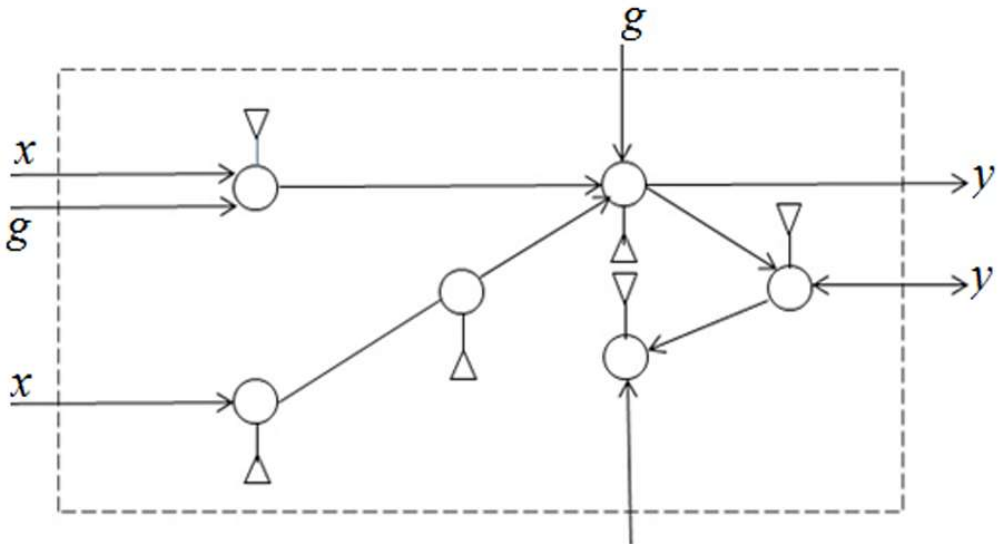


Рисунок 1.1 - Орієнтований граф константної структури

На рис. 1.1 круги позначають операції, трикутники позначають робочі місця, де ці операції виконуються за певний проміжок часу. Інший погляд на цей метод наведено на рис. 1.2.

Таке представлення було названо таблицею функціонування (ТФ) і , R RC - будемо представляти мережу у вигляді ТФ. На цій мережі можна фіксувати параметри потоку та режим роботи мережі в часі. Інтервали часу (t_0 ,t_1),(t_1,t_2),...,(t_n–1,t_n), протягом яких структура ТФ залишається незмінною, будемо називати технологічними циклами.

Крім того, за допомогою ТФ можна пов'язати функції різних систем, що має вирішальне значення для таких великих КС, як автоматизовані системи. Що стосується математичного моделювання, то можливий крайній (але іноді необхідний) випадок, коли виписуються моделі різних операцій.

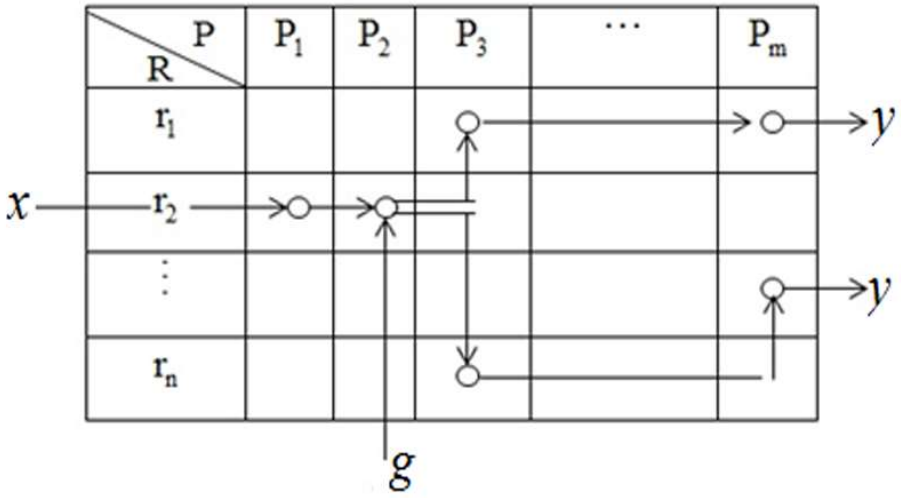


Рисунок 1.2 - Таблиця функціонування

Такий шлях не приводить до узагальнень, не виключає дублювання, а тому є неприйнятним. Тому при побудові алгоритмічних систем використовується метод генерації моделей із загальних закономірностей функціонування систем, а конкретні моделі функціонування систем, тобто конкретні операції виписуються в граничних випадках.

Динамічні таблиці функціонування були визначені наступним чином:

$TA = \{P, D, I, O, A, T, \Delta, F\}$, де P, D, I,O, A,T, Δ відповідно множина позицій, операцій, вхідних та вихідних станів, інтервалів часу та координат системи;

$F(t)$ функція зміни таблиці функціонування в часі. Якщо для $\forall t_i \in T$ існує функція $F(t_i) = const$, то така таблиця функціонування називається статичною (стаціонарною). Функція $F(t)$, яка визначає зміни в робочому стані, називається функцією управління агрегатної системи або функцією планування процесів в системі.

Введемо алгебру над ФТ і для формальних операцій над ФТ визначимо відповідні правила в матричній формі. На кожному часовому інтервалі t_i представляється у вигляді маркованої мережі: $M = \{P, D, I, O, \mu\}$, де μ функція

$N:\mu:P\to N$. Кожне маркування μ може бути подати у вигляді вектора $\mu=(\mu 1,$ $\mu 2,...,\mu_n)$, $n=P\,u$, $\forall\mu_i\in N, i=1,n.$

Інтервали часу, протягом яких мережа Петрі не змінюється, називають технологічними циклами (ТЦ). ТЦ є різновидом мереж Петрі, тому за аналогією з мережами Петрі визначаються основні мовні засоби опису процесів.

Він говорить, що мова - L називається мовою ТЦ L - типу, якщо є

$\delta:T\to\sum$, «перехідна кімната», і початкове маркування μ та скінченна множина кінцевих маркувань F такі, що $L=\{\delta(B)\in\sum^*,\quad B\in T\}$ і $\delta(\mu,B)\in F$, де $\delta\ (\mu,\ B)$ функція переходу, тобто функція маркування є результатом послідовного запуску $(t_{i1},t_{i2},...,t_{ik}\ )$.

Відомий алгоритм [5] який дозволяє оптимізувати параметри щодо визначення оптимального раціону харчування людини.

Функціональні вимоги до системи розробляються на основі стандартів харчування. Вони визначають, які аспекти повинні бути включені в автоматизований генератор страв. Функціонал системи повинен враховувати:

- пристосованість до кількості прийомів їжі та продуктів в одному прийомі до довжини хромосоми,
- кількісні потреби поживних речовин у страві та добовому раціоні,
- вимогу до присутності груп продуктів в раціоні,
- можливість виключення продуктів або груп продуктів згідно з меню.

Вони являють собою сукупність обмежень, які впливають на сформовані страви і є вхідними даними для системи. Для визначення структури вхідного файлу з обмеженнями було розроблено XML-схему. На рисунку 1.3 зображено дерево вимог, яке описано в XML-схемі.

Обмеження визначаються для кожної особи і вводяться в систему у вигляді XML-файлу. У файлі також визначаються такі норми, як GDA, RDA та харчова піраміда. Вхідний файл XML включає в себе елементи слотів, які

представляють вимоги до наявності груп продуктів в стравах. Наприклад, рядок з XML-файлу:

<slot type="majorGroup">5</slot>

вказує на використання в страві продукту з групи "Крупи". Ідентифікаційний номер для групи "Крупи" - 5. Визначення:

<attribute id="24">800</attribute>

описує потребу у вітаміні А у кількості 800 мкг. Ідентифікаційний номер вітаміну А - 24. Потреби в харчуванні задовольняються на рівні одноразового прийому їжі та добового раціону.

Цей алгоритм генерації раціону працює наступним чином:

1. Створити популяцію *P*
2. Повторювати до тих пір, поки не буде виконана умова зупинки:

2.1. Кросовер *P*.

2.2. Мутація *P*.

2.3. Провести селекцію на *P*

Популяція P складається з індивідуумів, які складають щоденний план харчування. Добові потреби задаються лікарем-дієтологом і враховують кількість прийомів їжі та вказують кількість продуктів. Продукти беруться з довільної бази даних продуктів. Припускаючи, що раціон включає j = 5 прийомів їжі на день, і кожен прийом їжі складається з kj = 4 продуктів, хромосома являє собою вектор з 20 значень.

Для проведення селекції необхідно оцінити кожну особу в популяції. Оцінка та модель відбору будуть впливати на алгоритм та його можливість досліджувати простір розв'язків. У запропонованій системі функція пристосованості має наступний вигляд:

$$F(X) = \frac{\sum_{i=0}^{dr-1} f(v_i, w_i) + \sum_{a=0}^{j-1} \sum_{b=0}^{mr_a-1} f(v_b, w_b)}{dr + \sum_{a=0}^{j-1} mr_a},$$

де *dr* - кількісне задоволення добової потреби, mr_a - виконання кількісних вимог до раціону, а

- індекс продукту, v_i - величина потреби в поживній речовині, і - індекс поживної речовини, w_i - необхідна кількість поживної речовини в раціоні, v_b та w_b - кількісні вимоги до раціонів харчування. Функція F визначає, чи дотримані кількісні вимоги, і розраховується за формулою:

$$f(v,w) = e^{\frac{(v-w)^2}{0.5w}},$$

де v - кількість поживної речовини в раціоні, а w - необхідна кількість. У знаменнику експоненти функції f використано коефіцієнт для значення w. Коефіцієнт вибрано довільно на рівні 0,5.

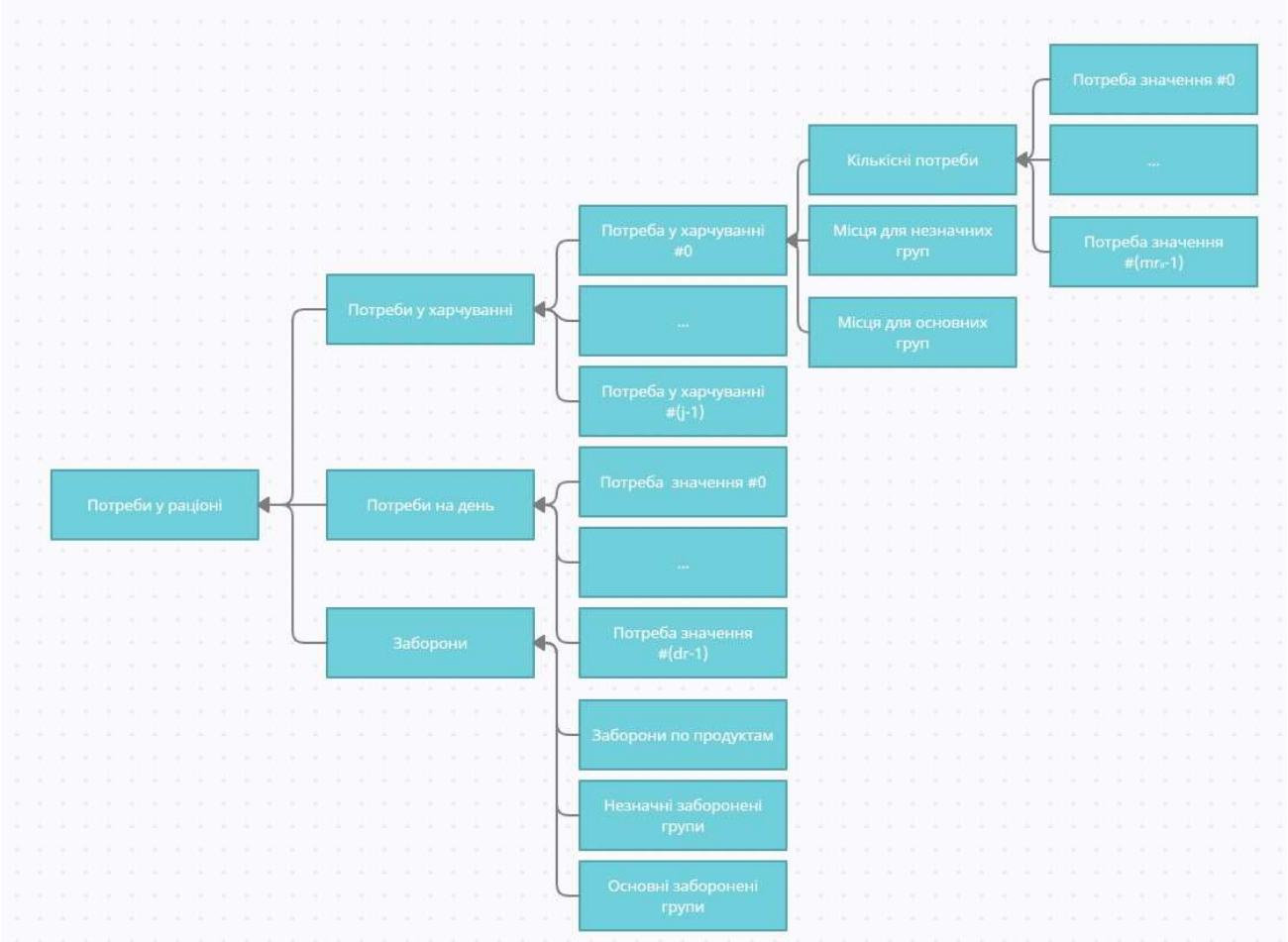


Рисунок 1.3 - Дерево вимог до раціону харчування

Відомий алгоритм [6] який дозволяє оптимізувати параметри щодо визначення оптимального раціону харчування людини.

Економіко-математична модель задачі була представлена наступним чином:

$$C = \sum_{j=1} c_i x_i \to min,$$

за умов, що раціон має містити поживні речовини, не менші за мінімально-допустиму кількість:

$$\sum a_{ij} x_i \geq b_i.$$

Розв'язання задачі було здійснено за допомогою програми Optim v.2.0 (рис 1.4) розробленої доцентом кафедри економічної кібернетики Косніковим С.М. [5, 6, 7, 11].

	X3 Гречневая	X7 Креатин	X8 Крупа овся	X11 Куриная гр	X13 Молоко кор	Сумма	Вид	Объём
Знач.X	0,25	0,24	0,16	0,66	0,49			
1 Калор, к	770,53	512,56	535,24	743,46	151,22	2713,00	=	2713
2 Белки, г	31,52	120,02	18,46	155,27	14,73	340,00	=	340
3 Белки Ж	0	0	0	155,27	14,73	170,00	=	170
4 Жиры, г	8,26	0	9,00	12,50	0,25	30,00	=	30
5 Углевод	142,85	0	101,46	2,63	24,06	271,00	=	271
C =	12,51	346,71	3,88	95,40	22,09	480,59	--->	MIN

Рисунок 1.4 – Вікно програми Optim v.2.0

Цей підхід до розрахунку раціону харчування людини, з використання економіко-математичних методів, дає змогу визначити оптимальні параметри з урахуванням типів метаболізму і підтримки маси.

Робота знайомить читача з результатами, розрахованими для конкретної людини, при цьому автори припускають, що в модель можна внести низку доповнень і уточнень, які дадуть змогу її адоптувати за потребами кожного.

Сформулюємо недоліки існуючих рішень підбору раціону харчування, виявлені на етапі аналізу відомих систем:

1. Коли людина вирішує почати харчуватися раціонально без відповідних початкових знань, їй буде складно знайти інформацію про норми, продукти та їх нутрієнти, раціони харчування або способи їх складання. Тому існує потреба у розробці комплексного рішення, яке буде супроводжувати користувача від прийняття рішення до споживання їжі.

2. Існує багато способів підрахування норм для раціонального харчування людини, вченими розробляються нові та удосконалюються існуючі. Сучасні системи пропонують багато різних методів розрахунку, які достатньо вагомо відрізняються за своїми результатами або взагалі не надають інформацію про метод розрахунку норм, а лише показують результат розрахунку. Такий підхід не є коректним, адже деякі методи розрахунку можуть бути застарілими, а систему можуть використовувати і більш інформаційно обізнані люди. Для цього потрібно пропонувати користувачам максимально актуальні та точні методи розрахунку, але їх не повинно бути багато.

3. Існуючі системи або ресурси пропонують користувачам готові раціони харчування, не враховуючи потреб кожного індивідуального користувача. Навіть якщо у відомих системах користувачам виводяться їх норми раціонального харчування, то після цього їм потрібно вручну підбирати собі індивідуальний раціон харчування, а це становить певні труднощі для користувача, тим більше майже ніколи не вдається підібрати раціон харчування, який би за своїми складовими максимально точно відповідав би нормам користувача.

4. Припустимо, що користувач підібрав для себе раціон харчування. Після цього необхідно буде здійснити корегування даного раціону, для того щоб останній відповідав денній нормі споживання. Для цього необхідно самостійно додавати або віднімати вагу того чи іншого продукту харчування, який входить до цього раціону або взагалі його прибрати. При цьому кожного разу після корегування ваги, потрібно перераховувати показники компонентів

(нутрієнти) продуктів харчування. Наведені вище дії потребують додаткових знань і витрат часу.

5. У готовому раціоні харчування можуть бути продукти харчування, які людина не може споживати через медичну, релігійну або просто індивідуальну заборону. Недоліком відомих систем є неможливість користувачам обирати вподобані продукти харчування в певному раціоні. Також слід зазначити, що більшість раціонів складають із «загальних» продуктів харчування (наприклад, молоко, сир, хліб тощо), тобто без зазначення виробника. Це означає, що харчова цінність одного і того ж продукту харчування може бути різною.

6. Відомі системи з формування раціонів харчування не мають відомостей про вартість продуктів харчування, які входять до них. Це достатньо вагомий показник при підборі раціону, адже різні продукти можуть дуже сильно відрізнятись за ціною, але харчова цінність буде фактично однаковою. Також може бути ситуація, при якій продукт харчування одного і того ж виробника може коштувати по різному у різних
регіонах.

7. Існуючі системи не несуть у собі адаптивний характер. Вибравши раціон харчування, відкоригувавши його показники під свої норми, користувач не може поділитися таким раціоном із кимось іншим, тому що кожна людина має індивідуальні потреби у раціональному харчуванні.

Після виконання аналізу вищеописаних проблем існуючих сучасних систем з підбору раціонального харчування, досліджень оптимізації раціону було вирішено провести розробку аалгоритму підбору раціонального харчування людини, який був би максимально корисним, простим і зручним у використанні, а також вирішував би зазначені завдання.

2. РОЗРОБКА АЛГОРИТМУ ФОРМУВАННЯ ХАРЧОВОГО РАЦІОНУ ЛЮДИНИ

Нехай об'єкт ∑О або група об'єктів {∑О}. Припустимо, що така сукупність об'єктів представлена множиною атрибутів Аі, де і = 1, 2, ..., n. Виходитимемо з того, що Аі покривають повну сукупність властивостей об'єкта ∑О.

Кожен атрибут Аі може набувати безліч альтернативних значень α:

$$Ai:(\alpha i = (\alpha 1i,\ldots, \alpha 1j,\ldots, \alpha 1M,\ldots,))$$

Альтернатива – необхідність вибору між взаємовиключними можливостями. Значення є альтернативними, тому що передбачається, що в кожний момент часу атрибут може приймати одне і одне значення. Складні завдання завжди формують рішення з урахуванням різних поєднань вихідних даних, з чого утворюються сукупності альтернативних рішень. Для складного об'єкту:

$$A = (A1, ..., Ai, ..., An)$$

Кожен атрибут визначається безліччю його значень, і рішення задаватиметься матрицею атрибутів:

$$\left.\begin{array}{l} A_1 = (a_{11}, \ldots, a_{1j}, \ldots, a_{1m1}) \\ A_n = (a_{n1}, \ldots, a_{nj}, \ldots, a_{nmn}) \\ A_N = (a_{N1}, \ldots, a_{Nj}, \ldots, a_{NmN}) \end{array}\right\}$$

Елементарний блок альтернатив (ЕБА) можна як клас, визначальний безліч об'єктів-альтернатив.

В елементарному блоці має місце три види вершин:

а) вершини першого рангу: вхід та вихід;

б) вершини другого рангу: значення атрибутів;

в) допоміжні вершини: рекурсія та транзит.

У цій структурі має бути реалізована функція вибору альтернативи (ФВА) за умови існування значення альтернативи. Зазвичай подібна функція містить у своєму тілі дві складові: рекурсивний (R) та транзитний (T) блоки.

Транзитний блок використовується в тому випадку, коли жодне значення атрибуту не використовується, і в результаті можна пройти з входу на вихід через транзитну вершину Ti. Рекурсивний блок використовується, коли потрібно вирішити завдання пошуку альтернативного значення масиві альтернатив, т. е. організувати циклічний процес.

В результаті доповнивши дворівневу схему атрибуту замикаючою вершиною (якорем), транзитним і рекурсивним вершинами, отримаємо структуру елементарного блоку альтернатив (рис.2.1.).

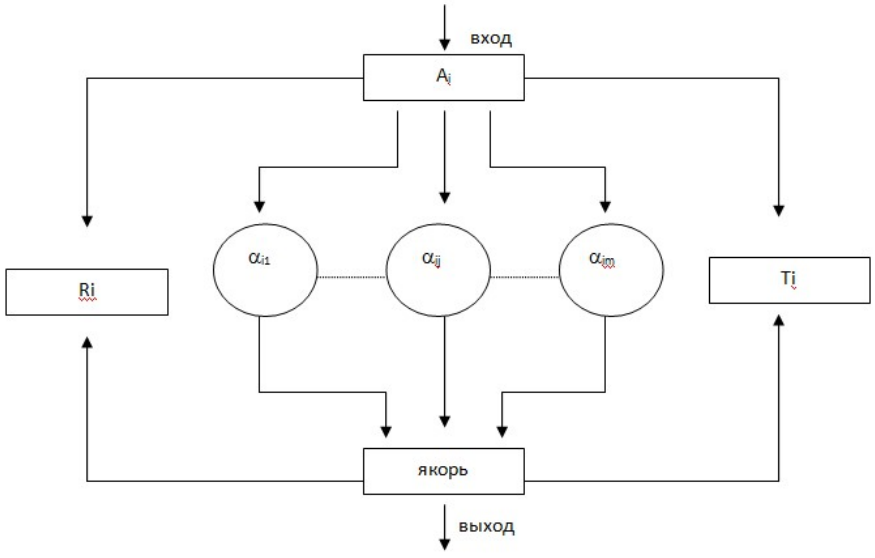


Рисунок 2.1 - Структурна схема блоку альтернатив(ЕБА)

де, Ai – ім'я блоку;

ai1, …, aij, …, aim – значення атрибутів (сукупність альтернатив);

Ri – рекурсивна вершина;

Ti – транзитна вершина;

A*i - замикання альтернатив (якір).

Для спрощення сукупність альтернатив назвемо блоком альтернатив (БА); спрощений вид ЕБА представлений на рис.2.2, де $БА_i$ складається з {ai1, …, aij, …, aim}

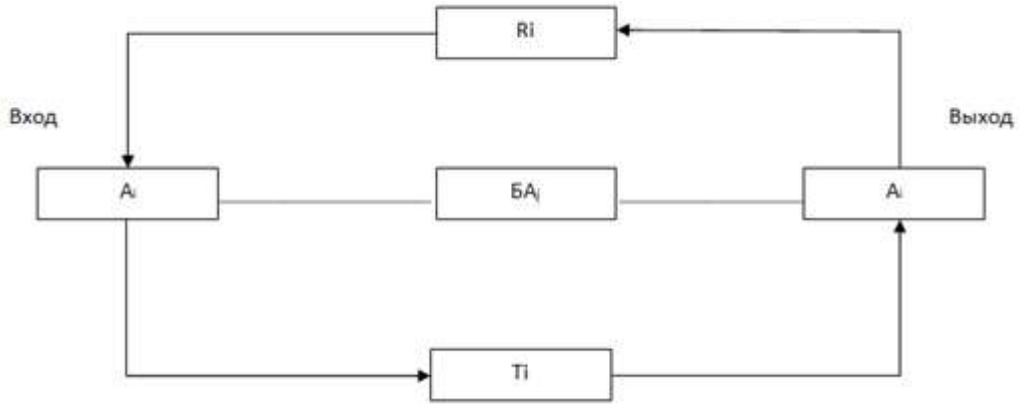


Рисунок 2.2 – Спрощений вигляд ЕБА

Сукупність таких послідовно з'єднаних елементарних блоків утворює просту БАМ. ЕБА – це базовий блок для формування мереж. Його використання дозволяє породжувати будь-які конфігурації мереж чи структур.

У методі буде використано рекурсивний блок, оскільки необхідно вирішити задачу пошуку альтернативного значення на масиві альтернатив. Якір, тобто вихідний блок, представлятиме дієту.

Для роботи з БАМ необхідно створити алгоритми навігації на мережі. Існує три методи навігації на мережі:

• послідовний;

• паралельний;

• змішаний.

Результатом роботи алгоритму навігації є формування вершинного маршруту. Формування маршруту має такий вигляд:

$$M = (V1, ..., Vq, ..., VQ)$$

Основна мета таких алгоритмів полягає у визначенні кожного елемента Vq, в оцінці узгодженості Vq з іншими. Кожен елемент Vq інтерпретується як

окреме локальне рішення. Маршрут інтерпретується як модель результату рішення.

Якщо необхідно згенерувати деяку сукупність рішень, формується кілька маршрутів, що утворюють парадигму рішень. На підставі рішень можливі реалізації завдань аналізу, вибору, упорядкування та оптимізації.

Для послідовної мережі послідовний алгоритм навігації можна реалізувати двома базовими способами.

1. Проходження мережі реалізується послідовно, починаючи з першого блоку a1 і закінчуючи останнім блоками aN. Алгоритм звертається до блоку a1, переглядає його і через транзитні вершини передає результат. Далі переходить до наступного блоку. Через розгалуження утворюється певний вершинний маршрут Mj =(a1j, ..., anj, ..., aNj), тобто результат рішення. Якщо якесь рішення несумісне, то виявляється причина несумісності та шукається нове рішення.

2. Алгоритм послідовно звертається до кожного блоку і результат з кожного блоку передається назад в алгоритм. Результатом роботи є маршрут із приватних рішень, далі процедура продовжується.

При послідовній навігації визначається порядок входу до кожного з блоків, порядок пошуку приватного рішення всередині блоку, порядок виходу з блоку, входу до наступного блоку та «склеювання» приватних рішень.

Нехай заданий кортеж атрибутів (безліч альтернатив):
A = {an: (n = 1, 2, …, n)}. Здійснимо послідовну генерацію наслідків A* = {an*: (n = 1, 2, …, n)} для кожної з альтернатив за допомогою послідовної БАМ.

БАМ із послідовною стратегією представлена рис. 2.3.

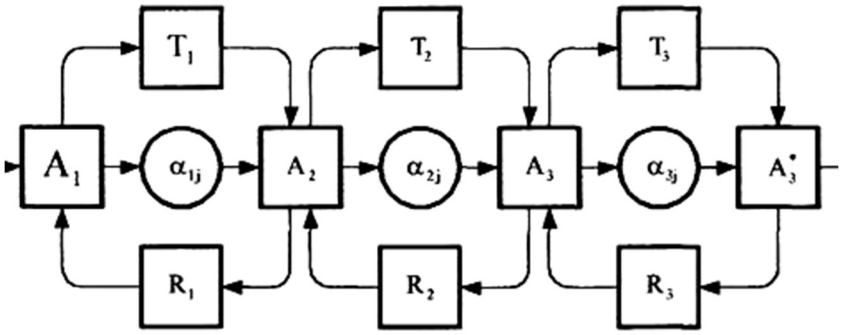


Рисунок 2.3 - Приклад послідовної розімкнутої триблочної БАМ

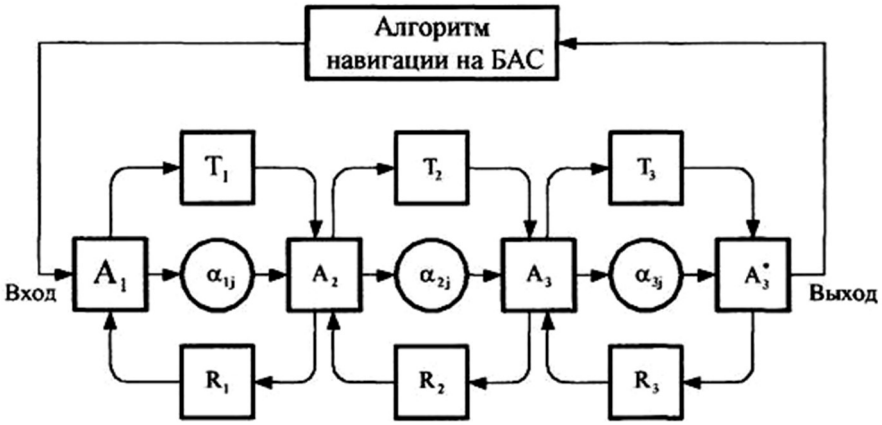


Рисунок 2.4 - Приклад послідовної замкнутої триблочної БАМ

У послідовних БАМ альтернативні рішення, що генеруються, з'єднуються в одну зв'язку з генеруючими наступного ЕБА попарно. В результаті вершини A*n та An+1 зливаються в одну An+1.

При алгоритмі з паралельною організацією навігації можливі дві схеми:

• однорівневий алгоритм;

• дворівневий алгоритм.

За схемою однорівневого алгоритму всі елементи мережі включають координуючу та виконавчу функції.

На рис.2.5 зв'язок реалізується через загальну транзитивну вершину (роздільний вхід та вихід).

При паралельній генерації рішень у кожному блоці БАі працює свій алгоритм формування результатів. Алгоритми працюють одночасно, і матриця альтернативних рішень заповнюється рядково.

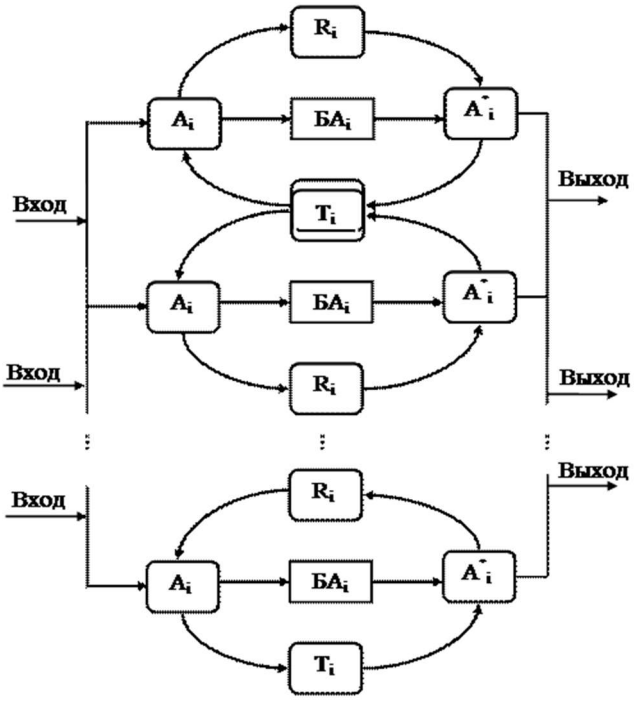


Рисунок 2.5– Паралельна однорівнева структура БАМ

На верхньому рівні дворівневого алгоритму (рис.2.6) перебуває координуючий алгоритм, але в нижньому - виконуючий. За командою координуючого алгоритму кожен виконуючий алгоритм входить у свій блок, визначає приватне рішення всередині блоку та передає результат у координуючий алгоритм. Останній здійснює функцію поєднання приватних рішень на єдине загальне рішення.

Цей підхід до розрахунку раціону харчування людини, з використання економіко-математичних методів дає змогу визначити оптимальні параметри, з урахуванням типів метаболізму і підтримки маси.

Застосування блочно-альтернативних мереж Nα на вирішення різноманітних завдань полягає в використанні їх властивості породжувати безліч альтернативних маршрутів MN.

У БАМ використовують вершинний тип маршрутів. З погляду мережі маршрути поділяються на внутрішньоблокові та мережеві. Останні, у свою чергу, формуються із внутрішньоблокових та міжблокових.

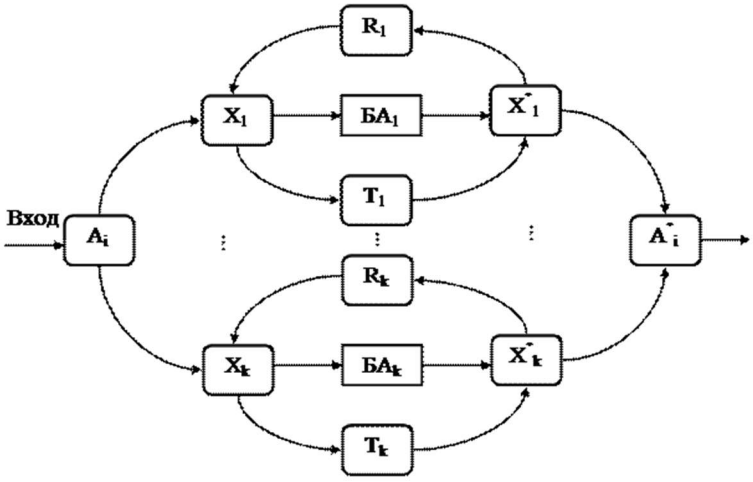


Рисунок 2.6 - Паралельна дворівнева структура БАМ

Застосування блочно-альтернативних мереж Nα на вирішення різноманітних завдань полягає в використанні їх властивості породжувати безліч альтернативних маршрутів MN.

У БАМ використовують вершинний тип маршрутів. З погляду мережі маршрути поділяються на внутрішньоблокові та мережеві. Останні, у свою чергу, формуються із внутрішньоблокових та міжблокових.

Внутрішньоблоковий – це маршрут MiNMN (i = 1,…, n), який пов'язує дві сусідні вершини (πi, πi+1) першого рангу, що належать і-му блоку.⊂

Міжблокові – маршрути MilN, які пов'язують деякі пари вершин першого рангу {(πi+k, πi), i = 1, …, n; k = 1, 2, …, n; l=(i+k)≤n}. Міжблокові маршрути використовуються при формуванні циклів, і, отже, вершини (πi+k, πi), що зв'язуються, для таких маршрутів ототожнюються.

При використанні БАМ для вирішення конкретних завдань можуть виникати ситуації, коли той чи інший внутрішньоблоковий маршрут MiN формується неоднозначним чином. При цьому можливі три випадки:

1) внутрішньоблоковий маршрут MiN проходить один раз зліва направо;

2) внутрішньоблоковий маршрут MiN для маршруту MN є забороненим;

3) внутрішньоблоковий маршрут MiN повинен бути пройдений неодноразово.

Відповідно до названих випадків, визначимо три типи маршрутів: ациклічні AMiN, транзитні TMiN і циклічні CMiN.

Найбільш простими маршрутами MNN є ациклічні (або незамкнуті) AMiN.⊂

Ациклічний маршрут (AMi) формується як послідовність вершин спільно з відношенням між вершинами:

AMi: (Ai, rij, aij),

де Ai – атрибут;

rij - визначає відношення між атрибутом та вершиною-значенням aij;

aij – значення атрибуту Ai.

Повне представлення внутрішньоблокового маршруту за схемою витік-стік буде об'єднанням:

AMi: (Ai, rij, aij) U (aij, rji, A*i),

або загалом для вершин-альтернатив отримаємо вершинний ациклічний маршрут:

AMi: (Ai, aij, A*i).

Аналогічно для маршруту, що проходить через транзитивну вершину:

AMiT: (Ai, rT, A*i),

Ациклічні маршрути мають місце у тих випадках, коли здійснюється одноразове проходження зліва направо через блок Niα, між блоками (Niα, Njα) або по мережі Nα в цілому.

Внутрішньоблокові ациклічні маршрути завжди проходять через вершини πji (j = 1,2, …, m; i = 1,2, …, n) другого рангу. У загальному випадку маршрут AMiN Niα може бути заданий послідовністю:⊂

AMiN = {(πi, rij, πji, rj(i+1), πi+1); (j = 1,2, …, m;∀i= 1, 2, …, n)}

У цій послідовності rij = (πi, πji) та ri(j+1)=(πji, πi+1) позначають дуги між відповідними парами вершин усередині i-го блоку. Коротко цей вираз можна записати так:

$$AMiN = (3.1)\left\{AM_N^i(\pi_{ji}), \forall(j = 1,2, \dots m_i, i = 1,2, \dots, n)\right\}$$

На мережі N будь-який внутрішньоблоковий маршрут AMiN завжди починається з вхідної вершини πi+1.

Досить часто використовуються транзитні маршрути. При використанні мережі Nα можуть бути випадки, коли проходження через блок Niα (i = 1, 2, …, n) або сукупність блоків заборонено.$\left\{\left(N_{\alpha,\ldots,}^{?k}N_{\alpha}^{i}\right); j = (i + k), k \geq 1\right\}$

Для опису таких випадків введено поняття транзитного маршруту TMN. Перш ніж дати визначення внутрішньоблокового транзитного маршруту TMiN, введемо і визначимо поняття транзитної вершини . Транзитними є такі вершини (i = 1, 2, …, n) другого рангу, які не несуть семантичного навантаження відповідно до ознаки πi, а визначають лише маршрут проходження всередині блоку Niα Nα. Таким чином, внутрішньоблоковий транзитний маршрут є TMiN такий маршрут, який проходить через транзитну вершину. Загалом внутрішньоблоковий транзитний маршрут TMiNNα визначається послідовністю:$\pi_T^i \pi_T^i \subset\subset$

$$\text{TMiN} = \{(\pi i, \text{ri}T, \pi \text{T}i, \text{r}T(i+1), \pi i+1, \}\forall\ (i = \overline{1, (n+1)})$$

У тих випадках, коли здійснюється неодноразове проходження через блок Niα (i = 1, 2, …, n) або {(Niα, Njα), l = i+k, k1} або через мережу Nα в цілому, то мають місце циклічні маршрути .≥

Основою циклічних маршрутів CMiN є ациклічні AMiN. Замикання внутрішньоблокового маршруту AMiN здійснюється через вершини (πi+1, πi), які визначають кінець і початок. У загальному випадку для будь-якого блоку Niα циклічні маршрути CMiN Niα можна подати у вигляді суми відповідних ациклічних маршрутів AMiN, кожен з яких повторений Kji раз. Можна записати:

$$\text{CMiN} = \sum_{j=1}^{m_i}\left[\text{AM}_N^i(\pi_{ji})\right] K_{ji}$$

де Kji0 - кількість j-х циклів в і-му блоці.≥

Внутрішньоблокові циклічні маршрути CMiN використовуються в тих випадках, коли при формуванні маршруту MN Nα виникає необхідність

неодноразового проходження через будь-який блок Niα з метою включення в такий маршрут будь-якої кількості будь-яких вершин⊂

$$\pi ji \ (j = 1,2, \ldots, m; i = 1,2, \ldots, n).$$

Міжблокові та мережеві маршрути формуються на основі склеювання внутрішньоблокових. Для цих цілей використовуються спеціальні алгоритми, які здійснюють як формування самого маршруту, так і склеювання внутрішньоблокових в єдиний мережевий:

$$MNa = U \ (МБі),$$

де MNa – мережевий маршрут;

МБі – внутрішньоблоковий маршрут.

За такого алгоритму навігації шляхом склеювання буде отримано маршрут MNa зі своїм набором рішень:

$$R = (R1,, \ldots, Ri, \ldots, RN)$$

До кожного блоку альтернатив визначається свій алгоритм вибору альтернативи. Алгоритм паралельної навігації реалізує функції координації, які взаємодіють із кожним блоковим алгоритмом. Робота здійснюється паралельно. Алгоритм координації передає вихідні дані в локальні алгоритми та запускає їх у роботу. Кожен із локальних алгоритмів формує внутрішньоблоковий маршрут і одержує відповідний результат (R). Далі формується послідовність (R11, ..., Ri1, ..., RN1) = Rl незв'язаних між собою рішень. Після цього вирішується завдання склеювання приватних рішень на загальне.

Лікар-дієтолог, проаналізувавши наявні в асортименті продукти, складає з них страви, що входять до тієї чи іншої дієти. Для приготування будь-якої страви використовується певний набір продуктів {П}. Набір {П} є набором атрибутів блюда Бі:

$$Бі = \{П1, П2, ..., Пm\}$$

Для надання маршрутів вибору страв можна використовувати метод блочно-альтернативних мереж (БАМ). Вид елементарного блоку такої мережі для вибору альтернативних страв Бі представлений на рис. 2.7

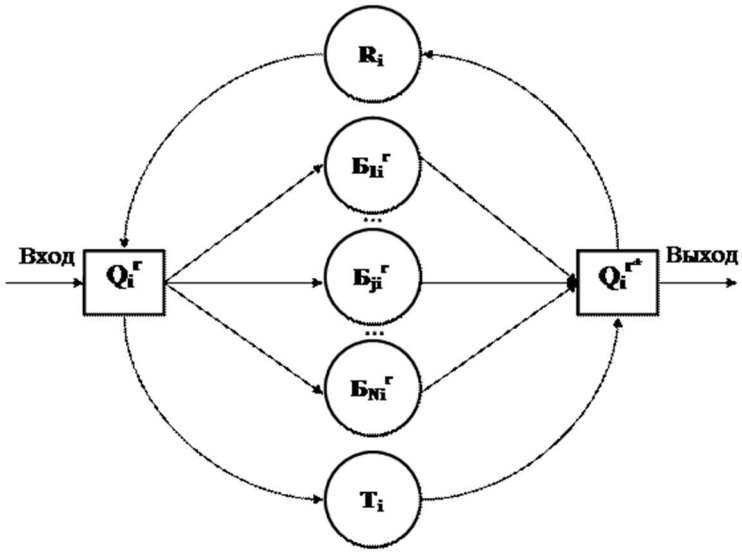


Рисунок 2.7 Елементарний блок альтернатив

де Qir – ім'я блоку;

Qir* – замикання альтернатив;

Змінна r означає прийом їжі (сніданок, обід, полуденок та вечерю) і може приймати значення: r = {З, О, П, У};

Змінна і визначає категорію страв (закуски, перші, другі, треті страви та десерт), i = 1, 2,…,5.

Показник Qir може приймати одне з безлічі значень {Бjir}, сукупність яких представляє альтернативні вершини блоку БАМ. Також ЕБА є Т – транзитна вершина, і R – рекурсивна вершина.

Для всіх прийомів їжі існують такі категорії страв:

Q1 r - закуски (салати та ін);

Q2 r – перша страва (супи);

Q3 r – друга страва (м'ясна або рибна страва з гарніром);

Q4 r – третя страва (напої);

Q5 r – десерт (солодощі чи хлібобулочні вироби).

Отже, один прийом їжі в загальному випадку може містити всі ці категорії страв:

$$Qr = \{Q1r, Q2r, Q3r, Q4r, Q5r\}$$

Для кожної з категорій існує свій набір страв:

Q1r = (Б11r, Б12r, ..., Б1jr, ..., Б1mr);

Q2r = (Б21r, ..., Б2jr, ..., Б2lr);

Q3r = (Б31r, ..., Б3jr, ..., Б3kr);

Q4r = (Б41r, ..., Б4jr, ..., Б4hr);

Q5r = (Б51r, ..., Б5jr, ..., Б5gr).

При цьому одна і та ж страва може належати до різних категорій.

На основі цих даних можна сформувати блочно-альтернативну мережу для сніданку (рис. 2.8). Набір категорій страв сніданку включає:

Q3 = (Q13, Q43, Q53).

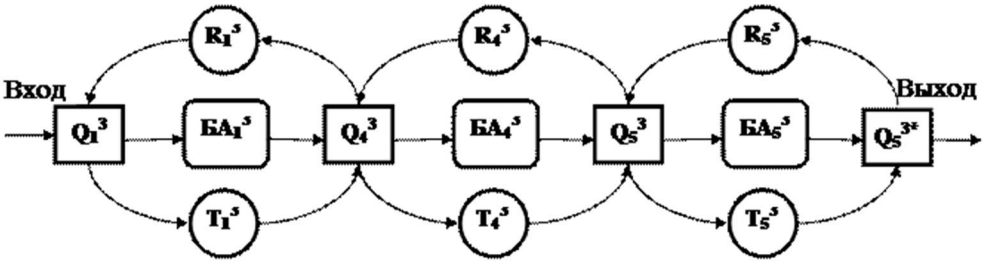


Рисунок 2.8 - БАМ сніданку

Поєднання різних страв {БijЗ} утворює маршрут МеЗ на мережі сніданку. Маршрут вибирається цілеспрямовано, відповідно до певної дієти, тобто з урахуванням страв, дозволених для цієї дієти Дk.

На мережі отримуємо безліч маршрутів МkЗ = {МkеЗ}, де

k - номер дієти,

k = 1, ..., D, де D - кількість дієт;

e = 1, ..., LЗ, де LЗ - кількість маршрутів за даними D дієтам для сніданку;

Mke3 = (Б1j3, Б4j3, Б5j3),

де індекс j – номер страви у списку страв певної категорії.

Загалом маршрут вибору страв на весь день для пацієнта з певною дієтою Дk можна записати у вигляді:

Mke = {Mke3, MkeO, MkeП, MkeУ},

де k – номер дієти,

e – номер маршруту.

Таким чином, для кожної з дієт маємо певну кількість маршрутів вибору страв. Тоді дієту як сукупність маршрутів вибору страв можна записати у вигляді:

Дk = {Mk1,…, Mke,…, MkL}.

Отже, для сніданку певна дієта Дk матиме вигляд:

ДkЗ = {Mk1З,…, MkeЗ,…, MkLЗ}, k = 1, 2, …, D.

Кожен із маршрутів Mke3 характеризується калорійністю, вмістом білків, жирів, вуглеводів, а також вітамінів та мінеральних речовин:

Mke3 = Mke3 (Kke3, БЛke3, Жke3, Уke3, Bke3, MBke3),

k=1,…, D,

e = 1, ..., L.

Аналогічно може бути представлена БАМ для обіду QO (рис.2.9)

QO = {QiO}, i = 1, ..., 5;

QiO = (Бi1O ,Бi2O ,…BijO ,…BiNO).

Маршрути на БАМ для обіду визначаються шляхом вибору однієї альтернативної вершині для кожної категорії страв з урахуванням дієти:

ДkO = (Mk1O, Mk2O, ..., MkS O)

MkO = {Mk1O}, k =1, ..., N;

MksO = (Б1jO ,Б2jO ,Б3jO ,Б4jO)

MksO = MksO (KksO , БЛksO , ЖksO ,УksO ,BksO ,MBksO);

k =1,…, N; s = 1, ..., S.

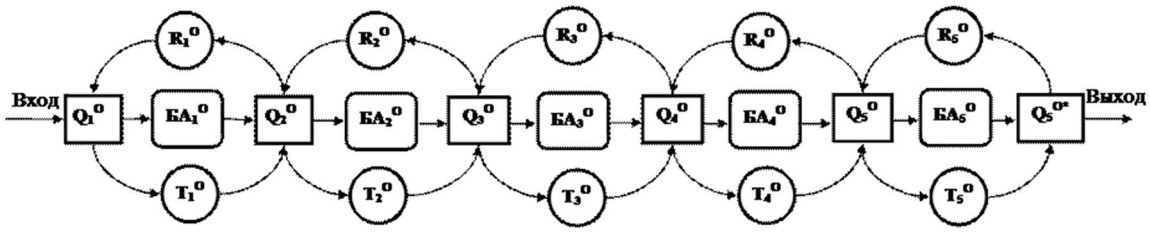


Рисунок 2.9 - Блочно-альтернативна мережа обіду

Аналогічно, можна уявити БАМ для полуденку та вечері.

Послідовно з'єднавши всі БАМ сніданку, обіду, полуденку та вечері, отримаємо повну БАМ одного дня; вона представлена на рис. 1.10.

Рисунок 2.10 – Повна БАМ страв на один день

Дієта є безліччю всіх маршрутів вибору страв:

$$Дk = \{Mke\}$$

де k - Номер дієти, k = 1, ..., D;

e - номер маршруту, e = 1, ..., L.

А кожен маршрут включає безліч маршрутів вибору страв кожного прийому їжі:

$$Mke = \{MkeЗ, MkeO, MkeП, MkeУ\}.$$

Безліч всіх маршрутів на БАМ представлено на рис. 2.11.

З огляду на те, що лікаря-дієтолога необхідно складати раціони харчування пацієнтів щодня, то алгоритм навігації на БАМ маршрутів вибору страв має бути замкнутим. Замкнена БАМ маршрутів вибору страв представлена на рис. 1.11.

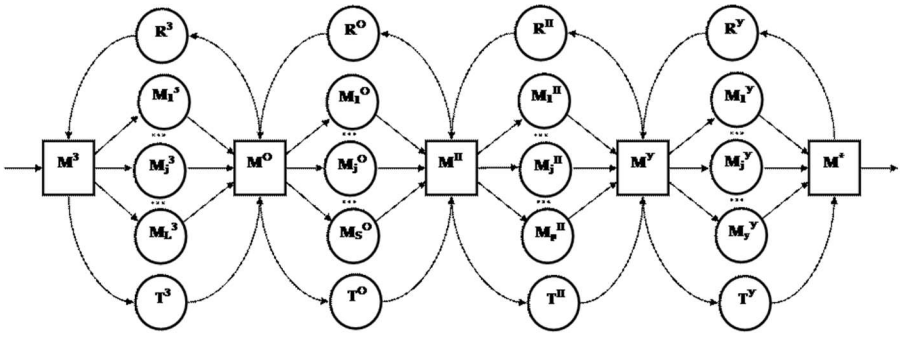


Рисунок 2.11 – БАМ маршрута вибору страв

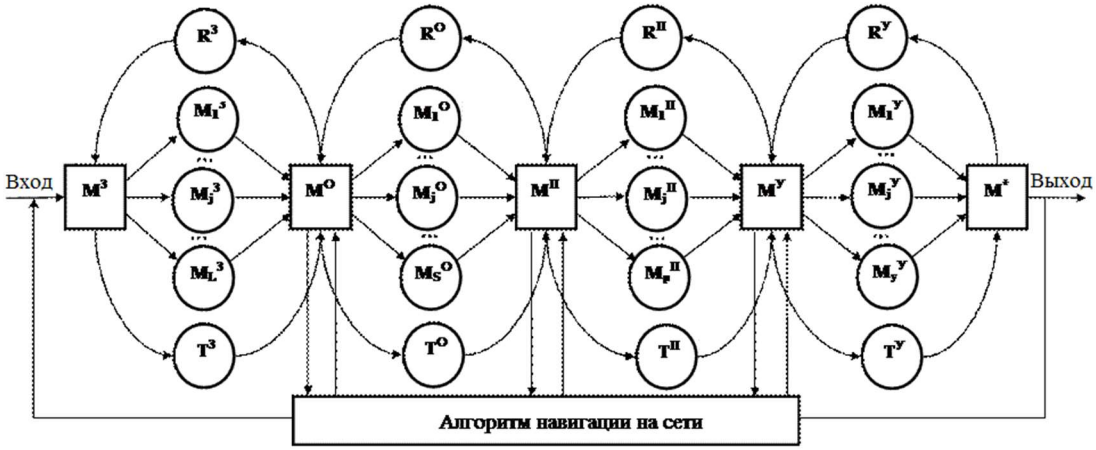


Рисунок 2.12–БАМ маршрутів вибору страв

3. ФОРМУВАННЯ ХАРЧОВОГО РАЦІОНУ ЛЮДИНИ ПРИ ВИРАЗКОВІЙ ХВОРОБІ ШЛУНКА.

При отриманні даних стан здоров'я пацієнта дієтолог призначає дієту (Ді). При формуванні дієти лікар повинен враховувати такі фактори, як

енергетичний та вітамінний склад продуктів, які використовуються для приготування страв. Кожен продукт оцінюється кількістю білків, жирів, вуглеводів, мінеральних речовин (макроелементи та мікроелементи) та енергетичною цінністю. Тому всі продукти будуть складатися з певного набору атрибутів. Складемо таблиці вихідних даних у якому рядки – значення атрибутів, а стовпці – атрибути.

де Б – білки;

Ж – жири;

У – вуглеводи.

Для приготування блюда використовується певний набір продуктів. Набір продуктів {П} є набором атрибутів для приготування Бj. Наведемо загальний вигляд складу страв:

Вважатимемо, що об'єкт – це дієта, атрибут – це страва, а альтернативне значення – це продукти.

Набір продуктів для приготування страв:

- Б1 (каша рисова молочна слизова) = (П11 (рис), П21 (молоко), П31 (вода), П41 (масло вершкове));

- Б2 (омлет) = (П12 (яйце), П22 (молоко), П32 (масло вершкове));

-Б3 (молоко);

- Б4 (суп вівсяний молочний слизовий) = (П14 «Геркулес», П24 (молоко), П34 (яйце), П44 (масло вершкове), П54 (цукор), П64 (вода));

- Б5 (суфле з пареного м'яса) = (П15 (м'ясо телятина), П25 (яйце), П35 (молоко), П45 (масло вершкове), П55 (борошно));

- Б6 (компот із сухофруктів) = (П16 (сухофрукти), П26 (цукор), П36 (вода));

-Б7 (яйце некруто);

-Б8 (каша гречана молочна слизова) = (П18 (крупа гречана), П28 (молоко), П38 (вода), П48 (масло вершкове));

- Б9 (каша манна молочна) = (П19 (крупа манна), П29 (молоко), П39 (цукор), П49 (масло вершкове));

- Б10 (суп рисовий молочний слизовий) = (П110 (рис), П210 (молоко), П310 (яйце), П410 (масло вершкове), П510 (цукор), П610 (вода));

- Б11 (суфле з судака з олією парове) = (П111 (риба), П211 (масло вершкове), П311 (борошно пшеничне), П411 (яйця), П511 (молоко));

- Б12 (каша вівсяна молочна слизова) = (П112 «Геркулес», П212 (молоко), П312 (вода), П412 (масло вершкове));

-Б13 (суфле з вареного м'яса) = (П113 (телятина), П213 (масло вершкове), П313 (яйця), П413 (молоко), П513 (борошно));

- Б14 (кисіль вишневий) = (П114 (вишня), П214 (цукор), П314 (крохмаль), П414 (вода));

- Б15 (суп манний молочний) = (П115 (крупа манна), П215 (молоко), П315 (яйце), П415 (масло вершкове), П515 (цукор), П615 (вода));

- Б16 (суфле сирне з вишневою підливою парове) = (П116 (сир), П216 (крупа манна), П316 (масло вершкове), П416 (цукор), П516 (яйце), П616 (вишня суха), П716 (крохмаль) .

- Б17 (кисіль апельсиновий) = (П117 (апельсин), П217 (цукор), П317 (крохмаль), П417 (вода)).

Так як різні страви складаються з різної кількості продуктів, то довжина набору атрибутів для кожної страви буде різною. Для приготування різних страв можуть використовуватися ті самі продукти в одному випадку і зовсім різні в іншому тому безліч продуктів для приготування кожної страви можуть перетинатися і навпаки.

Значення атрибуту (що характеризують страву) визначатиметься вагою брутто (для однієї порції). Взаємозв'язок між продуктами та стравами при виразковій хворобі показано на рис. 2.13 та рис. 2.14. З цього відношення можна отримати дані про склад продуктів для приготування страв та кількість кожного продукту.

Безліч вершин від{Б1} до {БМ} відбиває безліч страв, а безліч вершин від {П1} до {ПL} - безліч продуктів (рис.3.1). Дуги – це зв'язок кожної страви з продуктами, які потрібні для її приготування. Дуги відображають вагу брутто Вji - вага i-го продукту, необхідний для приготування однієї порції j-го блюда. Визначаючи одну з вершин Бj отримуємо для страви Бj перелік продуктів та їх вагу, для однієї порції, необхідних його приготування.

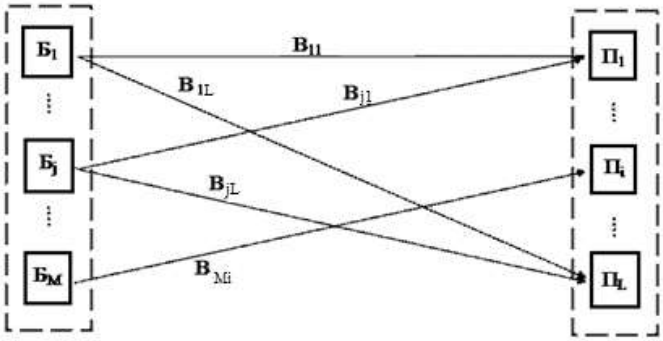


Рисунок 2.13 – Відносини між стравами та продуктами

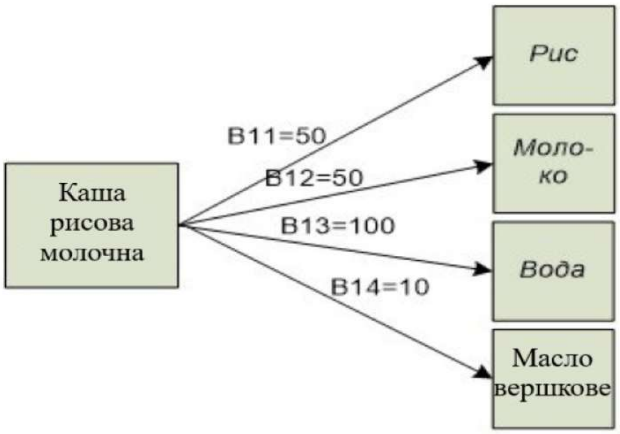


Рисунок 2.14 – Складові страви

Вага кожного продукту для приготування однієї порції страви показано у табл. 2.4 табл. 2.5 табл. 2.6 табл. 2.7 табл. 2.8 табл. 2.9.

Таблиця 2.1 -Вміст мінеральних речовин у продуктах

Продукти, 100г	Мінеральні речовини, мг	
	макроелементи	Мікроелементи

	Mg	До	Na	Ca	P	Zn	Fe	Mn	Cy	I
Рисова крупа	143	200	30	23	333	0,45	1,47	3,74	0,28	0,002
Манна крупа	68	166	3	41	101		1,6			
Гречана крупв	114			56	294		1,8			
Пластівці «Геркулес»	68	355		41	101		1,6			
М'ясо кролика	25	364		21	224		1,6			
Куряче яйце	12	135		50	214		2,5			
Пшеничне борошно вищ.. гат.	16	122		18	86		1,2	10	0,64	10
Крохмаль кукурудзяний	8			17	20		1,6			
Вода питна	1	0,3	0,9	4,5	-					
Пастериз молоко. 2,5-ної жирності	14	146	50	120	90		0,06			
Сметана 20% жирності	8	109	35	86	60		0,2			
Яловичина	22	325	65	9	188		2,7			
Судак	25	280	35	35	230		0,5			
Вершкове масло	0,5	15		12			0,2		0,01	9
Компот із яблук	30	580	12	111	77		6			
Телятина першої категорії	23,7	345	108	12,5	206	3,17	2,92	0,34	0,23	0,003
Сир 0,1%	23	112	41	164	220		0,4			

У моделі харчового раціону застосовується лікувальне (при виразковій хворобі) харчування та харчування на вибір пацієнта (в рамках дієти).

Аналогічно можна висловити відповідність страв Бj та дієт Ді за допомогою графів, в яких дуги відповідають kij – калорійність j-ї страви в і-ій дієті.

Таблиця 2.2 - Зміст білків, жирів, вуглеводів у продукті та його енергетична цінність

Продукти	Білки, г	Жири, г	Вуглеводи, г	Енергетична цінність, кКал
Гречана крупа	4,5	2,3	25	132
Пластівці «Геркулес»	11	6,2	65,7	305
Рисова крупа	8	1	76	345
Манна крупа	10,3	1	73,3	328
Вершки 35% жирності	2,5	35	3	337
М'ясо кролика	21,1	11	-	183
Телятина першої категорії	19,7	2	-	97
Яловичина першої категорії	18,6	16	-	218
Куряче яйце	12,7	11,5	0,7	157
Судак	18,4	1,1	-	84
Масло оливкове	0	99,8	0	898
Пастериз молоко.	2,82	2,5		52
Сметана 20% жирності	2,8	20	3,7	206
Яловичина	18,6	16		218
Вершкове масло	2,5	61,5	1,7	566
Сир 0,1%	16,7	0,10	2	76

Аналогічно можна висловити відповідність страв Бj та дієт Ді за допомогою графів, в яких дуги відповідають kij – калорійність j-ї страви в і-ій дієті.

Приймемо, що Д1 – понеділок, Д2 – вівторок, Д3 – середа, Д4 – четвер, Д5 – п'ятниця, Д6 – субота, Д7 – неділя.

В основі побудови дієт лежать усі фізіологічні процеси, що відбуваються в організмі.

Вага кожного продукту для приготування однієї порції страви показано у табл. 2.4 табл. 2.5 табл. 2.6 табл. 2.7 табл. 2.8 табл. 2.9.

Таблиця 2.4 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда					
	Рис, мг.	Молоко, мол.	Вода, мол.	Олія, мл.	Яйце, шт.	Цукор, мг.
Каша рисова молочна	50	50	100	10		
Омлет		50		10	1	
Молоко		200				
Яйце некруто					1	
Суп рисовий молочний.	30	300	200	10	1/4	2

Таблиця 2.4 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда					
	Рис, мг.	Молоко, мол.	Вода, мол.	Олія, мл.	Яйце, шт.	Цукор, мг.
Каша рисова молочна	50	50	100	10		
Омлет		50		10	1	
Молоко		200				
Яйце некруто					1	
Суп рисовий молочний.	30	300	200	10	1/4	2

Таблиця 2.5 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда

	Каша "Геркулес", р.	Молоко, мол.	Вода, мол.	Олія, мл.	Яйце, шт.	Цукор, р.
Суп вівсяний молочний.	40	150	350	10	1/4	2
Каша вівсяна молочна	50	50	100	10		

Таблиця 2.6 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда						
	Каша манна, р.	Сир, р.	Суха вишня, р.	Крохмаль, р.	Олія злив.	Яйце, шт.	Цукор, р.
Суфле сир. з вишневою підливою парова	10	120	25	5	10	1/2	15

Таблиця 2.7 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда						
	Каша манна, р.	Сир, р.	Суха вишня, р.	Крохмаль, р.	Олія, мл.	Яйце, шт.	Цукор, р.
Суфле сир. з вишневою підливою парова	10	120	25	5	10	1/2	15

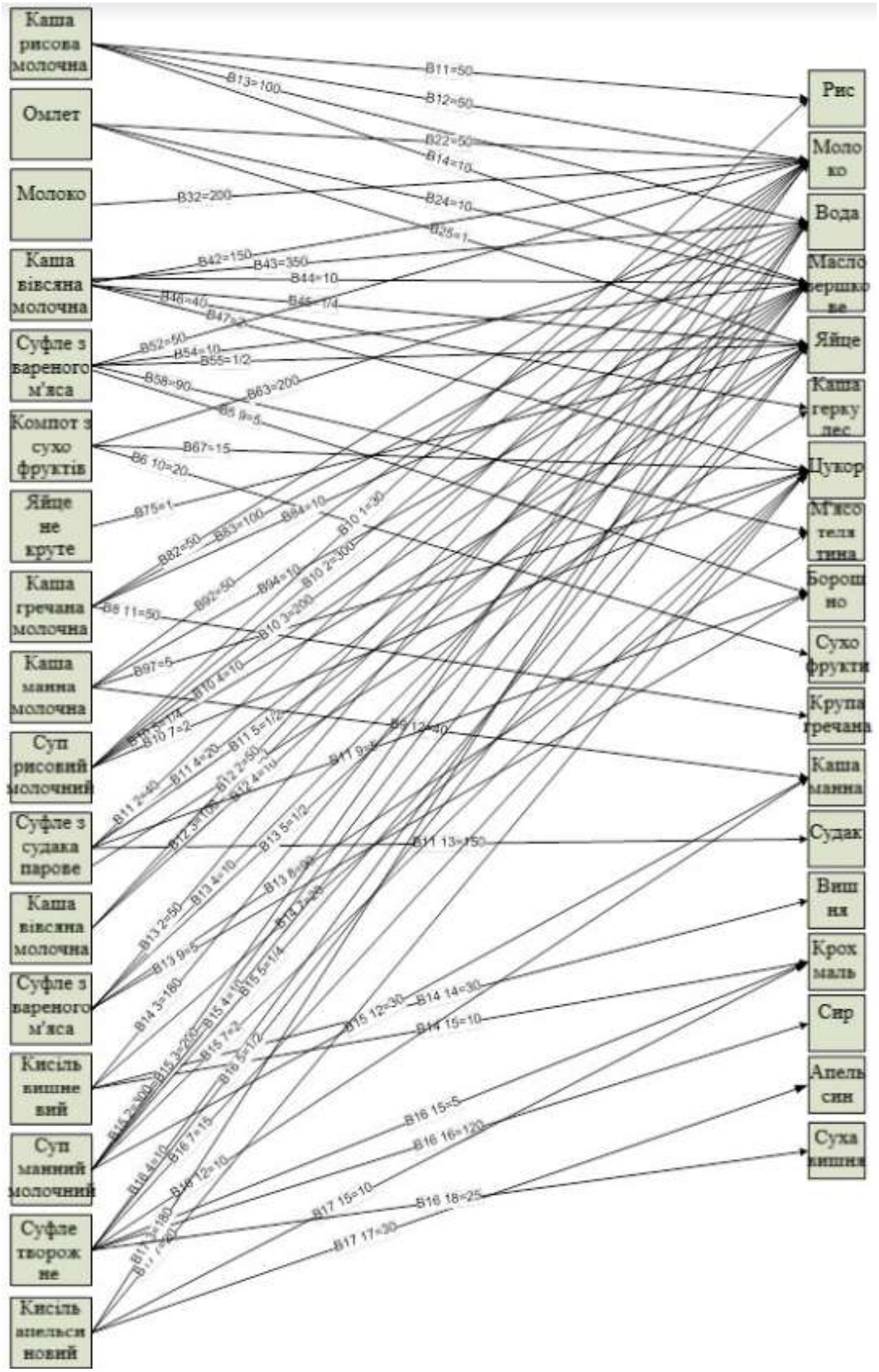


Рисунок 2.15 – Складові страви

Таблиця 2.8 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда						
	Каша манна, р.	Каша грец., г	Молоко, мол.	Вода, мол.	Олія злив.	Яйце, шт.	Цукор, р.
Каша манна молочна	40		50		10		5
Суп манний молочний.	30		300	200	10	1/4	2
Каша гречана молочна		50	50	100	10		

Таблиця 2.9 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда					
	М'ясо,	Риба, р.	Молоко, мол.	Борошно, г	Олія злив., мг.	Яйце, шт.
Суфле із пареного м'яса	90		50	5	10	½
Суфле із судака з маслом парова		150	40	5	20	½
Суфле із вареного м'яса	90		50	5	10	½

Таблиця 2.10 – Відповідність ваги кожному продукту у блюді

страва	Кількість продуктів для блюда					
	Сухо-фрук., М.	Цукор, р.	Крохмаль, р.	Вишня, р.	Апельсини, р.	Вода, мол.
Компот із сухофруктів	20	15				200
Кисель вишневий		20	10	30		180
Кисіль апельсиновий		20	10		30	180

Харчування здорової людини характеризується рядом важливих показників: калорійністю їжі, її хімічним складом та кратністю харчування. Кожна дієта також будується цих принципах, але з надання лікувального впливу необхідно керуватися переважно певними принципами. По-перше, дуже важливу роль відіграє калорійність їжі, по-друге, її хімічний склад, по-третє, різноманітні фізичні властивості їжі: її обсяг, консистенція, температура, по-четверте, велика роль належить режиму харчування.

Щоденний раціон харчування складається із сніданку, обіду, полуденку та вечері (рис. 2.15). У аналізованій дієті пацієнт має 6-разове харчування і складається з: першого сніданку, другого сніданку, обіду, полуденку, першої вечері, другої вечері. Кожен пацієнт може вибрати страви на сніданок, обід, полуденок та вечерю відповідно до своєї дієти.

Харчування при вільному виборі стравскладається з: сніданку першого [З1], сніданку другого [З2] обіду [О], полуденку [П], вечері першого [У1] та вечері другого [У2].

Таблиця 2.11 - Відповідності між стравами з їх калорійністю та дієтами

Продовження таблиці 2.11 наведено на додатках А та Б

Дієти	
Д7	
страва	Калорійність
З1:	
Каша вівсяна молочна	253,3
Яйце некруто	157
Молоко	104
З2:	
Молоко	104
В:	
Суп рисовий молочний	381,13
Суфле із судака з маслом парова	391,45
Кисіль апесьсиновий	89,89
П:	
Яйце некруто	157
Молоко	104
У1:	
Каша манна молочна	250,95
Омлет	257,8
Молоко	104
У2:	
Молоко	104
Разом	2458,52

У меню шість наборів страв залежить від часу харчування:

{Д1З1,…, ДіЗ1,…, ДNЗ1}, {Д1З2,…, ДіЗ2,…, ДNЗ2}, {Д1О,…, ДjО,…, ДМО}, {Д1П,…, ДhП,…, ДLП}, {Д1У1 ,…, ДfУ1,…, ДКУ1}, {Д1У2,…, ДfУ2,…, ДКУ2}.

Розглянемо з якої сукупності страв складаються сніданки, обід, полуденок та вечері.

- Сніданок (БЗ1) = БЗ11 (каша рисова молочна слизова), БЗ12 (омлет), БЗ13 (каша гречана молочна слизова), БЗ14 (молоко), БЗ15 (каша манна молочна), БЗ16 (яйце некруто), БЗ17).
- Сніданок (БЗ2) = БЗ21 (молоко), БЗ22 (яйце некруто)

- Обід (БО) = БО1 (суп вівсяний молочний слизовий), БО2 (суфле з пареного м'яса),БО3(компот із сухофруктів),БО4(суп рисовий молочний слизовий),БО5(суфле з судака з олією парове),БО6(кисіль апельсиновий),БО7(суфле з вареного м'яса),БО8 (кисіль вишневий),БО9 (суп манний молочний),БО10(Суфле сирне з вишневою підливою парове).
- Полудень (БП) = БП1 (яйце некруто),БП2(молоко)
- Вечеря (БУ1) = БУ11 (каша манна молочна), БУ12 (каша вівсяна молочна слизова),БУ13(Каша гречана молочна слизова),БУ14(яйце некруто),БУ15(каша рисова молочна слизова),БУ16(молоко),БУ17(омлет).
- Вечеря (БУ2) =БУ21 (молоко).

З огляду на стан здоров'я пацієнта вибирається дієта, пацієнт вибирає набір страв, а обравши страву – набір продуктів.

Заміна продукту на еквівалентний та його блок схема

Відповідно до дієти можна замінювати телятину на яловичину і кролятину, тобто на нежирні сорти м'яса без сухожиль або птицю без шкіри, або нежирну рибу.

Приклад формування раціону при виразковій хворобі наведено у Додатку В .

Зважаючи на те, що виразкова дієта № 1а не повністю відповідає фізіологічним потребам, нею користуються лише протягом тижня. Потім переходять на дієту № 16, яку одержують протягом наступних 2 тижнів. Дієта №16 майже повністю покриває фізіологічні потреби. Тому при уповільненому перебігу репаративних (відновних) процесів її можна залишити ще на один тиждень. Зазвичай з третього тижня хворого переводять на дієту №1

Таблиця 2.12 – Відповідність страв та їх калорій

Страви	Калорійність
Каша рисова молочна	273,3
Омлет	257,8
Молоко	104
Яйце некруто	157
Суп рисовий молочний	381,13
Суп вівсяний молочний	321,63
Каша вівсяна молочна	253,3
Каша манна молочна	250,95
Суп манний молочний	376,03
Каша гречана молочна	166,8
Суфле із пареного м'яса	283,15
Суфле із судака з маслом парова	391,45
Суфле із вареного м'яса	283,15
Компот із сухофруктів	96,65
Кисель вишневий	94,69
Кисіль апельсиновий	89,89
Суфле сир. з вишневою підливою парова	410,295

4. РЕАЛІЗАЦІЯ АЛГОРИТМУ АВТОМАТИЧНОГО ФОРМУВАННЯ ХАРЧОВОГО РАЦІОНУ ЛЮДИНИ

4.1. Обґрунтування вибору програмної реалізації алгоритму автоматичного формування харчового раціону людини.

Мова програмування C# в першу чергу призначена для операційної системи Windows і доступна в інтегрованому середовищі розробки (IDE) Visual Studio. Мова добре підходить для швидкої розробки додатків, оскільки її IDE є дуже багатофункціональним, а сама мова містить надзвичайно велику колекцію допоміжних інструментів та класів. За продуктивністю C# знаходиться між C++ та Java, причому деякі бенчмарки показують до 17% приросту в продуктивності порівняно з Java [20]. Еталонним компілятором C# є Microsoft Visual C#, який має закритий вихідний код. Однак, специфікація мови не стверджує, що компілятор C# повинен бути орієнтований на Common Language Runtime, або генерувати Common Intermediate Language (CIL), або генерувати будь-який інший специфічний формат. Теоретично, компілятор C# може генерувати машинний код подібно до традиційних компіляторів C++ або Fortran. [21]

Додатки, розроблені на C#, як правило, функціонують тільки в операційній системі Windows, за винятком додатків, розміщених у всесвітній павутині. Однак для C# існує багато компіляторів з відкритим вихідним кодом. Найбільш широко використовуваний компілятор з відкритим вихідним кодом називається Mono, який має подвійну ліцензію GPLv3, MIT/X11, а бібліотеки - LGPLv2 [22]. Використовуючи компілятор з відкритим вихідним кодом, додатки, розроблені на C#, можуть бути розгорнуті на інших операційних системах, таких як Linux. Для виконання додатків на C# необхідна платформа .NET Framework, якщо не розгорнутий компілятор з відкритим вихідним кодом, наприклад, Mono.

Елементи коду задання данних до алгоритму та його роботи наведений на рис. 4.1, 4.2

```
private static bool ContainsDuplicateEfficient(List<string> elements)
{
    if (elements == null)
        throw new ArgumentNullException("elements");

    if (elements.Count > 0)
    {
        for (int count = 0; count < elements.Count; count++)
        {
            for (int innerCount = count; innerCount < elements.Count; innerCount++)
            {
                if (elements[count].Equals(elements[innerCount]))
                {
                    return true;
                }
            }
        }
    }
    return false;
}
```

Рисунок 4.1 – Елемент коду реалізованого алгоритму

Потрібно було розробити алгоритм програми, яка обчислює значення функції в заданому діапазоні. Програма повинна прочитати значення початку і кінця інтервалу, крок збільшення аргументу і значення n.

Алгоритм містить такі частини:

1. Зчитування даних
2. Основний цикл, у якому встановлюється нове значення аргументу, Розраховується значення функції, виводяться на екран значення аргументу і функції та здійснюється збільшення значення аргументу на величину кроку.

```
{
  Form form = new Form();
  Label label = new Label();
  TextBox textBox = new TextBox();
  Button buttonOk = new Button();
  Button buttonCancel = new Button();

  form.Text = title;
  label.Text = promptText;
  textBox.Text = value;

  buttonOk.Text = "OK";
  buttonCancel.Text = "Cancel";
  buttonOk.DialogResult = DialogResult.OK;
  buttonCancel.DialogResult = DialogResult.Cancel;

  label.SetBounds(9, 20, 372, 13);
  textBox.SetBounds(12, 36, 372, 20);
  buttonOk.SetBounds(228, 72, 75, 23);
  buttonCancel.SetBounds(309, 72, 75, 23);

  label.AutoSize = true;
  textBox.Anchor = textBox.Anchor | AnchorStyles.Right;
  buttonOk.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;
  buttonCancel.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;

  form.ClientSize = new Size(396, 107);
  form.Controls.AddRange(new Control[] { label, textBox, buttonOk, buttonCancel }
  form.ClientSize = new Size(Math.Max(300,label.Right+10), form.ClientSize.Height
  form.FormBorderStyle = FormBorderStyle.FixedDialog;
  form.StartPosition = FormStartPosition.CenterScreen;
```

Рисунок 4.2 – Задання данних до алгоритму

Далі здійснюється створення одного, або декількох вікон, після чого до них додаються візуальні елементи управління. Для цих елементів створюються та реєструються функції обробки певних подій. Основний цикл отримання та обробки подій здійснюється стандартними засобами – за допомогою каркасу застосунку (framework). Потім додаються необхідні елементи та прописуються функції обробки подій.

4.2 Розробка програмного продукта для вибору раціона харчування.

В якості методу реалізації було вибрано розробити гру-раннер яка буде використовувати алгоритм автоматизації, розроблений раніше, але буде орієнтована на молодшу аудіторію.

В даному проекті я буду використовувати рушій Unity для створення нескінченної гри - раннера, що може використовувати алгоритм для автоматичного формування раціону людини щоб представляти гравцеві на шляху продукти та страви, які необхідні його харчовому раціону. У даному розділі я опишу більшість можливостей рушія Unity, та максимально використовуватимуть їх у нашому процесі. Оскільки гра робиться не тільки за допомогою ігрового рушія, мені також потрібно дослідити зв'язки між Unity та іншим програмним забезпеченням, пов'язаним з іграми.

На сьогоднішній день існує багато ігрових рушіїв, які дозволяють людям створювати власні ігри більш простим і зручним способом, і Unity є одним з найбільш підходящих ігрових рушіїв для початківців. Unity3D використовується для розробки відеоігор для різних платформ, таких як Web, PC, MAC, IOS та PS3 тощо. З іншого боку, Unity3D підтримує різноманітні мови програмування, що дозволяє програмістам програмувати на звичній для них мові.

В останні роки ігрова індустрія вступила в стадію бурхливого розвитку. З'явилося більше різних типів ігор на різноманітних нових платформах, оскільки відносно низька вартість і величезні прибутки мотивують все більше людей долучатися до цієї сфери. Якщо раніше люди грали в ігри тільки на консолях або комп'ютерах, то зараз люди з більшою охотою грають в ігри на мобільних телефонах. Різноманітність платформ призводить до більш інтенсивної конкуренції. З іншого боку, на додаток до гри в ігри, все більше людей готові брати участь у процесі виробництва ігор.

У цьому розділі я почну з базових теоретичних основ ігрового дизайну. Потім я представлю рушій Unity. Після цього я опишу рівні реалізації проекту. Реалізація містить повні деталі ігрового процесу. Нарешті, я зроблю висновок про цей проект, що передбачає майбутній розвиток. Завдяки кожному етапу вивчення та дослідження в цьому проекті, він дозволяє нам мати краще уявлення про процедуру виробництва гри, крім того, проект допомагає визначити труднощі, які можуть виникнути в процесі виробництва, і як вирішити ці проблеми.

Виробництво ігор часто включає в себе багато елементів. Оскільки в процесі виробництва гри беруть участь різноманітні елементи, перш ніж почати робити цю нескінченну запущену гру. Має сенс пояснити кожен елемент, що міститься в процесі, щоб мати базове розуміння перед створенням гри. Тому я поясню наступні поняття, які можуть з'явитися в грі. Це розробка стратегії гри, баланс гри, ігровий рушій, мови програмування і так далі. Почнемо з розробки стратегії гри.

Існує багато ігор, які не є складними, і їх геймплей також не є чимось видатковим. Але після того, як гра була випущена, вона отримала широку популярність. Такі ігри були доступні на ринку в перші роки, і зміст ігор є порівняно оригінальним. Цінна ігрова ідея є однією з причин того, що ці ігри є успішними і дійсно популярними серед гравців.

Зазвичай геймдизайнери розробляють ігри для того, щоб зацікавити гравців. Крім того, геймдизайнери повинні враховувати інші важливі складові гри, такі як інтерфейс користувача, дизайн персонажів та ігрових історій. Крім того, гейм-дизайнерам необхідно враховувати, чи буде цей знак мотивувати гравців продовжувати грати і продовжувати грати, а не просто дублювати процеси і відчувати себе втомленими від цього незабаром.

Різні обставини у віртуальному світі можуть приносити гравцям задоволення. У віртуальному ігровому світі початковим наміром гравців є розвага, а не пошук справедливості та балансу. Більше того, гравці витрачають

час на ігри для того, щоб задовольнити себе і дати собі достатньо щастя. Тому, виходячи з цих умов, дизайн гри може бути дуже гнучким. Крім того, геймдизайнер повинен мати зріле мислення на етапах планування гри. В іншому випадку, ідеї принесуть в гру незбалансовані або фатальні помилки.

Більше десяти років тому ігри були дуже простими, а потужність ігор була дуже маленькою. Крім того, геймдизайнерам, які створювали кожну гру, часто доводилося переписувати всі коди. Було багато дублювання роботи і це забирало багато часу. З іншого боку, програмісти підвели регулярний патерн. Вони використовували загальний код в однотипних іграх. Таким чином, програмісти могли значно зменшити витрати на розробку та скоротити цикл розробки. Таким чином, ці загальні коди поступово стали прототипом ігрового рушія. Зрештою, з розвитком технологій він перетворився на сучасний ігровий рушій. На сьогоднішній день ігрові рушії вже перетворилися з ранніх ігрових аксесуарів на головну роль в ігровому дизайні. Від того, наскільки потужним є рушій, багато в чому залежить, якого ефекту можна досягти в грі.

Відмінний ігровий рушій зазвичай містить кілька переваг. По-перше, рушій повинен мати повну ігрову функцію. У наш час ігровий рушій - це вже не просто рушій 3D-графіки. Він охоплює, наприклад, 3D-графіку, обробку звуку, роботу штучного інтелекту, фізичні зіткнення та інші компоненти гри. Крім того, відмінний ігровий рушій включає в себе потужний редактор, в тому числі сцени, анімації моделей тощо. Чим потужніший редактор, тим більше ефектів він може внести в гру. Крім того, потужна підтримка сторонніх плагінів робить розробку ігор більш плавною, таких як 3Ds Max, Maya та інше стороннє програмне забезпечення. Крім того, ігровий рушій повинен надавати простий та ефективний набір для розробки програмного забезпечення, який допомагає розробникам ігор швидко розпочати роботу.

У сучасній індустрії ігровими рушіями для основної маси ігрових компаній є Unity3D, Source 2, Unreal Engine та CryENGINE. Починаючи з Unity3D, це дійсно доступний рушій для розробників ігор; він має незрівнянну

кількість користувачів у порівнянні з іншими рушіями. Що ще важливіше, розробникам потрібно заплатити лише один раз і незалежно від того, наскільки успішною стала їхня гра, їм не потрібно турбуватися про те, що Unity отримає свою частку від доходу. Це, безумовно, є привабливою особливістю, особливо для компаній-початківців та незалежних розробників. Крім того, поріг освоєння Unity3D дуже низький і він простий у використанні. Крім того, Unity3D має потужну підтримку спільноти розробників та сумісний з усіма ігровими платформами. З іншого боку, Unity3D має обмежену кількість інструментів, тому розробникам необхідно створювати власні інструменти. Внесення складних та різноманітних ефектів у гру займає набагато більше часу.

Вихідні тексти рушія відносно відсталі в порівнянні з іншими рушіями. Графічні технології відстали, але світлові та фізичні системи рушія дуже хороші. Найпотужнішою частиною є оригінальний дизайнерський задум вихідного двигуна. Він дуже підходить для гравців, які беруть участь у розробці гри. Крім того, вихідний движок має дуже потужний набір для розробки програмного забезпечення. Однак, рушій Source підходить тільки для платформи ПК і з ним важко працювати. Рушій Source engine 2 був анонсований, і він має всебічне оновлення у порівнянні з попередньою версією.

Unreal Engine - найпопулярніший ігровий рушій для створення ігор високого класу. Рушій Unreal широко використовується в ігрових компаніях. Також Unreal Engine має потужну підтримку спільноти розробників. Користувачам доступна велика кількість відеоуроків та ресурсів.

Крім того, Unreal Engine має найкращу підтримку та оновлення рушія. Кожне оновлення додає нові інструменти в рушій, а управління відносно просте. Але, ліцензійні умови підходять тільки для великих компаній, Unreal engine буде отримувати акції від гри, коли дохід гри буде більше п'ятдесяти

тисяч доларів. Також є деякі інструменти, які складні у використанні і вимагають більш високого порогу навчання.

CryENGINE отримав велике визнання від розробників ігор завдяки високій якості графічних можливостей гри. Можливості художнього програмування інструментів Flowgraph дуже потужні. Також CryENGINE має найпотужніші звукові інструменти. Ці інструменти можуть принести найкращий сенсорний досвід гравцям гри. Крім того, CryENGINE надає найпростішу технологію коду інтерфейсу користувача для нових розробників ігор. Але його спільнота розробників є відносно слабкою і вимагає високого порогу навчання.

Під час ігрового процесу однією з ключових частин гри є написання кодів. Перед початком програмування гри нам потрібно вибрати мову програмування, яку ми будемо використовувати для написання коду гри. У сучасному середовищі виробництва ігор розробники ігор мають багато варіантів мов для вибору. Кожна мова має свої особливості, а основними мовами програмування є JavaScript, C, C++, C#, Python та інші.

JavaScript має дуже високий попит на веб-додатки. JavaScript в основному використовується для веб-браузерів для забезпечення розширеного користувацького інтерфейсу та динамічних веб-сайтів. З розвитком Node.js JavaScript став основним напрямком серверної розробки. JavaScript простий у використанні і має схожий синтаксис з Java. Крім того, JavaScript можна редагувати за допомогою будь-якого текстового редактора. JavaScript може виконати програму, яка потребує лише браузера. З іншого боку, JavaScript не підходить для розробки великих додатків.

C та C++ базуються на мові C. Це найпопулярніші мови програмування ігор. C часто використовується для систем і додатків, таких як вбудовані системні додатки. C++ є розширеною версією мови C. Вона використовується для розробки системного програмного забезпечення, додатків, драйверів

пристроїв, високопродуктивних серверних і клієнтських додатків і розважального програмного забезпечення, такого як відеоігри.

C# - це точна, проста та об'єктно-орієнтована мова програмування, яка походить від C та C++. Крім того, C# успадкувала потужні можливості C та C++. В той же час, C# позбулася деяких складних характеристик C та C++. З іншого боку, C# не застосовується для написання дуже високопродуктивного коду, а також не має ключових особливостей, необхідних для високопродуктивних додатків.

Python - це динамічна мова, яка використовується для розробки широкого спектру додатків. Python часто легше писати коди порівняно з іншими мовами програмування. Синтаксис Python простий і зрозумілий; Python має багату і потужну бібліотеку класів. Крім того, Python може легко зв'язувати модулі інших мов разом, особливо C та C++. З іншого боку, його продуктивність відносно низька.

Перед початком гри гравець задає данні що стосуються його персонального раціону. У цій грі ми будемо керувати персонажем, який біжить з лівого боку екрану в правий. Персонаж не припиняє біг під час гри, якщо тільки персонаж не загине. Гравець може керувати персонажем тільки для того, щоб стрибати, щоб уникнути перешкод або ворогів. Також у гравця є тільки один шанс під час гри. Якщо персонаж помирає, гра закінчується. В процесі гри система буде фіксувати рахунок, виходячи з відстані пробігу. Якщо персонаж помирає, запис зупиняється. Коли почнеться наступна гра, на екрані буде відображатися попередній найвищий результат. Крім того, персонаж може збільшувати рахунок, отримуючи ігрові предмети. Крім того, в процесі розробки гри до ігрових предметів можуть додаватися деякі інші можливості. Таким чином, у цій грі буде показано більшу частину основного змісту нескінченної бігової гри.

Перед початком створення гри, нам потрібно спочатку налаштувати середовище, щоб гра працювала належним чином. У рушії Unity3D за

забезпечення такого середовища відповідають сцени. Сцени містять всі об'єкти гри. Вони також дозволяють створювати інші частини гри, такі як головне меню, різні ігрові рівні та інші деталі. Крім того, сцени можуть використовуватися як самостійний ігровий рівень, тому ми можемо створювати різні ігрові рівні, перемикаючи сцени. У кожній сцені розробники можуть розміщувати своїх персонажів, будівлі та перешкоди, а потім будувати свою гру.

В процесі створення гри, GameObject є одним з найважливіших об'єктів, які ми будемо використовувати в Unity. Визначення GameObject трохи розпливчасте. Це може бути будь-який об'єкт у грі, з яким гравець може взаємодіяти. Перш ніж GameObject стане конкретною річчю, нам потрібно надати об'єкту деякі спеціальні властивості, щоб GameObject міг стати різними фігурами в грі, такими як людина, будівля або щось, що гравці можуть використовувати в ігровій сцені. Тому ми можемо розглядати його як контейнер. Коли нам потрібно, щоб об'єкт став певним персонажем або середовищем, нам потрібно додати різні властивості до GameObject. Більше того, GameObject зазвичай містить деякі базові властивості, і кожна властивість матиме різний вплив на GameObject. GameObject також може впливати на значущість об'єкта в грі. Тому розуміння кожної властивості є дуже важливим.

Функції Transform GameObject в Unity призначені для керування його фізичними параметрами. Transform використовується для керування положенням GameObject'a у грі. Rotate (Обертання) використовується для керування орієнтацією GameObject'a. Масштаб (Scale) використовується для керування розміром GameObject'a у грі. Трансформація маніпулює у 3D просторі по осях X, Y та Z, а у 2D просторі тільки по осях X та Y, що показано на рисунку 4.1.

Функції трансформації зазвичай налаштовуються за допомогою скриптів, щоб трансформація могла бути ефективною в грі. У різних мовах

програмування форма скриптів може бути досить різною. У даному проекті ми будемо використовувати C# для виконання цієї гри. Тому, при налаштуванні GameObject на мові C#, базова форма всередині скрипту зображена на рисунку 4.2.

Функції трансформації зазвичай налаштовуються за допомогою скриптів, щоб трансформація могла бути ефективною в грі. У різних мовах програмування форма скриптів може бути досить різною.

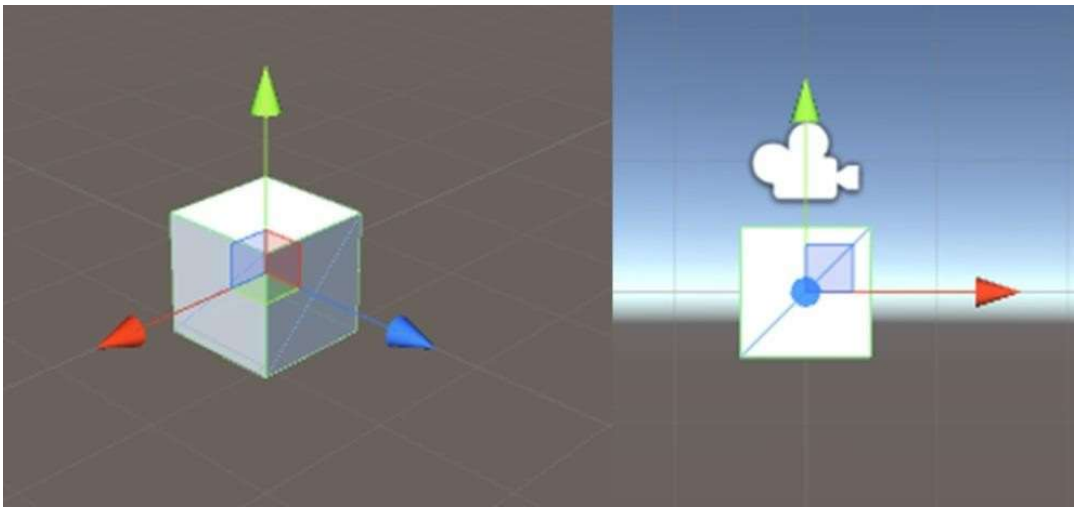


Рисунок 4.1 - Осі X, Y та Z у 3D та 2D просторі

У даному проекті ми будемо використовувати C# для виконання цієї гри. Тому, при налаштуванні GameObject на мові C#, базова форма всередині скрипту зображена на рисунку 3.2.

```
gameObject.transform.position = Vector3(x, y, z);
gameObject.transform.Rotate(new Vector3(x, y, z));
gameObject.transform.localScale = new Vector3(x, y, z);
```

Рисунок 4.2 - Базова форма скрипту перетворення GameObject Tranform

На рисунку 4.2 позиція та масштаб зберігаються як vector3, тому що GameObject знаходиться в 3D середовищі, а це означає, що ми можемо налаштовувати осі X, Y та Z. З іншого боку, у 2D середовищі ми можемо використовувати лише осі X та Y.

У Unity, наслідування є однією з найважливіших концепцій під час процесу створення гри. Наприклад, якщо GameObject є батьком іншого GameObject, його дочірній GameObject зробить ті ж самі зміни трансформації, що і його батько. У грі transform.parent може бути використаний для зв'язування GameObject'ів або камери з персонажем або іншими GameObject'ами. Transform.parent - це, по суті, скрипт, як показано на рисунку 3.3.

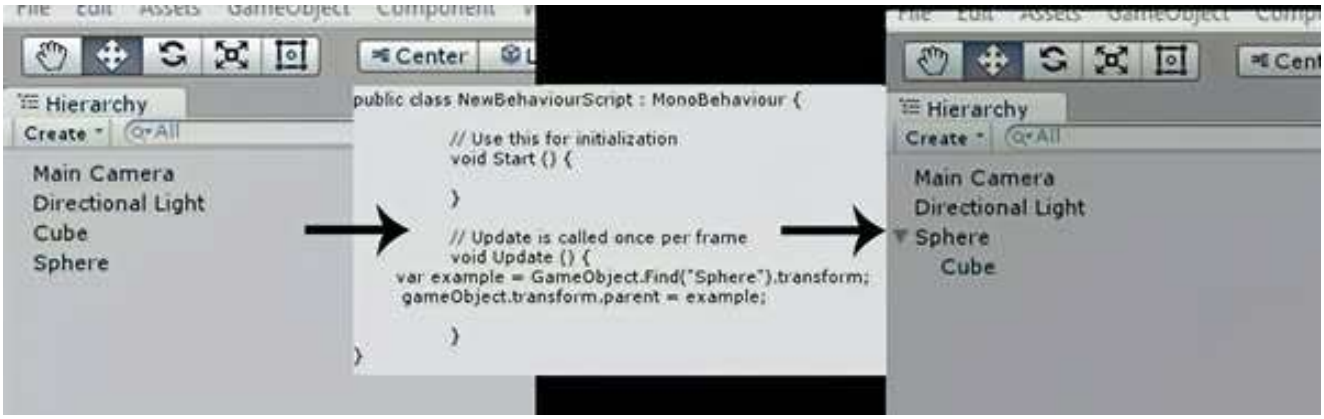


Рисунок 4.3 - Процес роботи Transform.parent

Також transform.parent може бути виконаний шляхом простого перетягування GameObject'а на інший GameObject. Але в більшості випадків ми будемо використовувати метод скрипту, оскільки він може викликати деякі непередбачувані проблеми. У грі часто використовуються зіткнення ігрових об'єктів. Тому нам потрібно створити середовище для ігрових об'єктів, яке називається Колайдер. Крім того, Collider невидимий і йому не потрібно мати таку ж форму, як сітка GameObject.

У 3D-грі існують Колайдер-бокс, Колайдер-сфера, Колайдер-капсула, Колайдер-сітка, Колайдер-колесо та Колайдер-рельєф. У 2D грі є Колайдер-бокс та Колайдер-сфера 2D. 3D колайдери як на рисунку 3.4. У більшості випадків найбільш ефективними колайдерами є колайдер типу Колайдер-бокс та Колайдер-сфера. Крім того, менша форма та розмір ігрового об'єкту може зменшити навантаження на процесор.

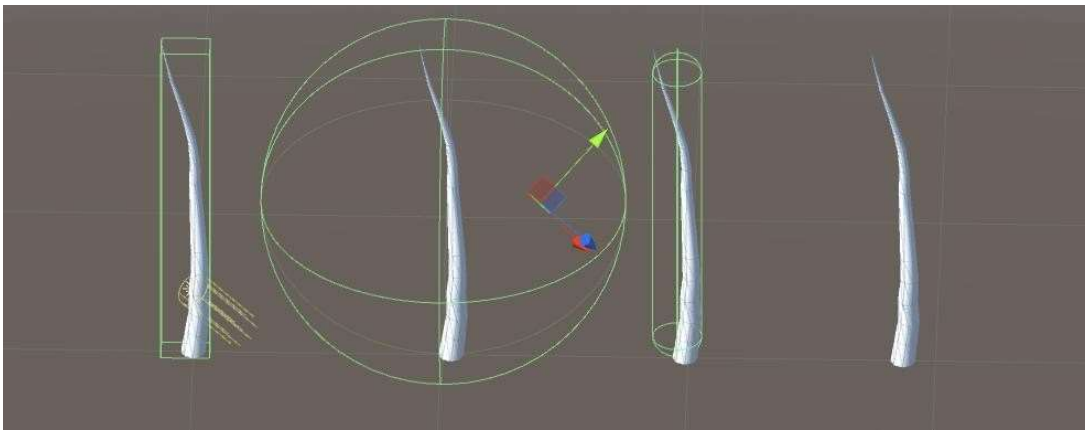


Рисунок 4.4 - Капсульний колайдер, сітчастий колайдер, колайдер-колесо та рельєфний колайдер.

Наприклад, будівлі та базові предмети в грі не мають багато форм, які потрібно обчислювати процесору, і в цьому випадку для таких ігрових об'єктів можна використовувати Колайдер-бокс. Більше того, коли Колайдер-бокс та Колайдер-сфера не підходять для ігрового об'єкту, ми можемо використати сітчастий колайдер, який відповідає сітці ігрового об'єкту. Колайдер сітка потребує більшої продуктивності процесора, ніж примітивні колайдери, тому не слід використовувати цей колайдер часто.

У деяких випадках колайдери можуть бути встановлені без Rigidbody, наприклад, при створенні підлог, будівельних стін та статичних об'єктів у нашій грі. Тому ми зазвичай не переставляємо ці статичні колайдери в позицію "Трансформація", тому що ці статичні колайдери будуть впливати на продуктивність ігрового рушія. Ці GameObjects з жорстким тілом можуть взаємодіяти зі статичними колайдерами, але статичні колайдери не будуть реагувати на будь-які зіткнення, оскільки вони не мають жорсткого тіла.При додаванні персонажа в гру, зазвичай дизайнеру потрібно додати матеріали, шейдери та текстури для того, щоб персонаж виглядав більш реалістично. Перш за все, матеріал визначає, як має бути відрендерена поверхня, і варіант рендерингу зазвичай залежить від того, який шейдер використовуватиметься у матеріалі. Потім, шейдер обчислює колір, що передається матеріалом.

Нарешті, текстура використовується для растрового відображення зображень на GameObject. У більшості випадків ми використовуємо станадартний шейдер для GameObjects. Як новий тип вбудованого шейдера, він може впоратися з більшістю ситуацій. Крім того, стандартний шейдер може допомогти розробникам скоротити час, витрачений на вибір шейдера зі списків шейдерів.Коли ми граємо в гру, ігровий інтерфейс пов'язаний з усіма взаємодіями, які ми здійснюємо під час гри. Крім того, ігровий інтерфейс відноситься до взаємодії між людиною і комп'ютером, логіки роботи і загального красивого дизайну ігрового інтерфейсу. Крім того, хороший UI дизайн не тільки зробить продукт відповідним особистим уподобанням, але і зробить так, що продукт стане більш зручним, простим, вільним і повністю відображатиме характеристики гри. Зазвичай, ігровий інтерфейс включає в себе ігрове меню, управління клавіатурою, управління мишею, внутрішньо ігрові іконки і всі елементи, за допомогою яких гравці могли б взаємодіяти в грі. На рисунку 4.5 наведений ігровий інтерфейс продукту.

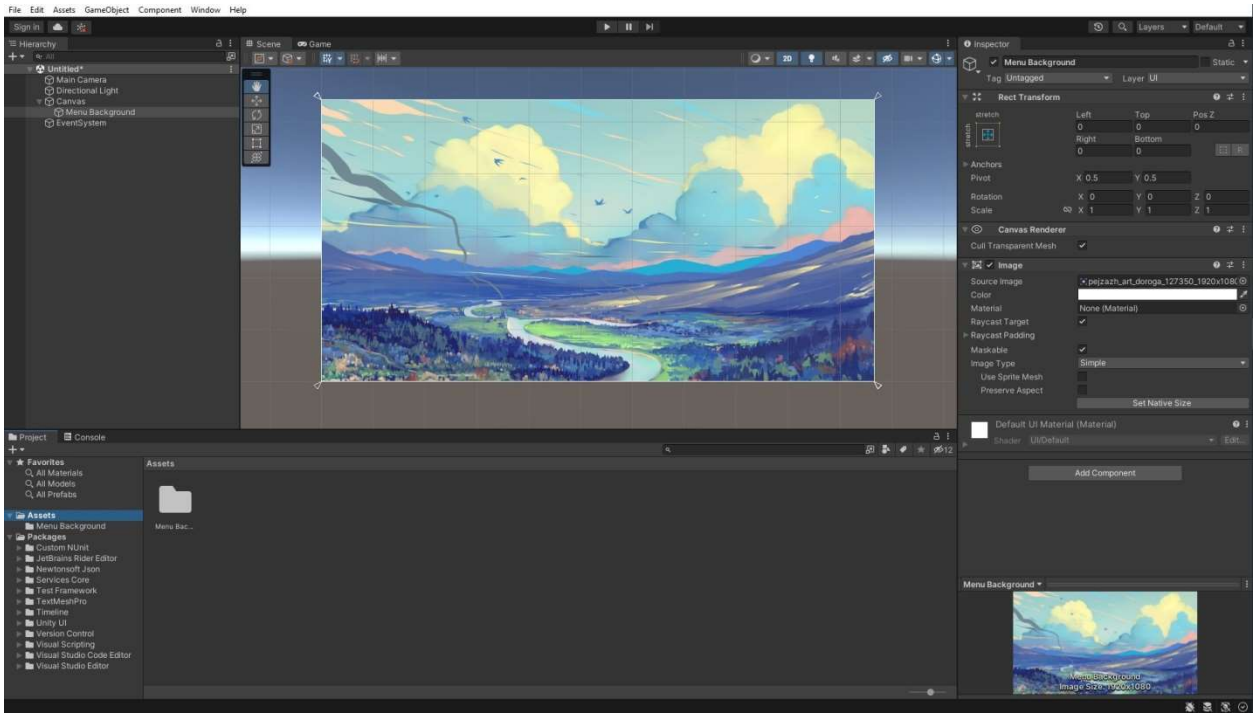


Рисунок 4.5 – Ігровий інтерфейс продукту

В Unity3D у нас є два варіанти графічного інтерфейсу на вибір, це UnityGUI і Next-Gen UI. UnityGUI - це офіційно вбудована система графічного

інтерфейсу, а Next-Gen UI - це плагін для Unity3D. У багатьох іграх гейм дизайнери вважають за краще використовувати NGUI для завершення системи інтерфейсу, тому що UnityGUI набагато складніший в операційній фазі. Крім того, розробникам потрібно написати сценарій всього інтерфейсу для міток, текстур та інших елементів інтерфейсу в грі. Крім того, Next-Gen UI простіше у використанні, оскільки елементи інтерфейсу Next-Gen UI є GameObjects в Unity3D. У Next-Gen UI можна легко редагувати свої мітки, створюючи віджети, які переглядаються камерою, щоб перевірити, як виглядає інтерфейс з вікна гри.

Ігровий персонаж є однією з найважливіших частин у грі, оскільки вплив анімації персонажа є значним у грі. анімаційному автоматі станів (Animation State Machine). Стан анімації часто використовується у грі. Крім того, не тільки в рольових іграх використовується машина станів анімації, але й у разі перемикання дії будь-якого об'єкту в грі, як показано на рисунку 4.6.

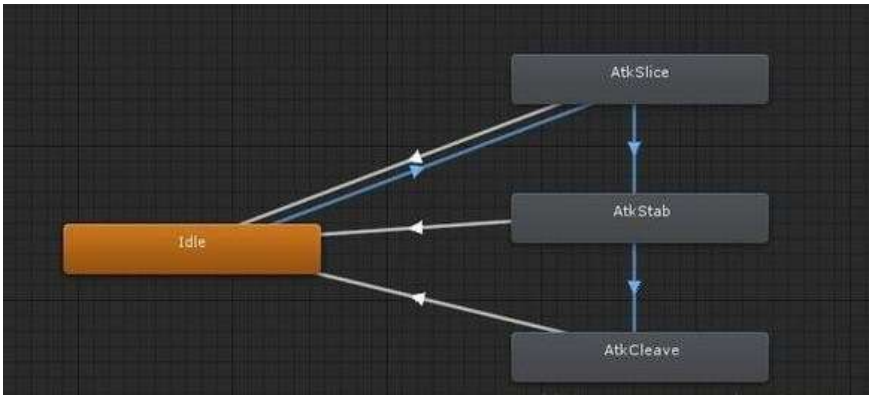


Рисунок 4.6 – Анімаційний автомат станів в Unity3D

Існує декілька основних моментів використання станів анімації у грі. Перш за все, ігровий персонаж зазвичай має 4-5 основних дій, таких як бездіяльність, ходьба, атака, поранення і так далі. Всі ці дії називаються станами. Нам потрібно з'єднати всі ці стани за допомогою параметра в анімаційний автомат станів. Потім, нам потрібно надати основні стани анімації ворогам. В Unity3D використання Animation State Machine для редагування ШІ ворогів є найшвидшим і найефективнішим способом.

Зазвичай вороги будуть мати деякі додаткові стани, такі як переслідування, втеча, смерть і так далі. Нарешті, ми повинні подумати про інструкції для акторів, щоб підтримувати стани дій персонажа. Ми виявили, що існують лише стани, коли стани дій персонажа були завершені в машині станів анімації, але нам також потрібно відредагувати деякі додаткові деталі, наприклад, трансформувати або масштабувати персонаж. За ці ситуації відповідає стан інструкцій персонажа. Це ще один шар над станом дії персонажа. Наприклад, інструкція по ходьбі використовує стан анімації ходьби і утримує ігрового персонажа в стані ходьби. Тоді нам потрібно мати справу тільки з Трансформацією для ігрового персонажа. Тепер у нас є і дія, і Трансформація, так що вся інструкція руху завершена.

У більшості ігор ШІ завжди є невід'ємною частиною. Чудовий дизайн штучного інтелекту може забезпечити гравцям приємний досвід гри та зробити візуальні ефекти та анімацію цікавішими. Оскільки основною метою штучного інтелекту є створення ігрового досвіду для гравців, тому ШІ має підтримувати загальний ігровий досвід, а не просто показувати, наскільки розумний наш ШІ. Під час створення чудового штучного інтелекту найважливішим є те, який ігровий досвід він привніс у гру та які проблеми він може допомогти нам у грі. З іншого боку, поганий штучний інтелект вважається серйозною загрозою в грі, яка може зіпсувати ігровий досвід, який ми намагаємося створити. І це також дає руйнівний вплив на гру від початку до кінця.

Ігровий штучний інтелект можна просто зрозуміти як персонажа, яким керує комп'ютер. Ці розумні персонажі можуть визначати будь-які зміни в навколишньому середовищі або подіях, так що ігровий ШІ може виробляти певну поведінку від дій гравця. Є деякі базові елементи штучного інтелекту, наприклад базова логіка штучного інтелекту, основні навички штучного інтелекту, основні властивості штучного інтелекту тощо.

Основну логіку ШІ можна розділити на сприйняття, дію та реакцію. По-перше, сприйняття — це здатність ШІ визначати зміни в навколишньому середовищі, яку дає розробник гри. Якщо встановлено, що штучний інтелект має лише цю перспективу, то вороги реагуватимуть лише тоді, коли гравець перебуває в цьому полі огляду. Тоді ШІ буде вирішено виконати низку дій, ці дії ШІ встановлюються на основі низки правил і логічного порядку розробником ігор. Наприклад, якщо розробник ігор хоче створити яскраве місто, ми можемо створити багато різних персонажів ШІ, і деякі з них можуть бігати з одного місця на інше по вулиці, деякі з них можуть спілкуватися на площі тощо. У деяких іграх для виживання, зомбі чекатимуть нагоди атакувати гравця після визначення позиції гравця. Якщо гравець стріляв у зомбі, вони намагатимуться ухилитися.

Створюючи базові навички штучного інтелекту, гейм дизайнер повинен спочатку проаналізувати основні можливості ігрового штучного інтелекту, що може допомогти гейм дизайнеру значно спростити розширення можливостей. Крім того, у більшості ігор здатність ігрового штучного інтелекту може, наприклад, мати потенціал виявлення погрози та підтвердження особистість іншого (друга чи ворога). Крім того, ми можемо використовувати ці базові можливості як параметри для розробки різних типів ШІ. Наприклад, ворог може виявити атаку, біг або смерть, також може виявити використання навичок, атаку, біг або смерть.

Дизайн базових властивостей для штучного інтелекту надзвичайно важливий, оскільки він впливає на баланс гри. Більшість основних властивостей ШІ в іграх можна розділити на чотири частини, включаючи ідентичність, бойові параметри (цінність життя, атака та захист), дальність взаємодії (відстань погоні, дальність виявлення ненависті, відстань атаки) та агресивну поведінку (часто - навички атаки). Розробники ігор можуть коригувати основні властивості та розробляти різні індивідуальні особливості ШІ. Наприклад, властивостями воїна можуть бути висока цінність життя,

нормальна вартість атаки та низька швидкість, а властивостями вбивці можуть бути низька цінність життя, висока цінність атаки та висока швидкість. Відмінності між ними полягають у зовнішньому вигляді, анімації, життєвій цінності та швидкості. Але у них однакова логіка ШІ.

Дуже важливо створювати ШІ гри з цими трьома елементами. По-перше, нам потрібно встановити основне визначення завдання ШІ в грі. Одиночне завдання штучного інтелекту часто вимагає комбінації дизайну складних рівнів, щоб створити базовий шаблон завдання. Перш ніж гейм дизайнер приступить до детального концептуального дизайну, він повинен спочатку зрозуміти, де в грі знаходиться основна частина геймплею, яка частина гри матиме ігровий штучний інтелект і за якого персонажа ми граємо. По-друге, нам потрібно розробити базові навички та базові властивості ШІ в шаблоні завдання. Коли комбінація різноманітних здібностей і атрибутів виконана, ми можемо досліджувати різноманітніші завдання. Потім нам потрібно розробити логічну схему штучного інтелекту, оскільки чітко структурований процес запуску ігрового штучного інтелекту допоможе програмістам краще зрозуміти поведінку ігрового штучного інтелекту. Як показано на рисунку 4.7.

Його також можна розуміти як машину станів, ШІ завершить певну поведінку в одному стані, коли будуть виконані задані умови, і ШІ перейде з одного стану в інший стан. Таким чином, логіка ШІ полягає в перемиканні між станами ШІ.

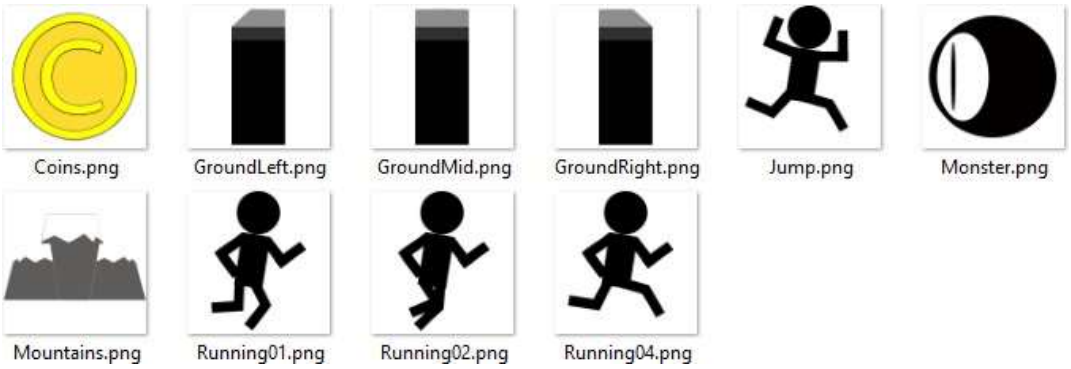
В якості кінцевої мети проекту мені потрібно створити нескінченну гру, яка б охоплювала всі можливості. За основними правилами гри, гравець буде бігати зліва направо по ігровій сцені. У грі будуть деякі перешкоди. Гравці повинні перестрибувати через ці перешкоди, інакше гра закінчиться.

Візуальний досвід гри для гравців часто досягається за допомогою ігрових наборів. Ігрові ресурси часто поділяються на художні ресурси та ресурси сценарію. Ігрові художні ресурси відносяться до ігрових моделей, текстур моделей, фонової музики гри тощо. Ці арт-активи складають

внутрішньо ігрові об'єкти та користувацькі інтерфейси. Натомість, скриптові ресурси є основними частинами гри. Ці ресурси запрограмовані кодом, який відповідає за адміністрування ігрового процесу та правил гри.

Під час цього ігрового проекту зображення призначені для створення користувальницького інтерфейсу, персонажів та сценічних елементів. Оскільки цей ігровий проект є 2D грою з бічними скролерами, нам не потрібна ніяка 3D ігрова модель в грі. Всі персонажі та ігрові елементи (наприклад, земля, фон, перешкоди тощо) розроблені та створені за допомогою Photoshop під час роботи над проектом. Як показано на рисунку 4.8.

Рисунок 4.8 - Ассети для програмного продукту біотехнічної системи підбору раціону харчування людини



Для персонажа було зроблено кілька наборів зображень для дій персонажа, таких як стрибки та біг. Кожне окреме зображення представляє один кадр дії. Анімація буде вироблятися шляхом зациклення набору зображень від першого до останнього зображення. Весь набір зображень дії буде налаштований в машині стану анімації. Крім того, буде необхідно налаштувати часову шкалу кожного зображення в машині стану анімації, щоб зробити дію природною.

Земля розділена на три частини, ліву, праву і середню, так як в процесі гри земля генерується справа наліво. Крім того, між майданчиками буде багато проміжків. Саме тому поле повинно мати початок і кінець.

Фон розділений на дві частини. Небо буде створено одним статичним зображенням. З іншого боку, гори і хмари будуть динамічно генеруватися, як

і земля, також з правого на лівий бік. Але швидкість руху гір і хмар буде меншою, ніж у землі. Також хмари і гори мають різну швидкість.

Крім того, скриптові ресурси також мають важливе значення для ігрового проекту. Сценарії є ключем до того, щоб зробити ігровий процес та продуктивність правильним чином. У цьому проекті я буду використовувати C# для створення скриптів.

Починаючи ігрову сцену, нам спочатку потрібно створити основну землю, щоб головний герой міг ходити по ній. По-перше, мені потрібно відредагувати ці арт-активи землі, щоб вона могла вписатися в гру. Одна з найважливіших речей - це додати 2D колайдер на землю, щоб земля могла контактувати з головним героєм. Також 2D колайдер може дозволити персонажу стояти на арт-активах у фізичному середовищі. У цьому проекті всі налаштування ігрового активу редагуються в розділі інспектора. Цей розділ дозволяє користувачеві змінювати початкові значення об'єкта, який містить основні значення, такі як Transform і Renderer, а також ми можемо додавати нові компоненти пізніше. Також інспектор дозволяє користувачеві змінювати значення в ігровому режимі роботи. Наземні активи не потребують редагування трансформації, але для 2D колайдера я змінив його зміщення та розмір колайдера, щоб він вписувався в ігрові активи, як показано на рисунку 4.9.

Також необхідно створити префаби майданчиків для подальшого використання. Основною метою використання префабів є надання можливості багаторазового використання ігрових об'єктів та ресурсів під час ігрового процесу. Префаби можуть покращити використання ресурсів та ефективність розробки в даному проекті. Важливо створити префаби для кожного з них. Також у розділі ієрархії було створено деякі порожні ігрові об'єкти, які називаються GroundsLayer, GameLayer і BgLayer. Ці порожні ігрові об'єкти можуть допомогти зібрати ці інстанції під час ігрового процесу. Таким чином також можливо зробити ієрархію більш стислою та конкретною. Під час

ігрового процесу, кожна збірна модель ґрунту буде генеруватися під підшаром GroundsLayer.

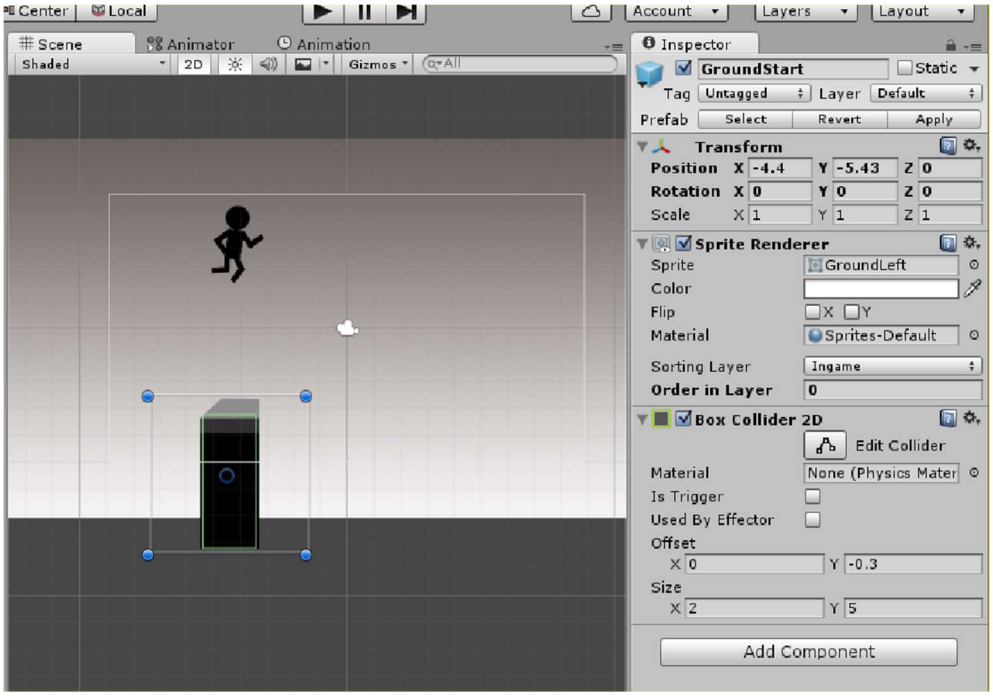


Рисунок 4.9 - Базові параметри наземного префабу

Крім того, потрібно створити скрипти для реалізації процесу генерації землі. У скрипті було створено декілька кодів для пошуку префабів у папці Resources в розділі Project. Після цього, відбувається інстанціювання префабів ґрунту та збір їх у шарі GroundsLayer. Крім того, я створюю ігровий об'єкт під назвою GroundStart, який є таким же, як і префаб GroundLeft, так що всі ґрунти можуть бути інстанційовані за GourndStart, щоб зробити повну дорогу. Рисунок 4.9 показує цей процес.

На цьому етапі з наземних активів можна зробити повноцінну дорогу. Кожен з наземних об'єктів з 2D колайдером може дозволити персонажу бігти по ньому. У всіх нескінченних іграх, що бігають, ігрова сцена часто рухається в горизонтальному або вертикальному напрямку. З рухом сцени рівень землі буде часто змінюватися. У цій грі земля буде змінювати висоту і ширину випадковим чином. Також випадковим чином змінюється ширина проміжку між майданчиками.

```
for (int i = 0; i < 20; i++)
{
    GameObject temg1 = Instantiate (Resources.Load ("GroundLeft", typeof(GameObject))) as GameObject;
    temg1.transform.parent = collectedGrounds.transform.FindChild ("gLeft").transform;
    GameObject temg2 = Instantiate (Resources.Load ("GroundMid", typeof(GameObject))) as GameObject;
    temg2.transform.parent = collectedGrounds.transform.FindChild ("gMiddle").transform;
    GameObject temg3 = Instantiate (Resources.Load ("GroundRight", typeof(GameObject))) as GameObject;
    temg3.transform.parent = collectedGrounds.transform.FindChild ("gRight").transform;
    GameObject temg4 = Instantiate (Resources.Load ("GroundBlank", typeof(GameObject))) as GameObject;
    temg4.transform.parent = collectedGrounds.transform.FindChild ("gBlank").transform;
}

collectedGrounds.transform.position = new Vector2(-60.0f,-20.0f);

GroundPosition = GameObject.Find("GroundStart");
startUpPositionY = GroundPosition.transform.position.y;
```

Рисунок 4.10 - Скрипт генерації префабів з папки Resources.

По-перше, поля і фони повинні бути в русі. Оскільки гра у двовимірній перспективі, нам потрібно лише налаштувати осі X та Y. Крім того, гра прокручується в горизонтальному русі, і тоді нам потрібно лише налаштувати положення трансформування осей X. Крім того, підстави та фони редагуються з різною швидкістю гри в реальному часі. Швидкість гри фону набагато повільніша, ніж землі, що робить гру більш реалістичною. Крім того, швидкість гри за замовчуванням встановлюється вручну, як показано на рисунку 4.11.

```
gameLayer.transform.position = new Vector2 (gameLayer.transform.position.x - gameSpeed * Time.deltaTime, 0);
bgLayer.transform.position = new Vector2 (bgLayer.transform.position.x - gameSpeed / 5 * Time.deltaTime, 0);
```

Рисунок 4.11 - Скрипт руху землі та фону.

Після завершення переміщення ділянок землі. Нам потрібно відредагувати рівень висоти ділянок. У процесі гри ширина і висота ділянки генеруються випадковим чином, а проміжок між ділянками неоднаковий. Крім того, випадковий діапазон ділянок та зазор повинні бути розроблені в межах діапазону, оскільки надмірна ширина ділянок та зазор можуть вплинути на ігровий баланс. Занадто великий простір зробить гру нудною, і гру неможливо продовжувати. Тому ми встановлюємо діапазон поля та проміжку, як наведено на рисунку 4.12.

Уся ситуація на ділянках показана на рисунку 4.12, так що якщо порожня ділянка або проміжна ділянка більше 0, то проміжна ділянка зменшує свої

підрахунки. Крім того, якщо остання ділянка є порожньою, сцена повинна згенерувати нову ділянку, або новий початок. Проміжне поле буде згенеровано в процесі гри, і кількість проміжних полів має бути у випадковому діапазоні, щоб гра не виглядала такою нудною. Більше того, якщо останнє поле виявиться правильним, це означає, що наступним згенерованим полем буде порожнє поле. Потім на сцені потрібно задати кількість порожніх полів у випадковому діапазоні.

Ще одним важливим елементом гри є головний герой. Гравці будуть керувати ігровим персонажем, щоб досягти більш високого рівня взаємодії протягом ігрового часу. У даному проекті персонажу потрібно додати набір анімації, щоб він рухався в просторі.

Для початку нам потрібно створити і помістити персонажа в редактор під назвою "Анімація". Редактор дозволяє дизайнеру редагувати анімацію персонажа по кадрам. Під час ігрового процесу біжуча анімація персонажа складається з трьох зображень. Крім того, для зв'язної анімації потрібно, щоб перший і останній кадр були абсолютно однаковими, щоб анімаційний цикл виглядав більш органічно. Як на рисунку 4.13.

Крім того, персонажу потрібно додати деякі компоненти, щоб він міг контактувати з землею.

```
private void generateGround()
{
    if (blankCounter > 0)
    {
        setGround("blank");
        blankCounter--;
        return;
    }
    if (middleCounter > 0)
    {
        setGround("middle");
        middleCounter--;
        return;
    }
    if (lastGround == "blank") {
        changeHeight ();
        setGround ("left");
        middleCounter = (int)Random.Range (2, 10);

    }
    else if (lastGround == "right")
    {
        blankCounter = (int)Random.Range (2, 4);

    }
    else if (lastGround == "middle")
    {
        setGround ("right");
    }
}
```

Рисунок 4.12 - Генерація полів та їх довільне встановлення

Перше, що потрібно зробити - це додати Box collider 2D, щоб персонаж міг мати зіткнення з ґрунтом.

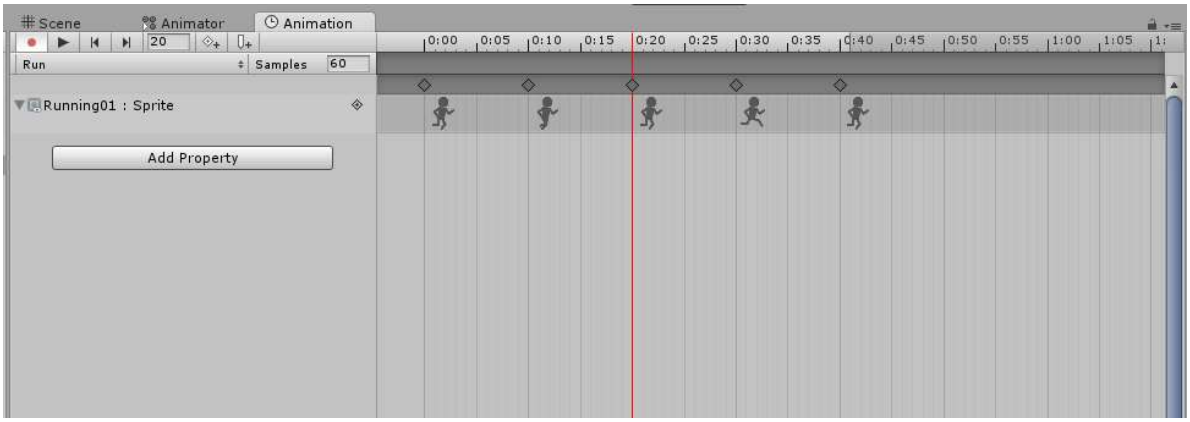


Рисунок 4.13 - Анімація бігу по кадрам

З іншого боку, оскільки розмір колайдера за замовчуванням занадто великий, і це може вплинути на ігровий процес, розмір персонажа потрібно зменшити, щоб він вписувався в гру. Крім того, важливо надати персонажу ще один компонент, який називається Rigid body 2D. Цей компонент може надати персонажу фізичний ефект, такий як маса, гравітація і так далі. При цьому обертання осі Z персонажа в Constraints слід заморозити, щоб персонаж не трясло в ігровому процесі. Це налаштування може запобігти тому, щоб персонаж не обертався під час гри, як на рисунку 4.14.

Також в цій грі персонаж може стрибати тільки один раз за одне натискання, так що в скрипті є дві ситуації, кнопка миші вгору і вниз, як показано на рисунку 4.15.

Після того, як з анімацією та властивостями персонажа закінчено, нам потрібно додати управління персонажу за допомогою деяких скриптів. Основним управлінням у цій грі є натискання правої кнопки миші, щоб персонаж перестрибнув через прірву на інший майданчик.

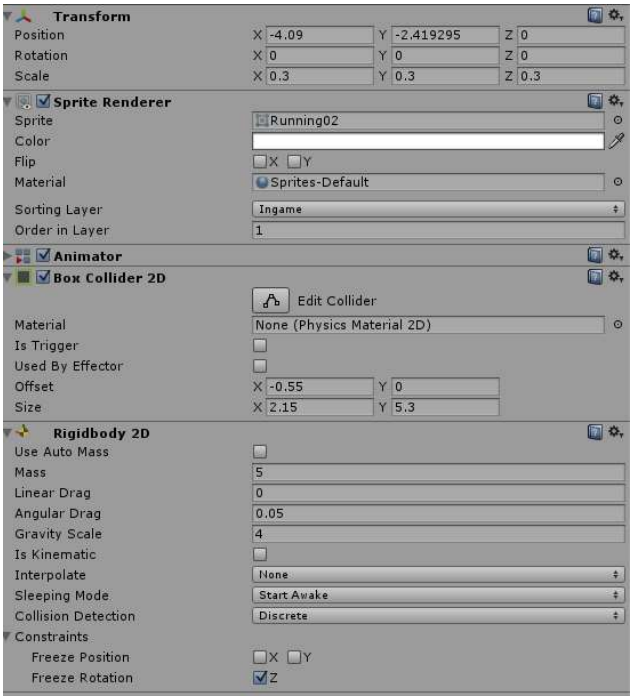


Рисунок 4.14 - Властивості персонажа

Оскільки персонаж буде бігати і стрибати в процесі гри, то анімація повинна перемикатися між цими двома діями. Тому нам потрібно задати параметр персонажу в Аніматорі, як на рисунку 4.16. Спочатку в редакторі анімації для персонажа потрібно створити анімацію стрибка. Потім анімації бігу та стрибка потрібно з'єднати між собою в Аніматорі за допомогою компонентів переходів. Після цього всім переходам потрібно додати параметр

в налаштуваннях умов під назвою AnimateState, щоб при виконанні персонажем дії і досягненні умови анімація перемикалася на іншу.

```
// Update is called once per frame
void Update () {
    if (Input.GetMouseButtonDown (0))
    {
        handleInteraction (true);
    }
    if (Input.GetMouseButtonUp (0))
    {
        handleInteraction (false);
    }

}

void handleInteraction(bool go)
{
    if (go) {
        player.jump ();
    } else {
        player.jumpHit = false;
    }
}
```

Рисунок 4.15 – Скрипт щодо базових налаштувань управління

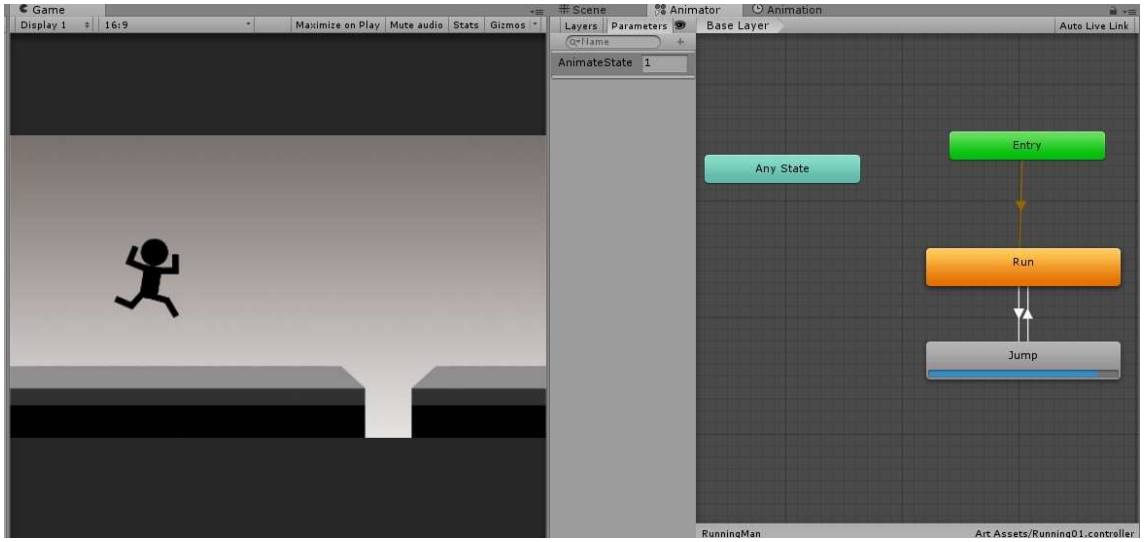


Рисунок 4.16 – Анімація персонажу за допомогою Аніматора

З іншого боку, ми також повинні більш детально описати умови за допомогою скриптів, щоб персонажу можна було дозволити перемикати анімацію в грі. У кодах анімації поділяються на два судження. Якщо вісь Y

персонажа перевищує встановлене значення, то персонаж визначається як такий, що вийшов за межі поля. Тому в умовах для відтворення анімації стрибка параметр встановлюється рівним 1. Потім, якщо вісь Y персонажа дорівнює заданому значенню, параметр встановлюється в 0 для відтворення анімації бігу, як наведено на рисунку 4.17.

```
// Use this for initialization
void Start () {
    TheAnimator = GetComponent<Animator> ();
}

// Update is called once per frame
void Update () {
    if(!OverGrounds && theRG.velocity.y > 0.05f)
    {
        TheAnimator.SetInteger ("AnimateState", 1);
        OverGrounds = true;
    }
    else if(OverGrounds && theRG.velocity.y == 0.00f)
    {
        TheAnimator.SetInteger ("AnimateState", 0);
        OverGrounds = false;
        if (jumpHit)
            jump ();
    }

}
```

Рисунок 4.17 - Перемикання між анімаціями за допомогою скриптів

Також персонаж буде контактувати з ворогами в рамках гри. Як і в більшості ігор, при контакті персонажа з монстром, у персонажа буде спрацьовувати анімація смерті. У цьому проекті ця анімація буде реалізована за допомогою кодів, показаних на рисунку 4.18.

```
void OnTriggerEnter2D(Collider2D coll)
{
    if (coll.gameObject.tag == "RunningMan")
    {
        GameObject tmpPlayer = GameObject.Find ("RunningMan");

        tmpPlayer.GetComponent<Rigidbody2D> ().AddForce (Vector2.up * 2000);
        tmpPlayer.GetComponent<Collider2D> ().enabled = false;
```

Рисунок 4.18 - Скрипт для анімації смерті

Коли персонаж мертвий, персонаж буде підстрибувати вгору зі значенням величини зусилля. Також персонаж буде провалюватися крізь землю, як на

рисунку 4.19. У коді персонаж вимикає свою колайдерну складову, щоб мати можливість проходити крізь колайдери землі.

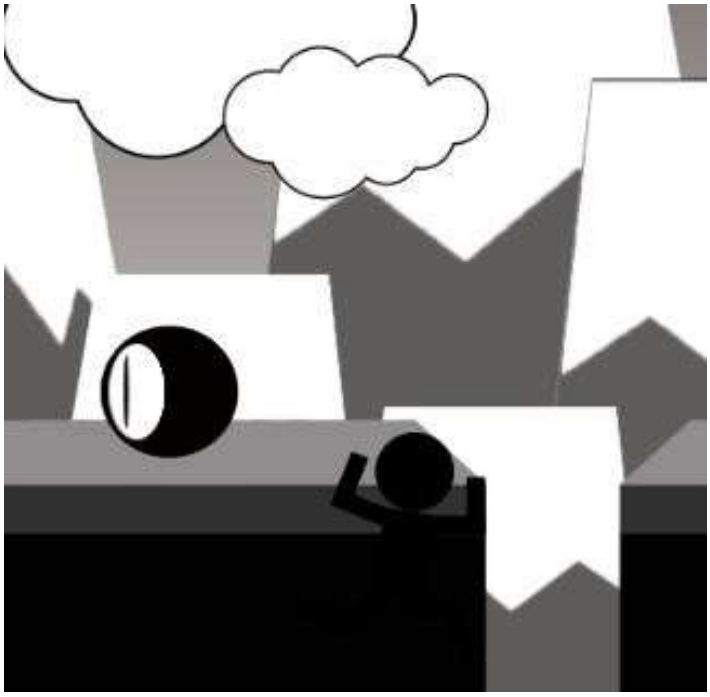


Рисунок 4.19 - Сцена смерті персонажа

У грі персонаж не просто перестрибує через розрив, щоб потрапити на інші майданчики. Є ще й вороги. Під час ігрового процесу персонажу також потрібно уникати монстрів на дорозі. Монстр має такі основні компоненти, як Renderer і Col- lider2D. Крім того, Collider 2d встановлений як тригер, щоб у монстра відбувалися тригерні події з персонажем. Код тригерної події знаходиться в коді зображеному на рисунку 4.18. Коли персонаж доторкнеться до монстра, тригер буде активований.

З іншого боку, монстри повинні динамічно генеруватися в грі. Монстри повинні знаходитися на майданчиках і рухатися разом з майданчиками. Крім того, на майданчику буде генеруватися лише один монстр, щоб гра не була надто складною для гравців, як на рисунку 4.20.

Положення монстра на майданчику генерується випадковим чином за допомогою кодів. Більше того, монстри завжди генеруються на середніх майданчиках, щоб захистити ігровий баланс. Адже якщо монстри генеруються

на першому або останньому полігоні, то персонаж не зможе пройти крізь монстра, щоб дістатися на інший бік.

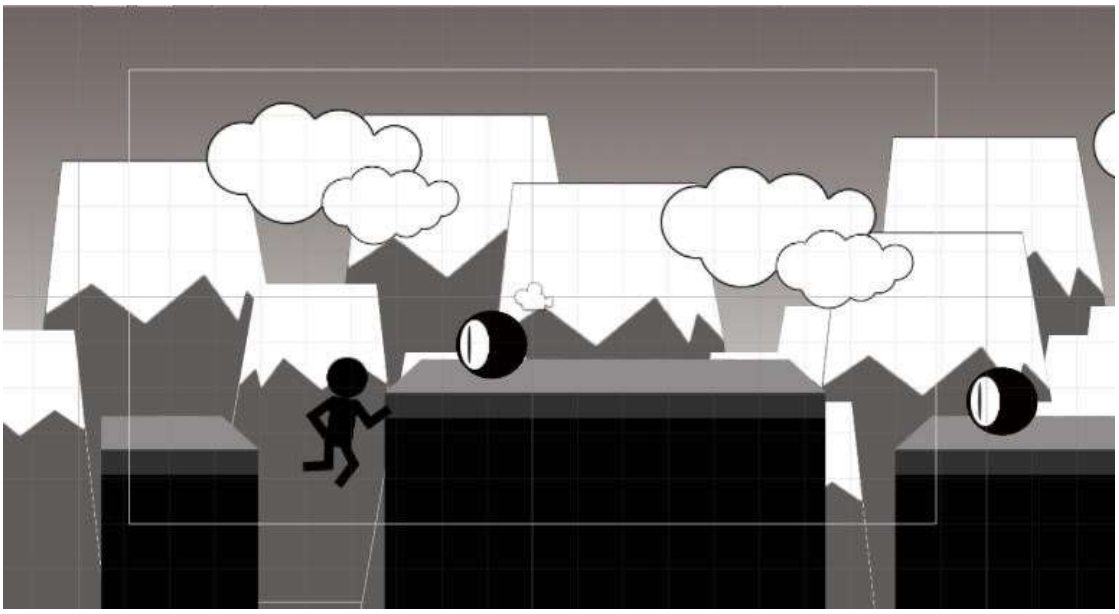


Рисунок 4.20 - Завжди один монстр на платформі (біле поле)

Ще одним дуже важливим ігровим елементом є монети винагороди. Персонаж не просто заробляє очки, просуваючись вперед, щоб досягти інших майданчиків. Також, монета винагороди може дати дуже хороші результати у підвищенні інтересу до гри. Нагородні монети мають деякі основні принципи в порівнянні з монстром. Але монета зникає, коли персонаж її зібрав, а потім з'являється з правого боку. Ці функції реалізовані за допомогою скриптів зображених на рисунку 4.21.

Отже, гравцям потрібно керувати персонажем, щоб отримати якомога більше монет. У той же час їм також потрібно ухилятися від монстрів, щоб уникнути смерті. Всі ці ігрові об'єкти можуть зробити гру більш гнучкою та цікавою.

У цій грі найвищий результат полягає у вимірюванні відстані, яку гравець досягнув на даний момент. Крім того, монети дають бонусні бали. Коли гравець досягає нового вищого результату, база даних рекордів замінює старий результат на новий, як на рисунку 4.22.

```
    if (!inPlay && !genCoins)
        respawn ();
}

void OnTriggerEnter2D(Collider2D coll)
{
    if (coll.gameObject.tag == "RunningMan")
    {
        GameObject.Find ("Main Camera").GetComponent<ScoreHandler> ().Points += 10;
    }

    inPlay = false;
    this.transform.position = new Vector2 (this.transform.position.x, this.transform.
}

void respawn()
{
    genCoins = true;
    Invoke ("placeCoins", (float)Random.Range (3, 10));
}
```

Рисунок 4.21 - Генерування монет та функція винагороди

```
public void sendToHighScore()
{
    if (TheScore > TheBest)
    {
        saveValue (TheScore);
    }
}
```

Рисунок 4.22 - Заміна старого результату на новий

У цій грі необхідно використати графічний інтерфейс для відображення рахунку та найвищого результату на ігровій формі. Графічний інтерфейс реалізується за допомогою кодів і основний інтерфейс також повинен бути відредагований у скрипті. На рисунку 4.24 є дві частини рахунку, включаючи поточний рахунок та максимальний рахунок. Поточний рахунок призначений для відстеження поточного результату, якого гравець досягнув у процесі гри. Максимальний рахунок - для відображення найвищого результату на даний момент.

```
void OnGUI()
{
    GUI.color = Color.black;
    GUIStyle Gstyle = GUI.skin.GetStyle ("Lable");
    Gstyle.alignment = TextAnchor.UpperLeft;
    Gstyle.fontSize = 20;
    GUI.Label (new Rect (20, 20, 200, 200), "Current: " + TheScore.ToString (), Gstyle);
    Gstyle.alignment = TextAnchor.UpperRight;
    GUI.Label (new Rect (Screen.width - 220, 20, 200, 200), "Highscore: " + TheBest.ToString (), Gstyle);

}
```

Рисунок 4.23 - Налаштування графічного інтерфейсу

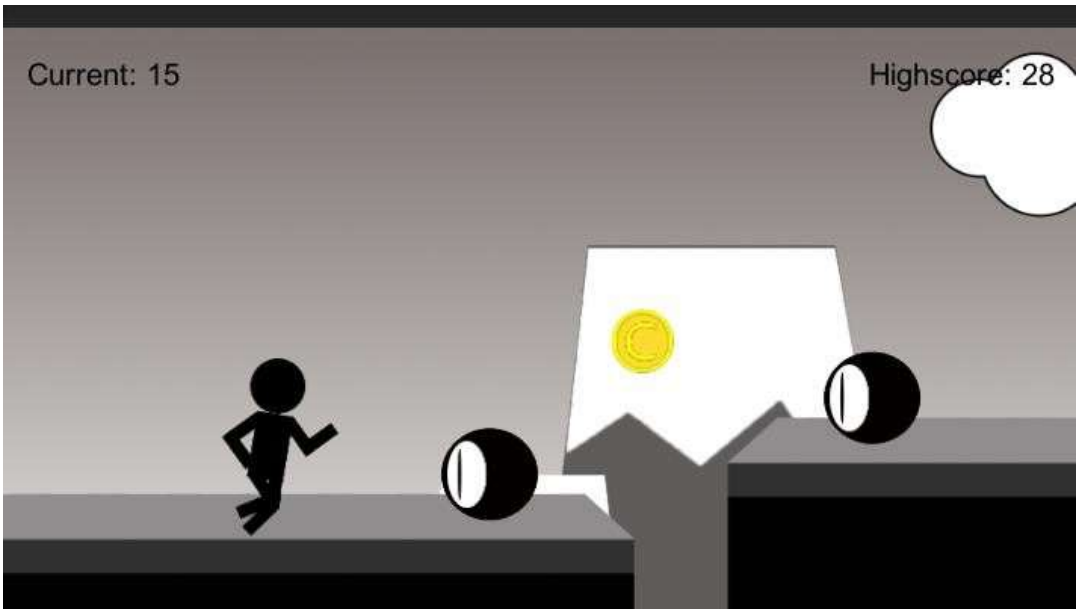


Рисунок 4.24 - Графічний інтерфейс поточного рахунку та рекорду

При запуску гри поточний рахунок почне відраховувати бали, які набрав персонаж. Також, в рекорді рекордів буде зберігатися попередній рекорд, поки його не замінить новий. Виглядає це так, як показано на рисунку 4.24.

Коли всі скрипти та внутрішньоігрові об'єкти готові до тестування, нам потрібно лише завершити, натиснувши кнопку Build & Settings в інструментах File, як на рисунку 4.25. Оскільки Unity 3D є крос-платформним рушієм, він підтримує майже всі ігрові платформи на ігровому ринку.

На рисунку 4.25 показані ігрові сцени у вікні Scenes in Build, яке включає в себе меню запуску гри та основну ігрову сцену.

Крім того, було вирішено запустити цю гру на платформі ПК, оскільки на іншій платформі можуть знадобитися деякі додаткові налаштування

Коли всі налаштування готові до запуску, достатньо натиснути кнопку Build або Build and Run, щоб створити exe-файл і запустити його.

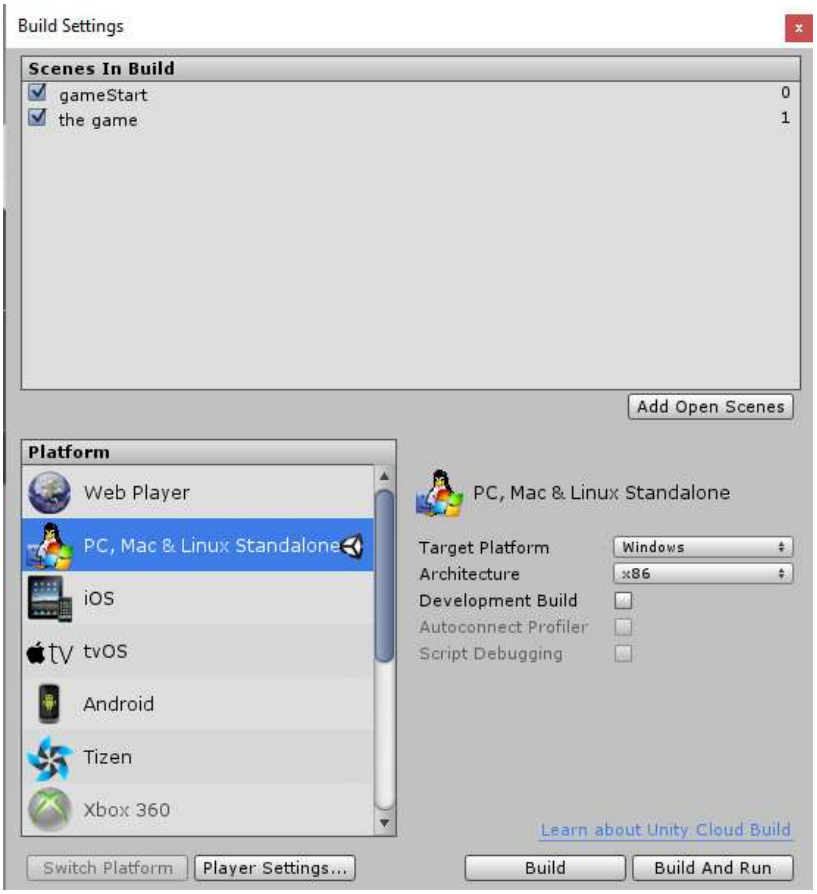


Рисунок 4.25 - Інструмент Build & Settings в Unity3D

ВИСНОВКИ

Була проведена розробка алгоритму автоматичного формування харчового раціону людини.

Проаналізовано поточну ситуацію на ринку та виявлено потребу в створенні системи, яка допоможе автоматизувати підбір харчового раціону на основі розпізнавання інгредієнтів, які містяться в зображенні страви. Актуальність тематики підтверджується величезним попитом на схожі системи таких масштабів на ринку. Для проєктування системи проаналізовано різноманітні алгоритми на методи формування харчового раціону. Здійснено порівняння ефективності від їх упровадження в різних сферах. Вибрано доцільний метод та встановлено оптимальний початковий набір даних.

Проаналізовано існуючі публікації, системи та програмні засоби, які пов’язані із раціональним харчуванням, виявлені їх переваги, недоліки та загальне призначення. Обґрунтовано наукову новизну і пратичну значущість власної розробки.

Виявлено особливості розробки додатків для операційної системи Windows та здійснено вибір інструментальних засобів розробки-додатку. Для розробки було обрано середовище Unity 3D, яке орієнтоване на розробку додатків для усіх популярних операційних систем, має простий і зрозумілий інтерфейс та інструментальні засоби для упаковки і маркування коду, багато помічників та шаблонів для загальних елементів програмування. У якості мови програмування було обрано C#.

Здійснено розробку та програмну реалізацію додатку для формування харчового раціону людини який надає їм можливість доступу до

автоматичного підбору раціону на основі назначеної лікарем дієти,та до інформації стосовно назначеного їм плану харчування.

Поставлені завдання виконано повністю, однак є ряд задач, які потребують подальшої розробки: синхронізація алгоритму з грою на основі визначеного плану харчування, налаштування вікон реєстрації та забезпечення можливості зворотного зв’язку між розробником та користувачем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ковальова К.А. Оптимізація раціону харчування людини для підтримання маси тіла з урахуванням різних типів тіла та різних типів метаболізму. / Ковальова К.А., Косніков С. Сидорко Н. // Науковий журнал КубГАУ, №105(01) – Краснодар, 2015. – 13 с.

2. Ляшенко О.А. Застосування функціонально-модульного підходу до проектування підсистеми визначення збалансованого раціону харчування / Ляшенко О.А, Шулак В. О. // ISSN 2521-6406, Комп'ютерне моделювання: аналіз, управління, оптимізація. No. 2 – 2018 р. – 34-44 с.

3. Algorithmization of constructing control models of complex systems in the language of functioning tables / Кабулов А., Норматов І., Карімов А. // European journal of molecular and clinical medicine, volume 7, issue 2. – 2020 р. – 758-770 с.

4. Kołodziejczyk J. Automatic dietary menu planning based on evolutionary algorithm. – Польша, 2012. – 9 с.

5. Береговий В. К. Основи наукової організації здорового харчування. Ефективна економіка, № 11. – 2011.
Режим доступу: http://nbuv.gov.ua/UJRN/efek_2011_11_19. – Назва з екрана.

6. Глущенко Л. Перспективи використання штучного інтелекту для розробки функціональному харчуванні // Вісник Львівського університету. Серія біологічна. Вип. 73. – 2016. – 437 с.

7. Клименко Д. О., Руденко О. А. Веб-додаток для сервісу складання раціону здорового харчування та доставки продуктів. // Системи управління, навігації та зв'язку. Вип. 2. – 2019 р. – 103–109 с. Режим доступу: https://doi.org/10.26906/SUNZ.2019.2.103. – Назва з екрана

7. Лещенко О. Б., Хлюпіна А. С., Богдан Д. О. Веб-додаток для ведення щоденника харчування та тренування: вимоги, розроблення і впровадження. // Радіоелектронні і комп'ютерні системи. № 3. – 2018. – 49-62 с.
Режим доступу: https://doi.org/10.32620/reks.2018.3.06. – Назва з екрана
8. Мостенська Т. Л., Кундєєва Г. О. Харчування як складова продовольчої безпеки. // Наукові праці Національного університету харчових технологій, Т. 22, № 3. – 2016 р. – 113–122 с.
9. Мотузка Ю. Харчова та енергетична цінність продуктів для спеціальних медичних цілей. // Товари і ринки, № 2(1). – 2017 р. – 59–66с.
10. Сластин В. В., Самусева Е. С., Москальчук Л. В. Сбалансированный рацион питания. // Проблеми харчування, № 1. – 2014 р – 33–39 с.
10. Попова О.В. Інформаційні системи в економіці: методичний
посібник для економічних спеціальностей. Частина II Access PowerPoint (2-е вид.): метод. посібник / Попова О.В., Комісарова К.А. – Краснодар, КубДАУ 2014, – 46 с.
11. Теорія ухвалення рішень: навчальний посібник/ С. М. Косников;
за ред. д-ра екон. наук, проф. А. Г. Бурда. – Краснодар, КубДАУ, 2013. – 54 с.
12. Ciborowska H. Dietetyka. żywienie zdrowego i chorego człowieka. Wydawnictwo Lekarskie PZWL, Warsaw Eckstein EF. Menu planning by computer: the random approach. / Ciborowska H., Rudnicka A. – Польша, 2006. - 529–533 с.
13.Gaál B. Application of Artificial Intelligence for Weekly Dietary Menu Planning, Studies in Computational Intelligence/ Gaál B., Vassányi I., Kozmann G. – Volume 65. – Польша, 2007. - 27-48 с.
14.Orzycki D. Vitalmax: Food diary. – Електрон. дан. (1 файл). – 2011. – 11 с. – Режим доступу: http://www.darkthorn.republika.pl. – Назва з екрана.
15.Petot G.J. An artificial intelligence system for computer- assisted menu planning / Petot G.J., Marling C., Sterling L. – J Am Diet Assoc. – США, 1998. –14-19 с.

16.Seljak B.K. Computer-based dietary menu planning, Proceedings of the 7th WSEAS International Conference on Evolutionary Computing, World Scientific and Engineering Academy and Society, –Польша, 2006. –39-44 с.

20. Фішер, Ф... Критерії проектування стратегічних ігор. – Електрон. дан. (1 файл). – 2014. – 1-14 с. Режим доступу: http://www.gamasutra.com/blogs/FabianFischer/20141201/231243/Criteria_for_Strat – Назва з екрана.

21.Френсіс К. 2014. The importance of Gameplay Balance. – Електрон. дан. (1 файл). – 2014. – 1-22 с. – Режим доступу: http://www.theoryofgaming.com/importance-gameplay-balance. – Назва з екрана.

22.Мастерс М. 2014. Unity, Source 2, Unreal Engine 4 або CryENGINE - який ігровий рушій обрати? – Електрон. дан. (1 файл). – 2016. – 8-22 с. – Режим доступу: http://blog.digitaltutors.com/unity-udk- cryengine-game-engine-choose. – Назва з екрана.

23.Хіскот Р. 2014. 10 мов програмування, які варто вивчити прямо зараз. – Електрон. дан. (1 файл). – 2014. – 17 с. – Режим доступу: http://mashable.com/2014/01/21/learn-programming-lan- guages/#gl6J3bZOskqL. – Назва з екрана.

24.Документація Unity. Unity Manual, Робота в Unity, Створення ігрового процесу, Scenes. – Електрон. дан. (1 файл). – 2016. – 1-9 с. – Режим доступу: ttp://docs.unity3d.com/Manual/CreatingScenes.html. 2012. – Назва з екрана.

25. Документація Unity. Unity Manual, Робота в Unity, Створення ігрового процесу, GameObjects. . – Електрон. дан. (1 файл). – 2016. – 1-28 с. – Режим доступу: http://docs.unity3d.com/Manual/GameObjects.html. – Назва з екрана.

26. Документація Unity. Unity Manual, Робота в Unity, Створення ігрового процесу, Transform. – Електрон. дан. (1 файл). – 2016. – 1-28 с. – Режим доступу: http://docs.unity3d.com/Manual/Transforms.html. – Назва з екрана.

27. Документація Unity. Unity Manual, Фізика, Огляд фізики, Колайдери.

– Електрон. дан. (1 файл). – 2016. – 1-32 с. – Режим доступу: http://docs.unity3d.com/Manual/CollidersOverview.html.. – Назва з екрана.
28. Документація Unity. Unity Manual, Unity Graphics, Огляд графіки, матеріали, шейдери та текстури. – Електрон. дан. (1 файл). – 2016. – 1-17 с. – Режим доступу: http://docs.unity3d.com/Manual/Shaders.html. – Назва з екрана.
29. Документація Unity. Unity Manual, Animation, Animation Reference, Animation Controller, Animation States. – Електрон. дан. (1 файл). – 2016. – 1-12 с. – Режим доступу: http://docs.unity3d.com/Manual/class-State.html. – Назва з екрана.
30. Пірсон Ч. Вивчаючи NGUI для єдності. – Бірмінгем 2015. – 25-26 с.

ДОДАТОК А - ВІДПОВІДНОСТІ МІЖ КАЛОРІЙНІСТЮ СТРАВ ТА ЇХ ДІЄТАМИ

Дієти					
Д 1		Д 2		Д3	
страва	Калорійність	страва	Калорійність	страва	Калорійність
Каша рисова молочна	273,3	Каша манна молочна	250,95	Каша манна молочна	250,95
Омлет	257,8	Омлет	257,8	Яйце некруто	157
Молоко	104	Молоко	104	Молоко	104
Молоко	104	Суп рисовий молочний	381,13	Молоко	104
Суп вівсяний молочний	321,63	Компот	96,65	Суп вівсяний молочний слизовий	321,63
Суфле із пареного м'яса (телят)	283,15	Суфле із судака з маслом	391,45	Суфле із вареного м'яса	283,15
Компот із сухофруктів	96,65	Молоко	104	Кисель вишневий	94,69
Яйце некруто	157	Каша вівсяна молочна	253,3	Яйце некруто	157
Каша гречана молочна	166,8	Молоко	104	Каша рисова молочна слизова	273,3
Яйце некруто	157	Яйце некруто	157	Молоко	104
Молоко	104			Молоко	104
Разом	2233,3		2204,28		2057,72

ДОДАТОК Б - ВІДПОВІДНОСТІ МІЖ КАЛОРІЙНІСТЮ СТРАВ ТА ЇХ ДІЄТАМИ

Дієти					
Д4		Д5		Д6	
страва	Калорій ність	страва	Калорій ність	страва	Калорій ність
Каша вівсяна молочна	253,3	Каша манна молочна	250,95	Каша рисова молочна	273,3
Молоко	104	Яйце некруто	157	Молоко	104
Суп манний молочний	376,03	Суфле із судака з маслом парова	391,45	Суп вівсяний молочний	321,63
Суфле сирне з вишневою паливною підливою	410,295	Суп вівсяний молочний	321,63	Суфле із вареного м'яса	283,15
Яйце некруто	157	Компот	96,65	Молоко	104
Яйце некруто	157	Молоко	104	Яйце некруто	157
Каша гречана молочна слизова	166,8	Яйце некруто	157	Компот	96,65
Молоко	104	Каша гречана молочна	166,8	Яйце некруто	157
		Молоко	104	Молоко	104
Разом	2137,075				1975,53

ДОДАТОК В - ПРИКЛАДНЕ МЕНЮ НА ТИЖДЕНЬ (ДІЄТА № 1А)

ПОНЕДІЛОК:

Перший сніданок. Каша рисова молочна слизова (рис - 50, молоко - 50, вода - 100, масло вершкове - 10); омлет (яйце - 1 шт., молоко - 50, сметана - 10, масло вершкове - 10); молоко - 200.

Другий сніданок. Молоко – 200.

Обід. Суп вівсяний молочний слизовий («Геркулес» - 40, молоко - 150, яйце - 1/4 шт., Масло вершкове - 10, цукор - 2, вода - 350); суфле з пареного м'яса (м'ясо — 90, яйце — 1/2 шт., молоко — 50, вершкове масло — 10, борошно — 5); компот із сухофруктів (сухофрукти - 20, цукор - 15, вода - 200).

Полудень. Молоко — 200, яйце некруто.

Вечеря. Каша гречана молочна слизова (крупа гречана - 50, молоко - 50, вода - 100, масло вершкове - 10); яйце некруто, молоко - 200.

На ніч. Молоко – 200.

Вівторок:

Перший сніданок. Каша манна молочна (крупа манна - 40, молоко - 50, цукор - 5, масло вершкове - 10); омлет (яйце - 1 шт., молоко - 60, сметана - 10, масло вершкове - 2); молоко - 200.

Обід. Суп рисовий молочний слизовий (рис - 30, молоко - 300, яйце - '/4 шт., Масло вершкове - 10, цукор - 2, вода - 200); суфле з судака з олією парове (риба - 150, масло вершкове - 20, борошно пшеничне - 5, яйце - 1/2 шт., молоко - 40); компот (сухофрукти – 20, цукор – 15, вода – 200).

Полудень. Молоко — 200, яйце некруто.

Вечеря. Каша вівсяна молочна слизова («Геркулес» - 50, молоко - 50, вода - 100, масло вершкове - 10); яйце некруто; молоко - 200.

На ніч. Молоко – 200.

СЕРЕДА:

Перший сніданок. Каша манна молочна; яйце некруто; молоко - 200.

Другий сніданок. Молоко – 200.

Обід. Суп вівсяний молочний слизовий; суфле із вареного м'яса; кисіль вишневий (вишня – 30, цукор – 20, крохмаль – 10, вода – 180).

Полудень. Молоко — 200, яйце некруто.

Вечеря. Каша рисова молочна слизова, молоко - 200.

На ніч. Молоко – 200.

ЧЕТВЕР:

Перший сніданок. Каша вівсяна молочна слизова; молоко - 200.

Другий сніданок. Молоко – 200.

Обід. Суп манний молочний (крупа манна – 30, молоко – 300, яйце – 1/4ШТ., масло вершкове – 10, цукор – 2, вода – 200); суфле сирне з вишневою підливою парове (сир - 120, крупа манна - 10, масло вершкове - 10, цукор - 15, яйце - 1/2 шт., Вишня суха - 25, крохмаль - 5); компот.

Полудень. Молоко - 200; яйце некруто.

Вечеря. Каша гречана молочна слизова; яйце некруто; молоко - 200.

На ніч. Молоко – 200.

П'ЯТНИЦЯ:

Перший сніданок. Каша манна молочна, яйце некруто; молоко - 200.

Другий сніданок. Молоко – 200.

Обід. Суп вівсяний молочний слизовий; суфле із судака з маслом парова; компот.

Полудень. Молоко - 200; яйце некруто.

Вечеря. Каша гречана молочна слизова; молоко - 200.

На ніч. Молоко – 200.

СУБОТА:

Перший сніданок. Каша рисова молочна слизова; молоко - 200.

Другий сніданок. Молоко – 200.

Обід. Суп вівсяний молочний слизовий; суфле із вареного м'яса; компот.

Полудень. Молоко - 200; яйце некруто.

Вечеря. Каша гречана молочна слизова; яйце некруто; молоко - 200.

На ніч. Молоко – 200.

НЕДІЛЯ:

Перший сніданок. Каша вівсяна молочна слизова; яйце некруто; молоко - 200.

Другий сніданок. Молоко – 200.

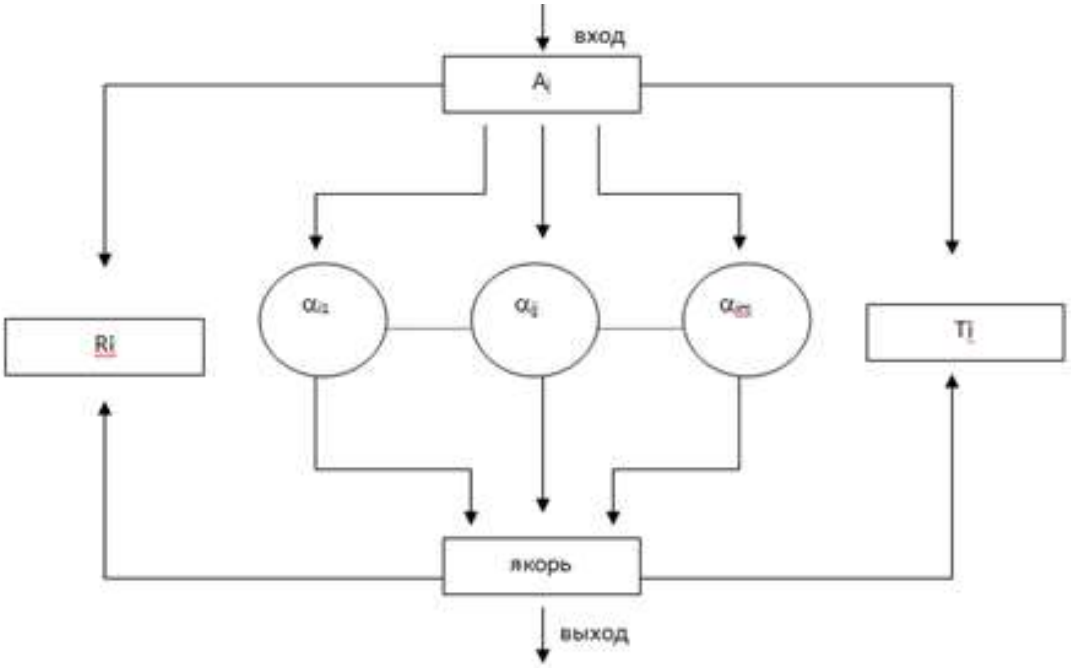
Обід. Суп рисовий молочний слизовий; суфле із судака з маслом парова; кисіль апельсиновий (апельсини – 30, цукор – 20, крохмаль – 10)

Полудень. Молоко - 200; яйце некруто.

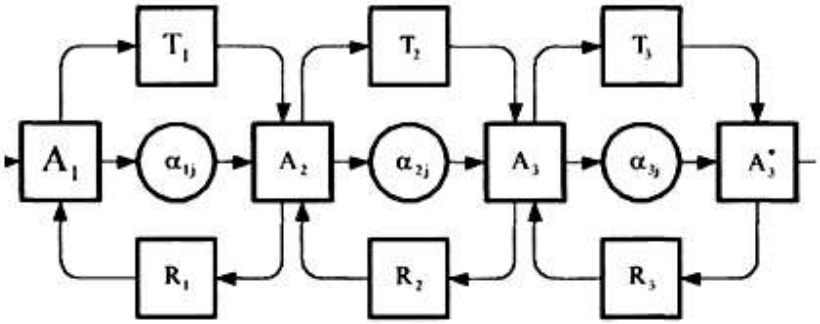
Вечеря. Каша манна молочна; омлет (яйце - 1 шт.; молоко - 60, масло вершкове - 10); молоко - 200.

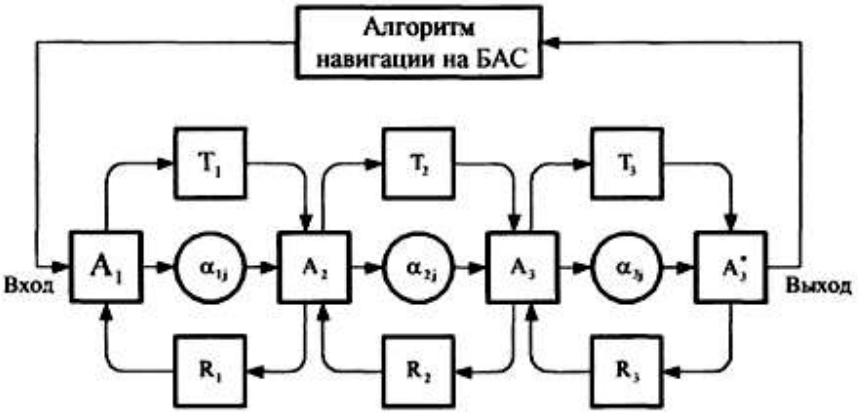
На ніч. Молоко – 200.

ДОДАТОК Г – Структурна схема блоку альтернатив



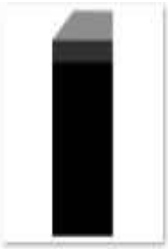
ДОДАТОК Д – Блочна мережа альтернатив вибору страв

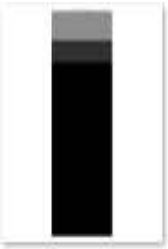


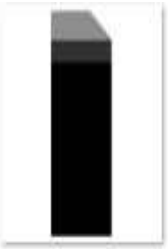


ДОДАТОК Ж – Асети до програмного продукту


Coins.png


GroundLeft.png


GroundMid.png


GroundRight.png


Jump.png


Monster.png


Mountains.png


Running01.png


Running02.png


Running04.png