Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет: Інформаційно-аналітичних технологій та менеджменту

Кафедра: Інформатика

Спеціальність: 122 Комп’ютерні науки

Освітньо-професійна програма: Інформатика

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до курсової роботи з дисципліни

“ Об’єктно орієнтоване програмування”

за темою

«розробка ігрової програми(PacMan)

із застосванням графічного движка(SFML)»

(тема роботи)

Виконав: Перевірив:

студент гр.\_\_ІТІНФ-20-1 доц. В.А. Любченко (шифр групи) (посада, ініціали, прізвище)

Самченко\_С.\_О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ініціали, прізвище) (підпис)

Харків 2021

Харківський національний університет радіоелектроніки

Кафедра Інформатика

(повна назва)

Дисципліна Об’єктно орієнтоване програмування

(повна назва)

Спеціальність 122 Комп’ютерні науки

Освітньо-професійна програма Інформатика

(повна назва)

Курс 1 Группа ІТІНФ-20-1 Семестр 2 .

ЗАВДАННЯ

на курсовий проект (роботу) студента

Самченко Станіслав Олександрович

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) розробка ігрової програми із застосванням графічного движка

2. Строк здачі студентом закінченого проекту (роботи) 24.06.2021 .

3. Вихідні данні проекту (роботи) *1. Відомості про предметну область*

*2 . Перелік використовуваних програмних засобів:*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

*1. Огляд основних методів.*

*2. Алгоритми*

*3. Комп’ютерна модель.*

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових   
креслень)

6. Дата видачі завдання 06.02.2021.

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Найменування етапів курсового проекту | Термін виконання етапів проекту | Примітки |
| 1 | Пошук необхідної літератури | 16.02.21 – 18.03.21 | Виконано |
| 2 | Пошук необхідного програмного забезпечення | 18.03.21 | Виконано |
| 3 | Ознайомлення з можливостями візуального програмування | 19.03.21 – 04.04.21 | Виконано |
| 4 | Розроблення алгоритму для вирішення поставленої задачі | 12.04.21 – 09.05.21 | Виконано |
| 5 | Створення комп’ютерної моделі по розробленому алгоритму | 12.04.21 – 09.06.21 | Виконано |
| 6 | Оформлення пояснювальної записки | 12.06.21 – 22.06.21 | Виконано |
| 7 | Захист курсового проекту | 24.06.12 |  |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Керівник роботи\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (підпис)

**ЗМІСТ**

Введення 5

1 Опис Комп'ютерної гри «PacMan» 7

2 Постановка задачі 8

3 Обґрунтування вибору середовища програмної реалізації 11

4 Реалізація 12

4.1 Алгоритми гри 15

4.2 Опис класів та їх значення у програмі 18

4.3 Програмна реалізація 26

5 Інструкція користувача 27

Висновки 28

Перелік посилань 29

Додаток А 30

# введення

Метою виконання цієї курсової роботи є створення програмного продукту, призначеного для розваги користувачів, вдосконаленням їхніх реакції. В рамках даної курсової роботи реалізується клон всім добре відома гри "Pacman" за допомогою мови програмування С++, середи розробки Microsoft Visual Studio і мультимедійної бібліотеки SFML. Таким чином, завданням в даному випадку є розробка і реалізація алгоритму даної гри.

Комп'ютерна гра — комп'ютерна програма, що служить для організації ігрового процесу (геймплея), зв'язку з партнерами по грі, або сама виступає в якості партнера.

Основними компонентами комп'ютерних ігор є такі складові частини: сіттинг і геймплей.

Сеттінг — це середовище, в якому відбувається дія комп'ютерної гри; місце, час і умови дії.

Геймплей — компонент гри, який відповідає за інтерактивне взаємодія гри і гравця. Геймплей описує, як гравець взаємодіє з ігровим світом, як ігровий світ реагує на дії гравця і як визначається набір дій, який пропонує гравцеві гра.

Системи штучного інтелекту є популярною, що швидко розвивалася облат інформатики. Такі системи знайшли своє застосування в промисловому виробництві, військовій техніці, персональних пристроях, а також в іграх. Таким чином, проблема дослідження ефективності, а

також практичного застосування різних алгоритмів є актуальною.

Існує безліч алгоритмів штучного інтелекту для ігор з повною інформацією, таких як шахи, шашки, хрестики-нулики і т.д. Найбільш відомим з них є алгоритм минимакс, який, при деякій оптимізації, показує хороші резултат в наведених варіантах настільних ігор. Однак, ефективність минимакса в інших типах ігор, наприклад, іграх з лабіринтом мало вивчена. Ще одним відомим прикладом є алгоритми машинного навчання. Вони знайшли широке застосування в сфері аналізу даних, але досить рідко використовуються в іграх.

Гра Pac-Man є відомою грою в жанрі аркади, що вийшла в 1980 році в Японії. Ця гра є яскравим представником ігор класу лабіринт з повною інформацією. Незважаючи на гадану простоту, Pacman є хорошим середовищем для реалізації і тестування алгоритмів штучного інтелекту, зважаючи на складну структури лабіринту, наявності очок, а також аппонент, яких представляють примари.

## 1 опис комп`ютерної гри «PacMan»

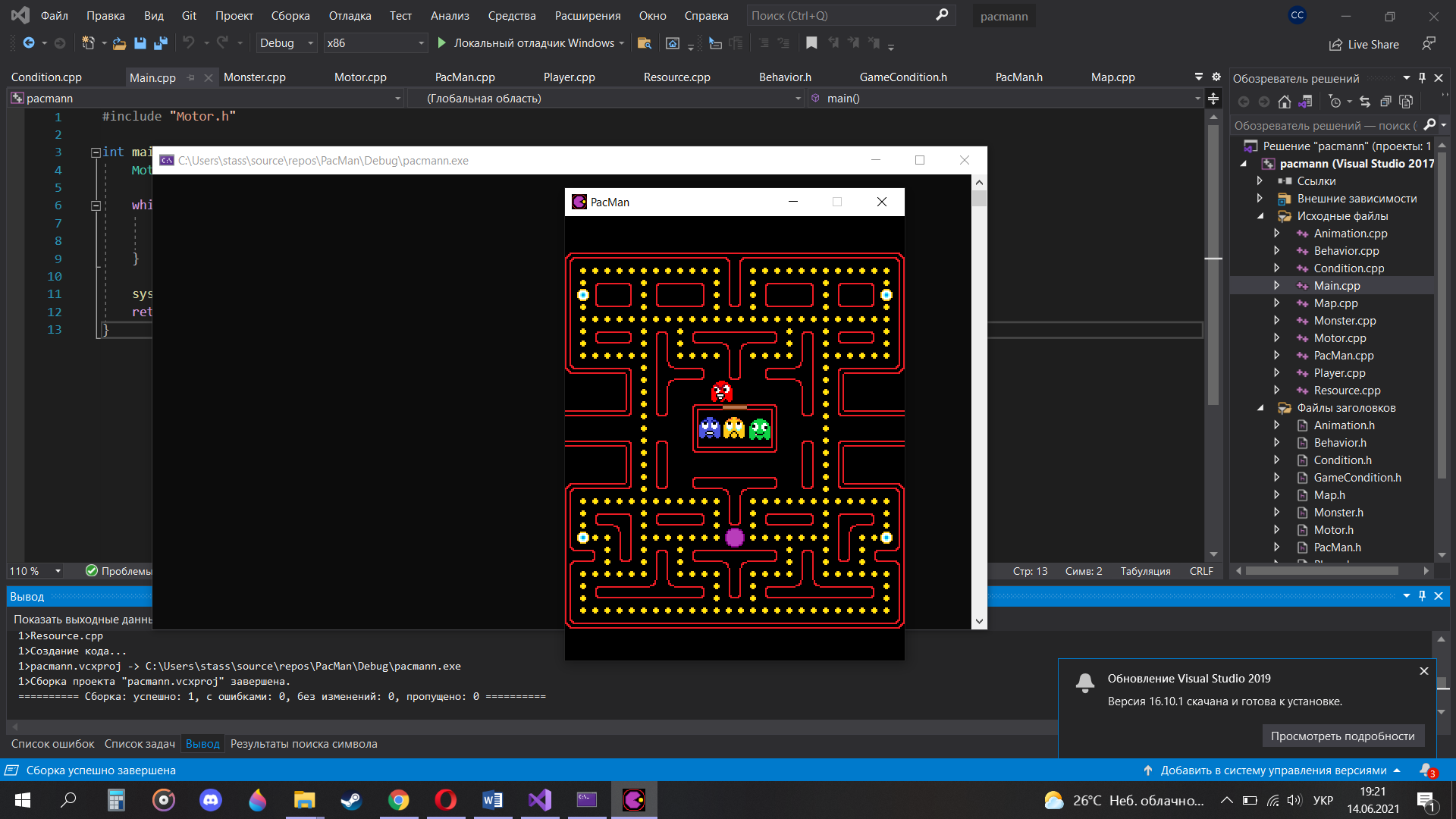


Рисунок. 1 – зовнішній вигляд ігрової програми

Комп'ютерна гра «PacMan» — це гра, в якій гравець грає за круглу фіолетову істоту з ротом. Основна задача — керуючи Пакменом зібрати всі жовті крапки на рівні, уникаючи зіткнень з монстрами, яких присутньо чотири штуки на карті. Гра закінчується, коли з'їдені всі крапки. У випадку, коли Пакмен пересікається з монстром — він(Пакмен) помирає і починає гру зі своєї точки відрождення. Монстри на початку рівня знаходяться в центрі екрану в невеликій кімнаті, звідки вони виходять по одному, а один монстер починає рівень поза кімнатою. Кожен монстер має індивідуальні характеристики. Якщо Пакмен з'їсть велику білу крапку, то на кілька секунд монстри стають нешкідливий і вразливими, а сутичка з ними призведе до погибелі самого монстра. З'їдені в такій ситуації монстри не зникають, а повертаються до центру екрана. Сама гра відбувається у невеличкому лабіринті. Якщо зайти в коридор, який веде праворуч або ліворуч за межі екрану, то Пакмен вийде з протилежного боку (це можуть зробити і монстри).

## 2 постановка задачі

Перш за все необхідно поглянути на складові частини гри, тобто з яких об’єктів вона складається. Перше на що необхідно звернути увагу це на місце, в якому відбувається всі ігрові події, а саме на карту. Вона представляє собою певний лабіринт, в якому присутні проходи(в центрі з ліва і з права), безліч коридорів і кімната з монстрами. Із об’єктів з якими головний персонаж повинен взаємодіяти присутні такі елементи як крапочки і чотири бонуси, які Пакман повинен їсти. І, звичайно, сам Пакман та його чотири супротивники — монстри.

Тепер необхідно розібратися з функціоналом кожного з цих ігрових фрагментів, тобто з’ясувати яким чином всі ці складові гри будуть взаємодіяти один з одним. Розглянемо кожен елемент окремо і проаналізуємо для чого він потрібен.

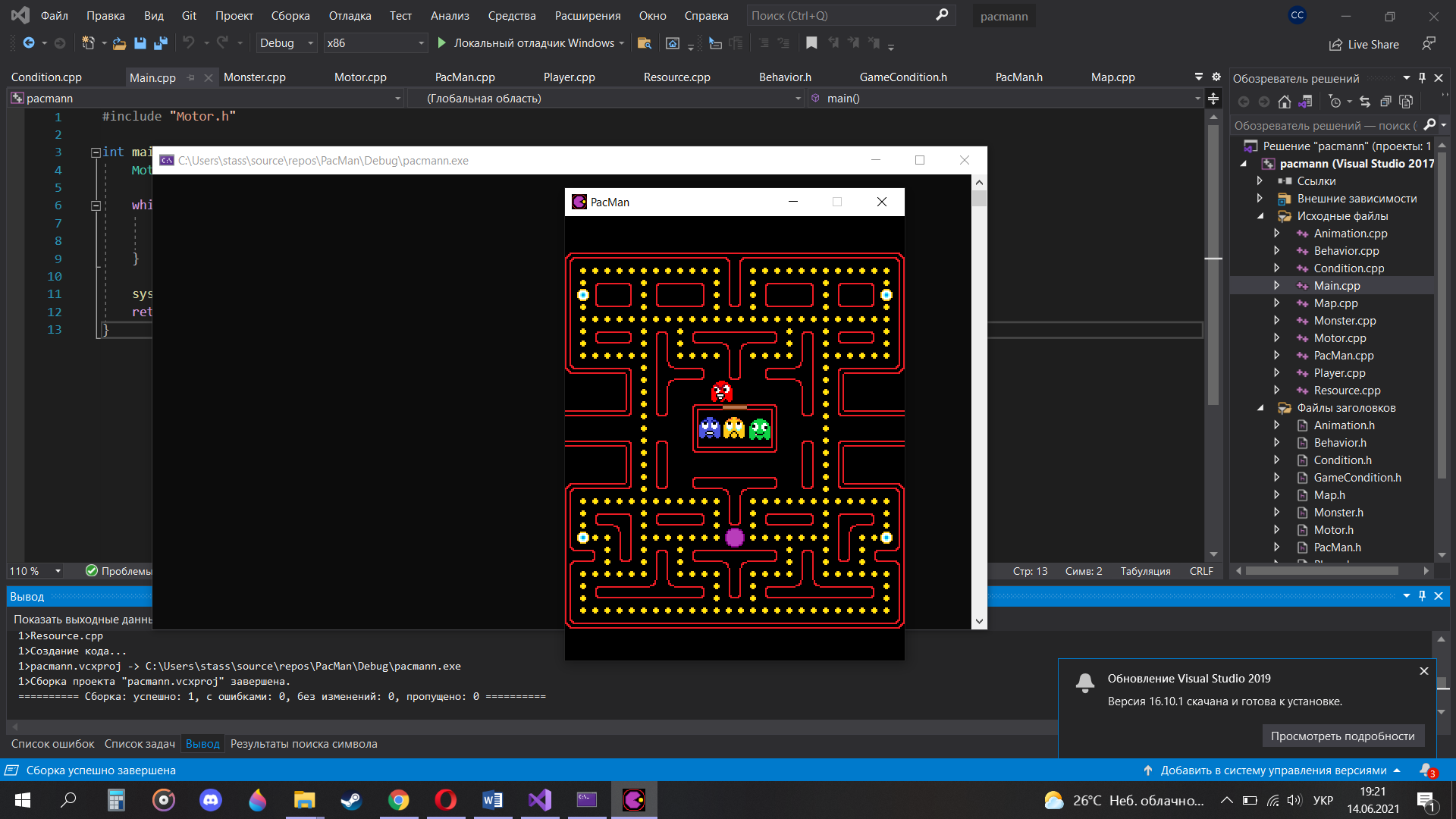
Карта: невеличкий лабіринт, в якому є коридори, по яким всі гравці повинні вільно пересуватися; є стінки, сутичка з якими повинна зупиняти гравців, і не давати їм проходити далі по їх траєкторії руху; і є тунель, який повинен проводити всіх ігрових персонажів з одного кінця в протилежний(чимось нагадує портал). Зауважимо, що на карті також присутня кімната з монстрами, в яку гравці не можуть проникнути. Монстри можуть тільки виходити з неї.

Рисунок 2. – карта

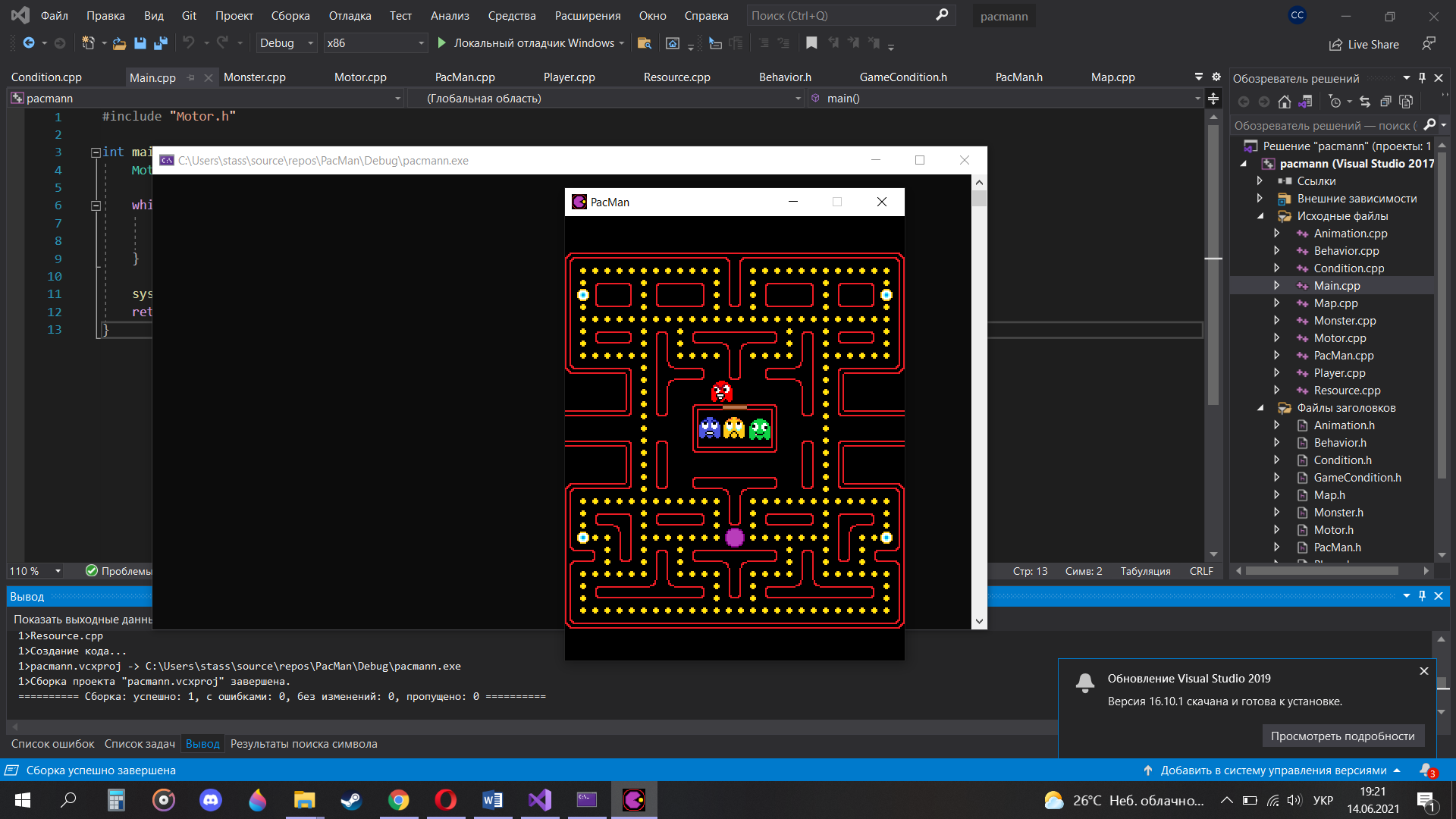
Крапочки: майже весь простір на карті, по якому гравці можуть пересуватися, заповнений цими крапочками. Головна задача Пакмана з’їсти всі ці крапочки. Тільки тоді гра вважається закінченою і може починатися спочатку.

Рисунок 3. - крапочки

Бонуси: в усіх чотирьох кутах карти присутні такі крапочки, які на фоні інших дещо більші і в цілому відрізняються від інших кольором. Це і є бонуси, поїдання яких робить монстрів вразливими на певний час. Сутичка Пакмана з монстром в такий момент призведе до «поїдання» монстра.

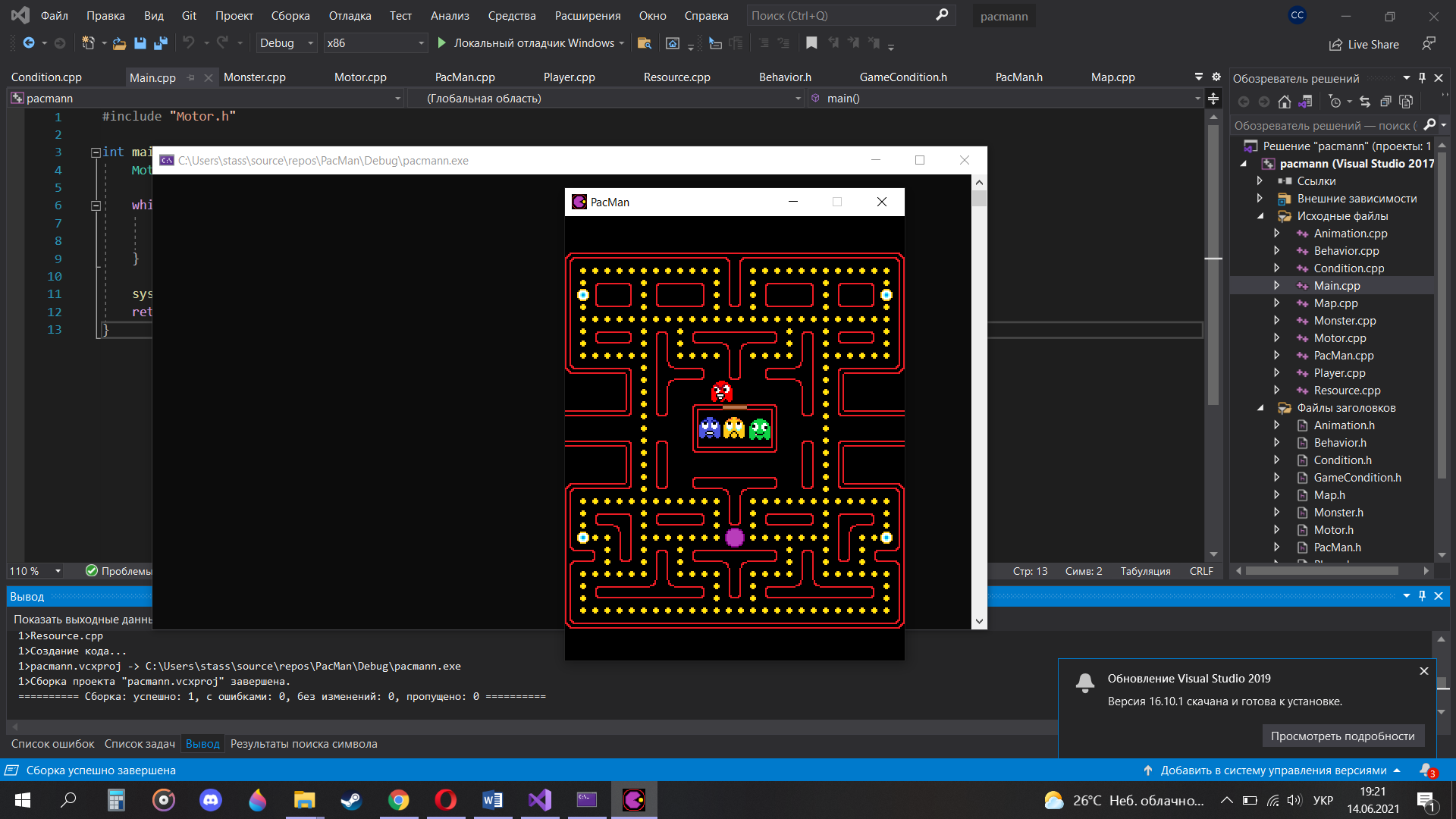
Проте крапочки і бонуси можна сприймати більш як складову карти.

Рисунок 4. – бонус

Монстри: вони представляють собою ботів, які повинні переслідувати Пакмана, перешкоджаючи йому шлях.

* Виходити з кімнати вони повинні послідовно, у певний момент часу, а саме тоді, коли Пакман з’їсть певну кількість крапочок.
* Також кожен з них повинен володіти своїм характером, тобто мати якісь особливості, наприклад завжди підстерігати Пакмана у якійсь частині карти.
* Також монстри, у випадку зіткнення зі стінкою карти не повинні зупинятися. Вони повинні змінювати траєкторію свого руху у коридор, що знаходиться поруч з ними.
* На певній відстані від Пакмана вони повинні починати рух в його сторону і постійно переслідувати його до кінця гри або до смерті Пакмана.
* У момент зіткнення монстра і Пакмана, Пакман повинен померти, після чого вся гравці переміщаються на свою точку відродження.
* Після того як Пакман поїдає бонус, монстри повинні ставати нешкідливими, і в такому разі сутичка Пакмана і монстра повинна призводити вже до смерті самого монстра. У випадку коли монстр помирає він починає гру зі своєї точки відродження, а саме біля дверей кімнати з монстрами.
* Монстри повинні рухатися повільніше за Пакмана.

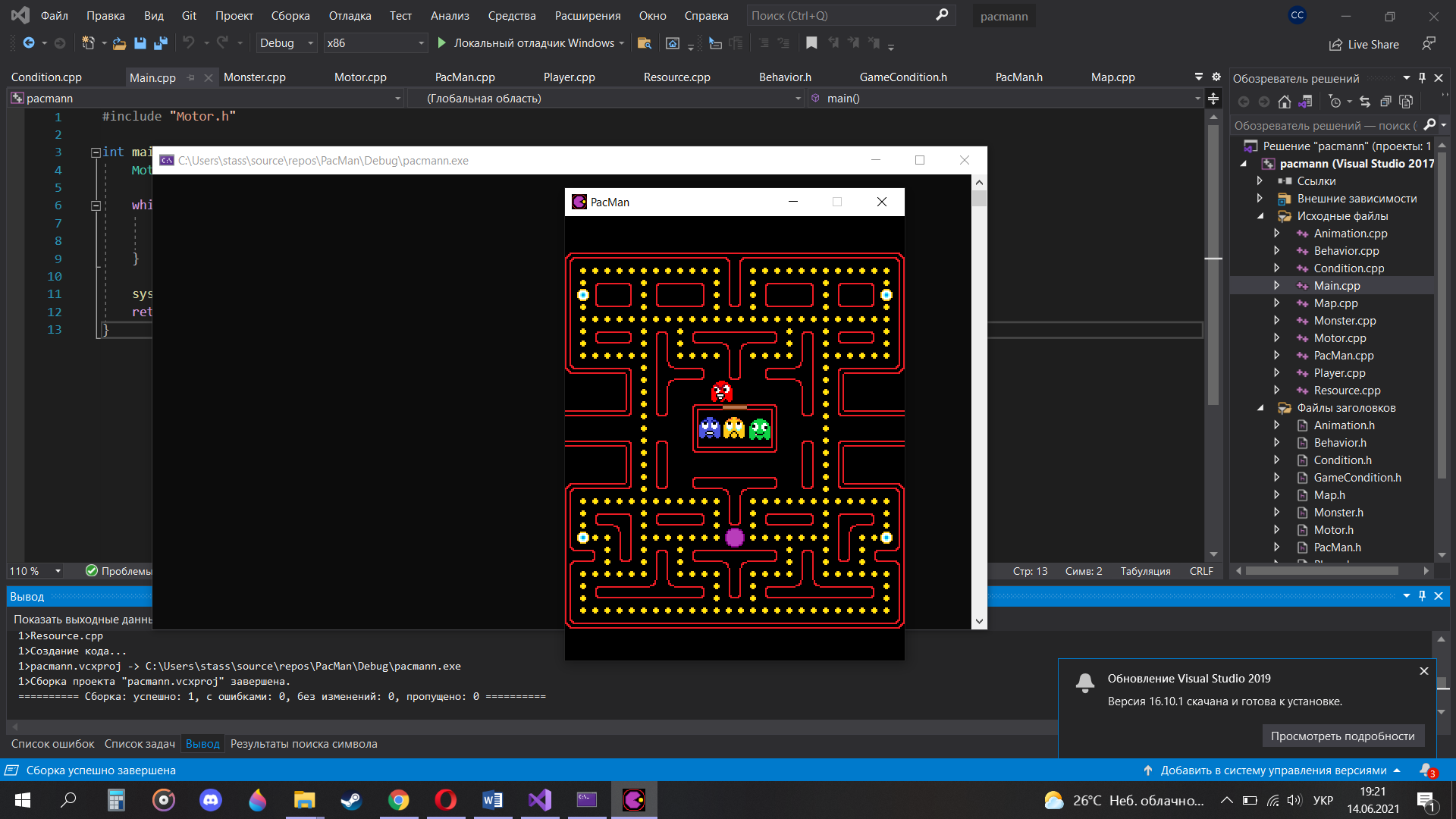
Аналізуючи функціонал монстрів можна прийти до висновку, що вони являють собою втілення примітивного штучного інтелекту.

Рисунок 5. – монстри

Пакман: головний персонаж в цій грі. Саме ним і керує гравець упродовж усього ігравого сеансу.

* Необхідно реалізувати керування персонажем за допомогою зовнішніх приладів взаємодії з комп’ютером, а саме за допомогою клавіатури.
* Як і у випадку з монстрами, при зустрічі зі стіною Пакман повинен зупиняти свій рух і чекати перенаправлення від грака. Якщо грок направляє Пакмана у напряму іншої поруч стіни — Пакман не повинен рухатися.
* Пакман після своєї смерті перероджується на своєму спавні до тих пір, поки не з’їсть усі крапочки і бонуси на карті.
* Як уже було сказано, сутичка Пакмана і монстра призводить до смерті одного з двох персонажів в залежності від стану, в якому перебуває монстр.

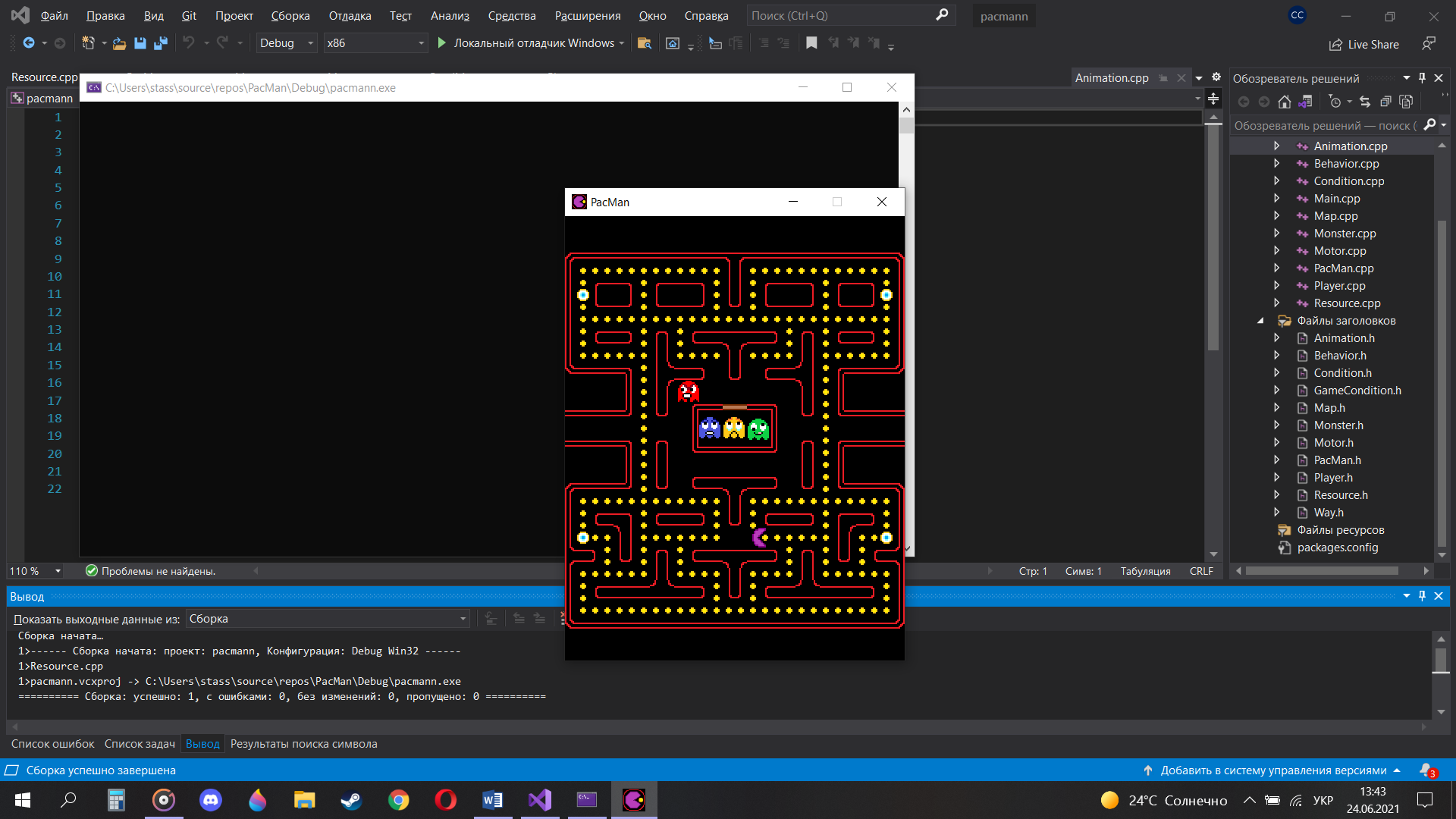
Зауважимо, що монстри і Пакман мають деякий спільний функціонал. Наприклад пересування по карті та телепортіцією в тунелі.

Рисунок 6. – Пакман

Також окрім логіки гри потрібно додати анімацію і текстуру її персонажам. Окремо додати текстуру для крапочок, бонусів, стінок, кутів карти і дверей у кімнаті монстрів. Ще необхідно додати іконку гри. Тобто наповнити гру візуальним контентом.

## 3 обгрунтування вибору середи програмної реалізації

C++ — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом в 1979 році та початково отримала назву «Сі з класами».

Об'єктно-орієнтоване програмування (ООП) — методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування.

Для того щоб розробити гру «PacMan» достатньо знати базові елементи об'єктно-орієнтованого програмування стосовно будови класів, принципу їх роботи, основи стандартної бібліотеки шаблонів (STL) та інші основні фрагменти ООП.

Класи і об'єкти в С ++ є основними концепціями об'єктно-орієнтованого програмування.

Бібліотека стандартних шаблонів (STL) - набір узгоджених узагальнених алгоритмів, контейнерів, засобів доступу до їх вмісту і різних допоміжних функцій в C ++.

Так як С++ підтримує об’ектно-орієнтовну розробку, можна вважати цю мову програмування підходящьою для розробки даного проекту.

Для застомування якоїсь мови програмування необхідна спеціальна програма в яку буде записуватися код. Така програма називається компілятор. В даному проекті використовується компілятор Microsoft Visual Studio 2019.

Microsoft Visual Studio — лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментів. Дані продукти дозволяють розробляти як консольні додатки, так і ігри та програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ.

У якості ігрового движка буде використовуватися стороння мультимедійна бібліотека SFML яка підтримується мовою програмування С++.

SFML — це вільна кроссплатформенная мультимедійна бібліотека, написана на мові C++. За її допомогою в грі буде реалізовано ініціалізація вікна та взаємодію користувача з ігровим процесом, відображення на екрані карти та ігрових персонажів та анімація персонажів.

## реалізація

В пункті «постановка задачі» було визначено, що в грі присутні такі елементи як Пакман, монстри і карта. Ці об’єкти будуть реалізовані за допомогою класів. Пакман і монстри рухаються за одним і тим самим принципом по карті. Тому можна винести цей функціонал в окремий клас, а Пакмана і монстрів зробити наслідниками від цього класу. За реалізацію поведінки персонажів(місце їх відродження, взаємодія персонажів з картою, початковий напрям руху кожного монстра, особливості монстрів, яким чином монстри будуть переслідувати Пакмана) також буде відповідати окремий клас. Також необхідно додати текстуру та анімацію персонажам. За цю роботу також будуть відповідати 2 окремих класи. Ще 2 класи будуть відповідати за створення та коректну роботу ігрового вікна, один з яких буде мати віртуальні методи. І Найголовніший клас, який буде об’єднуватиме в собі ініціалізацію вікна та поведінку персонажів. Тобто всього у рамках цього проекту буде використовуватися 10 класів. Також є окремий заголовочний файл, в який винесено принцип переміщення персонажів по карті і функція Main, яка буде запускати програму.

### 4.1 алгоритми гри

Розглянемо поведінку ігрових персонажів:

Пересікається з монстром

Їсть крапочки

Рухається

З’їв бонус

З’їв 5

З’їв 50

З’їв 100

З’їв всі

На карті з’являється новий монстр

Монстер помирає

Пакман помирає

Монстри переслідують Пакмана

Персонаж починає гру зі свого спавну

Монстри стають вразливими

Схема 1. Взаємодія персонажів між собою

На даній схемі зображено принцип дії усієї гри. При включенні гри, спочатку відбувається ініціалізація графічної системи та переказ відеокарти в графічний режим. На екрані з'являється зображення, починається гра. На початку гри Пакман не рухається, щоб привести його в рух необхідно натиснути клавіші керуючі Пакманом. Якщо натиснута кнопка, яка містить символ «S» або «↓» Пакман рухається вниз, «A» або «←» - вліво, «D» або «→» - вправо, «W» або «↑» - вгору, при цьому там, де він пройшов зникають точки, які Пакман повинен зібрати. Разом з цим кількість точок, очок вираховується і при їх кількості дорівнює 240 гра починається спочатку. Монстри з’являються на карті в момент коли Пакман з’їдає певну кількість крапочок після чого вони починають йти за ним. Якщо монстер і Пакман зустрінуться в межах однієї клітинки – Пакман помре. Але якщо Пакман з’їв бонус, то така зустріч призведе до смерті монстра. Якщо Пакмна з’їдає бонус – монстри стають білого кольору і стають вразливими на 2 секунди, після чого сутичка Пакмана з монстром призведе до смерті монстра. Кожен раз коли помирає якийсь ігровий персонаж – гра для нього починається спочатку. Також гра починається спочатку у випадку коли Пакман з’їсть усі крапочки на карті.

Щодо Пакмана і монстрів, вони також мають свої алгоритми.

Рухається

Переслідує Пакмана

Стикаєтьзя з стіною

Шукає вільний коридор

Схема 2. Поведінка монстрів

На схемі 2 описан штучний інтелект, який характеризує поведінку монстрів у грі. Їх головною задачею являється вбивство Пакмана. Тому вони повинні постійно знаходити найкоротший шлях для того щоб наздогнати його і з’їсти.

Принцип роботи Пакмана дещо схожий на принцип роботи монстрів за винятком того, що ним керує гравець.

Чекає команди від гравця зовні

Рухається у обраному гравцем напрямі

Стикаєтьзя з стіною

Схема 3. Як реалізоване керування пакманом

Схема 3 зображує принцип того як гравець обирає напрям в якому Пакман буде рухатися. Як ми бачимо, принцип пересування Пакмана по карті такий самий як і принцип пересування монстра. Єдина різниця в тому, що Пакман не обирає самостійно напрямок свого руху. На його рух впливає натискання певних кнопок на клавіатурі. В зв’язку з тим на яку клавішу натиснить гравець і буде обраний рух Пакмана.

### опис класів та їх значення у програмі

Як вже було зазначено раніше гра складається з 10 класів.

В першу чергу слід розглянути заголовочний файл “Way.h” в якому знаходиться інформація про те, в якому напрямі повинні рухатися ігрові персонажі.

**class Player**

Той самий клас від якого повинні наслідуватися класи, які описують всі необхідні методи для коректної роботи Пакмана та монстрів. В ньому знаходяться поля, які повинні приймати значення росположення персонажа на карті та на екрані. Також цей клас має гетери, які приймають ці самі значення і 2 методи: move і tp.

Метод move приймає 2 значення(значення х і у) і необхідний для того щоб гравці могли рухатися по карті;

Метод tp так само приймає 2 значення і необхідний для того щоб гравці заходячи в один бік тунелю виходили з іншого. І окремо цей метод необхідний для монстрів, так як для виходу зі своєї кімнати їм необхідно телепортуватися на спеціальне місце з координатами (13, 14) в яке в подальшому, в процесі гри, будуть переміщенні всі інші монстри.

**class PacMan**

Перший наслідник класа Player. Як можна здогадатися з назви цей клас існує для того щоб надати Пакману увесь необхідний для нього функціонал. Даний клас характерний такими полями як way, eatDot і Dead.

Клас також має гетери для вибору шляху та отримання значення стану крапочки і сетер смерть. Зпоміж цих методів також є метод queueWay який займається тим, що контролює коректність напрямку, тобто наглядає за тим щоб пакман правильно рухався у обраному гравцем напрямку;

Окремий метод від базового класу move, який вирішує у якому напрямку буде рухатися Пакман в залежності від отриманого значення;

Метод stop який у випадку сутички Пакмана зі стінкою або кутом, в яку можна упертися виходячи з коридору, зупиняє його і він чекає на новий сигнал зовні з клавіатури;

Метод eatDot який займається тим, що підраховує кількість з’їдених Пакманом крапочок

Метод isDead який повертає булівське значення і показує стан в якому перебуває Пакман, живий він чи мертвий.

**class Monster**

Другий наслідник класа Player. Цей клас вже описує весь необхідний функціонал для монстрів. Цей клас має в собі поля які відслідковують місце положення монстра по осі х і у на карті, поле шлях яке дає зрозуміти як монстер може рухатися, поле scattering яке необхідне для метода isScattering і сетера для цього поля, поле harmless яке необхідне для метода isHarmless і сетера для цього поля, булеве поле outCell(дивиться на те, знаходиться монстер у кімнаті монстрів чи ні) яке необхідне для метода outsideCell і метода tp і поле choice яке необхідне для метода shouldChoice і сетера для цього поля. З методів в класі є гетери для отримання окремо кожної координати по осі х і у і сетер який повертає обидва ці значення;

Гетер і сетер для взаємодії з параметрами переміщення, окремий від базового класу метод move який працює аналогічно з методом move у Пакмана, єдина різниця в тому, що монстри рухаються вдвічі повільніше від Пакмана;

Булівський метод isScattering, який вертає значення поля scattering в залежності від того, наляканий монстер чи ні;

Булівський метод isHarmless, який поступово віднімає по одиничці в лічильнику значення поля harmless, яке приймає нове значення при певних умовах у своєму сетері;

Метод outsideCell який повертає значення поля outCell;

Метод tp який необхідний для того, щоб монстри в певний момент могли спокійно вийти зі своєї кімнати який також пов’язаний з полем outCell.

Булівський метод shouldChoice, який повертає значення поля choice, тобто вибір монстра, в якому напрямку він збирається рухатися.

**Class Map**

Цей клас створює карту, яка являє собою квадратну матрицю розмірністю 28 на 36. Матриця заповнена числами від 0 до 30 кожне з яких являє собою якийсь елемент з лабіринту. В основному це стінки, крапочки і проходи;

Клас має метод isCross який повертає значення деяких координат. В цих місцях гра потребує уточнення, що в певний коридор персонажі можуть проходити. Без цього метода гравці не зможуть заходити в деякі повороти і коридори. Або взагалі будуть рухатися навстіж стінок і виходити за межі карти;

Також є метод takeDot який замінює значення активної крапочки на пустий коридор і додає одиничку в метод Пакмана eatDot;

Гетер який повертає значення місцеположення об**’**єкта по осям х і у;

І метод BlockChankPlayer який забороняє гравцям рухатися там де немає вільного коридору, крапочки або бонусу.

**Class GameCondition**

Базовий клас з віртуальними методами який необхідний для реалізації взаємодії зовнішніх пристроїв з грою.

В ньому є метод inicialization, cycle, render, keyPress і keyUnPress. Значення всіх цих полів дорінюють нулю. Їх функціонал буде реалізований в наслідниках цього класу. Сам по собі він не має якихось алгоритмів.

**Class Condition**

Наслідник від класа GameCondition. В класі є одне єдине поле condition, яке представлене у вигляді контейнера stack, і ті самі методи, що містяться у базовому класі. Кожен з цих методів регулює коректну роботу методів базового класу. Робить так, щоб останній елемент стеку завжди був першим на виході у всіх методах.

**Class Behavior**

Також являється наслідником від класа GameCondition. Цей клас займається тим, що встановлює поведінку штучного інтелекту монстрів і надає необхідний функціонал для взаємодії грака з Пакманом. Полями цього класу являються об’єкт класу Map(1 екземпляр), об’єкт класу PacMan(1 екземпляр),

об’єкти класу Monster(4 екземпляри) і поле waitTime. Методів в класі 10;

Метод render являє собою перевизначений метод базового класу, займається тим, що відмальвує текстуру персонажів в певний момент гри. Наприклад якшо монстер рухається вліво – він дивиться ліворуч;

Метод cycle також являється перевизначеним методом базового класу і займається тим, що реалізовує поведінку кожного з монстрів у момент коли вони являються небезпечними для гравця. Наприклад перший монстр просто переслідує Пакмана, тому що його особливістю є те, що він починаю гру одразу за межами кімнати. Другий монстр також переслідує Пакмана, але рухається він з такою ж самою швидкістю що і Пакман. Третій монстр з кожною з‘їденою Пакманом крапочкою швидше виходить із нешкідливого стану. Четвертий монстер завжди підстерігає Пакмана у лівому нижньому куту карти. Також в цьому методі описано, в який момент новий монстер виходить з кімнати, умова рестарту гри(момент коли Пакман зібрав всі крапочки на карті) і момент коли Пакман помирає(всі гравці повертаються на своє місце відродження але гра продовжується);

Метод keyPress займається тим, що прив’язує значення клавіш. За допомогою цього метода відбувається керування Пакманом за допомогою клавіатури;

Метод pacmanCanMove займаэться тим, що переміщує Пакмана по карті на відстань однієї клітинки в залежності від обраного гравцем напряму;

Таким же чином працює метод monsterCanMove. Тільки, як можна здогадатися з назви, він займається реалізацією переміщення монстрів по карті на відстань в одну клітинку карти;

Метод handleMonsterMovemet являє собою реалізацію штучного інтелекту монстрів. За допомогою цього метода монстри обирають напрямок свого руху, розуміють коли перед ними стоїть стінка і їм потрібно піти в іншому напрямку і помічають Пакмана на карті в певному радіусі;

Метод calcDist потрібен для того щоб монстри могли помічати Пакмана на певній від себе відстані;

Метод handleMonsterFrightened потрібен для того, щоб коли Пакман поїдав наляканого монстра він повертався на свою точку відродження і переставав бути наляканим;

Метод tpTunnel, як можна здогадатися з його назви, відповідає за телепортацію персонажів з кінця одного краю тунелю в інший край. Тобто якщо монстр або Пакман проходить в праву частину тунелю він завдяки цьму методу буде виходити з його лівої сторони.

**Class Resource**

Клас з статичними полями та методами, який необхідний для того щоб завантажити всі текстури лабіринту, крапочок, бонусів і персонажів. В ньому присутні такі методи як:

Метод load яки завантажує текстури карти.

Метод loadSprite який завантажує спрайти.

Метод get який встановлює зміну текстури персонажа в момент коли він рухається у певному напрямку.

Також в даному класі використовуються методи бібліотеки “SFML\Graphics.hpp” за допомогою яких в гру завантажуються всі необхідні спрайти для всіх персонажів у момент коли вони рухаються в певному напрямку.

**Class Animation**

Відповідає за анімацію персонажів. За допомогою методів цього класу відбувається зміна кадрів персонажів.

**Class Motor**

Являється по суті движком програми, найголовніший клас у програмі, оскільки об’еднує у собі функціонал гри, генерацію вікна та вивід всього контенту на екран.

Він вміщує в собі наступні методи:

Метод isRunning займається тим, що просто запускає вікно.

Метод initialization створює таке вікно, додає до ігрової програми іконку до додатку, робить так, щоб вікно не можна було деформувати або відкрити на весь екран(тому що якщо це зробити, гра не буде віглядати так, як повинна через співвідношення сторін монітору і розміру ігрового вікна).

Метод render відповідає за візуалізацію вікна. Також в цьому методі змінюється колір вікна на чорний.

Метод events потрібен для взаємодії з вікном. З його допомогою гра розуміє коли користувач натискає певну клавішу або закриває вікно.

**Функція Main**

Стартова точка програми. Саме в ній клас Motor ініціалізує вікно, і за допомогою циклу while працює і виводить зображення на екран до тих пір, поки користувач самостійно не закриє його.

### програмна реалізація

Якщо розглядати гру зі сторони програмної реалізації то виглядати вона повинна таким чином:

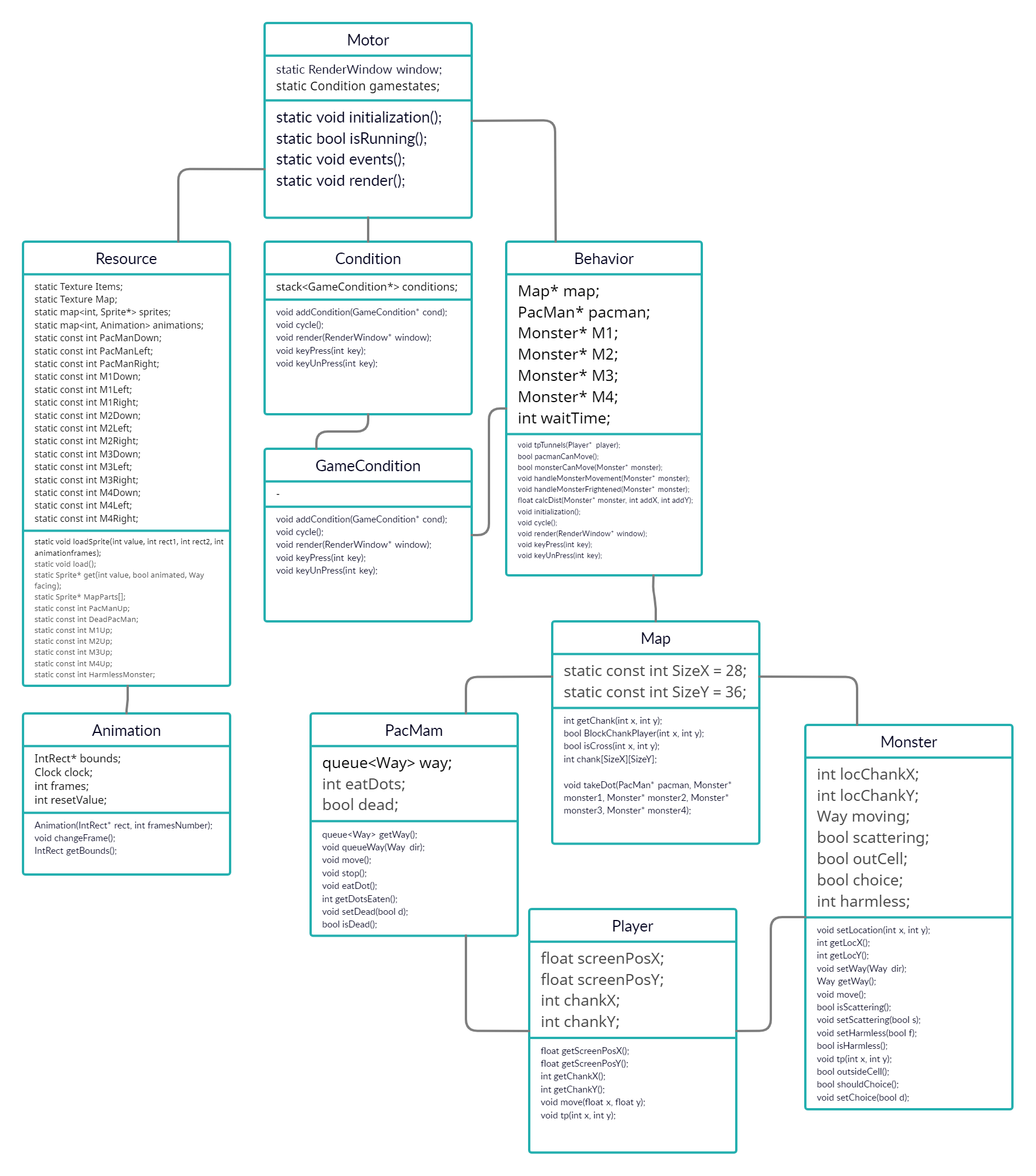


Схема 3. UML - діаграма, яка ілюструє принцип гри

Класами, які можна назвати фундаментом гри є Player, Animation і GameCondition. Саме з них починається будуватися основа ігрового процесу, анімації та зародження ігрових персонажів. Animation передає свої можливості в клас Resource, де до цієї анімації додаються текстури. Клас Player, завдяки наслідуванню, переходить в класи PacMan і Monster, в яких додається новий функціонал для ігрових персонажів, після чого вони переходять до класу Map де відбувається розстановка на карті Пакмана і монстрів. Потім цей клас разом з GameCondition переходить до класу Behavior в цьому класі відбувається насичення гри логікою. Також клас GameCondition переходить окремо в клас Condition, щоб відрегулювати правильну поведінку візуалізації вікна та розстановки гравців. І в самому кінці класи Condition, Resource і Behavior зтікаються у самий головний клас, який поєднує весь їх функціонал – Motor.

Таким чином і працює даний ігровий додаток. Використовуючи базові знання об'єктно-орієнтоване програмування та графічного адаптера SFML можна створювати прості але повноцінні 8-бітні ігри.

## інструкція користувача

Управління в грі класичне, здійснюється за допомого клавіш W, A, S, D або за допомогою стрілочок ↑, ←, ↓, →:

Натискаючи на «W» або «↑» - Пакман буде рухатися вперед.

Натискаючи на «A» або «←» - Пакман буде рухатися вліво.

Натискаючи на «S» або «↓» - Пакман буде рухатися назад.

Натискаючи на «D» або «→» - Пакман буде рухатися вправо.

Головна мета гри – з’їсти усі жовті крапочки та білі бонуси на карті. Для того щоб це зробити необхідно переміщуватися по лабіринту і старатися уникати сутички з монстрами.

Коли кількість зібраних крапочок буде дорівнювати 5 – в гру вступить новий монстр, який на початку знаходиться в центрі карти, в невелмчкій кімнаті. Аналогічна подія відбудиться коли персонаж збире 50 і 100 крапочок.

У випадку, якщо персонаж помре від рук хоча б одного монстра поки інщі знаходяться в своїй кімнаті – він продовжить гру, але зі своєї точки відродження. В той час всі монстри вже будуть задіяні в ігровому процесі незалежно від кількості з’їдених крапочок.

У випадку, якщо персонаж з’їдає бонус – монстри стають вразливими на 2,5 секунди. Сутичка з вразливим монстром призведе до смерті монстра, після чого той повернеться на свою точку відродження.

Якщо Пакман пересікається з ворожим монтром – Пакман помирає і переноситься на місце свого відродження. Монстри переносяться на місце свого відродження також.

Гра починається з початку коли Пакман з’їдає всі крапочки на карті.

# висновок

При розробці даної курсової роботи, була поставлена ​​задача розробити комп'ютерну гру "Pacman" з використанням алгоритмів штучного інтелекту. Для досягнення цієї мети були вивчені і відібрані відповідні алгоритми, а також розроблено математичну модель. Засобами платформи SFML і мови програмування C++, було розроблено додаток відповідає всім заявленим вимогам.

Таким чином мета даної курсової роботи була досягнута. В ході роботи вирішені такі завдання:

- вивчені і проаналізовані алгоритми штучного інтелекту;

- розроблена математичної моделі;

- розроблена гра Pacman на основі створеної моделі;

- реалізовані вибрані алгоритми;

- додаток протестовано;

- отримані знання побудова схем і UML діаграм.

# перелік джерел

1. <https://uk.wikipedia.org/wiki/C%2B%2B>
2. <https://uk.wikipedia.org/wiki/SFML>
3. <https://uk.wikipedia.org/wiki/Pac-Man>
4. <https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio>
5. Алгоритми. Побудова і аналіз / Томас Х. Кормен, Чарльз І. Лейзерсон, Рональд Л. Ривест, Кліффорд Штайн / Видавничий дім «Вільямс». 2012 р - 1290 с.
6. Мова програмування С ++ / Б. Страуструп / Біном. 2011 року - +1136 с.
7. Прийоми об'єктно-орієнтованого проектування. Патерни проектування / Еріх Гамма, Річард Хелм, Ральф Джонсон, Джон Вліссідес / Пітер. 2010 року - 366 c.

# додаток А

Файл «Animation.h»

#pragma once

#include "SFML/Graphics.hpp"

using namespace sf;

class Animation {

private:

IntRect\* bounds;

Clock clock;

int frames;

int resetValue;

public:

Animation(IntRect\* rect, int framesNumber);

void changeFrame();

IntRect getBounds();

};

Файл «Animation.cpp»

#include "Animation.h"

Animation::Animation(IntRect\* rect, int framesNumb) {

bounds = rect;

frames = framesNumb;

resetValue = rect->left;

}

void Animation::changeFrame() {

if (clock.getElapsedTime().asSeconds() >= 0.1f) { // скорость открывания рта

if (bounds->left >= (resetValue + 15 \* (frames - 1))) { // на сколько широко открывается рот

bounds->left = resetValue;

}

else {

bounds->left += 15; // чередование кадров

}

clock.restart();

}

}

IntRect Animation::getBounds() { return \*bounds; }

Файл «Behavior.h»

#pragma once

#include "GameCondition.h"

#include "Monster.h"

#include "Player.h"

#include "PacMan.h"

#include "map.h"

using namespace sf;

class Behavior : public GameCondition {

private:

Map\* map;

PacMan\* pacman;

Monster\* M1;

Monster\* M2;

Monster\* M3;

Monster\* M4;

int waitTime;

void tpTunnels(Player\* player);

bool pacmanCanMove();

bool monsterCanMove(Monster\* monster);

void handleMonsterMovement(Monster\* monster);

void handleMonsterFrightened(Monster\* monster);

float calcDist(Monster\* monster, int addX, int addY);

public:

void initialization();

void cycle();

void render(RenderWindow\* window);

void keyPress(int key);

void keyUnPress(int key);

};

Файл « Behavior.cpp»

#include "SFML/Graphics.hpp"

#include "Behavior.h"

#include "Resource.h"

#include "Monster.h"

#include "PacMan.h"

#include "Way.h"

#include <cstdlib>

#include <random>

#include <cmath>

// расстановка игроков и выбор их направления по умолчанию

void Behavior::initialization() {

srand(time(NULL));

waitTime = 0;

map = new Map();

pacman = new PacMan();

M1 = new Monster(13, 14, 10, 32);

M2 = new Monster(13, 17, 26, 4);

M3 = new Monster(11, 17, 23, 32);

M4 = new Monster(15, 17, 32, 4);

M1->tp(13, 14);

}

// отображение игроков на карте

void Behavior::render(RenderWindow\* window) {

Sprite pacmanSprite;

Sprite M1Sprite;

Sprite M2Sprite;

Sprite M3Sprite;

Sprite M4Sprite;

if (pacman->getWay().empty()) { pacmanSprite = \*Resource::get(Resource::PacManUp, false, Way::Unset); }

else { pacmanSprite = \*Resource::get(Resource::PacManUp, true, pacman->getWay().front()); }

if (pacman->isDead()) { pacmanSprite = \*Resource::get(Resource::DeadPacMan, true, Way::Unset); }

if (!M1->isHarmless()) { M1Sprite = \*Resource::get(Resource::M1Up, M1->outsideCell(), M1->getWay()); }

else { M1Sprite = \*Resource::get(Resource::HarmlessMonster, M1->outsideCell(), M1->getWay()); }

if (!M2->isHarmless()) { M2Sprite = \*Resource::get(Resource::M2Up, M2->outsideCell(), M2->getWay()); }

else { M2Sprite = \*Resource::get(Resource::HarmlessMonster, M2->outsideCell(), M2->getWay()); }

if (!M3->isHarmless()) { M3Sprite = \*Resource::get(Resource::M3Up, M3->outsideCell(), M3->getWay()); }

else { M3Sprite = \*Resource::get(Resource::HarmlessMonster, M3->outsideCell(), M3->getWay()); }

if (!M4->isHarmless()) { M4Sprite = \*Resource::get(Resource::M4Up, M4->outsideCell(), M4->getWay()); }

else { M4Sprite = \*Resource::get(Resource::HarmlessMonster, M4->outsideCell(), M4->getWay()); }

pacmanSprite.setPosition(pacman->getScreenPosX(), pacman->getScreenPosY());

M1Sprite.setPosition(M1->getScreenPosX(), M1->getScreenPosY());

M2Sprite.setPosition(M2->getScreenPosX(), M2->getScreenPosY());

M3Sprite.setPosition(M3->getScreenPosX(), M3->getScreenPosY());

M4Sprite.setPosition(M4->getScreenPosX(), M4->getScreenPosY());

for (int i = 0; i < Map::SizeX; i++) {

for (int j = 0; j < Map::SizeY; j++) {

Resource::MapParts[map->getChank(i, j)]->setPosition(i \* 16.0f, j \* 16.0f);

window->draw(\*Resource::MapParts[map->getChank(i, j)]);

}

}

window->draw(pacmanSprite);

window->draw(M1Sprite);

window->draw(M2Sprite);

window->draw(M3Sprite);

window->draw(M4Sprite);

}

void Behavior::cycle() {

// когда PacMan двигается а когда стоит

if (pacmanCanMove() && !pacman->isDead()) { pacman->move(); }

else { pacman->stop(); }

if (map->isCross(pacman->getChankX(), pacman->getChankY())) { pacman->stop(); }

map->takeDot(pacman, M1, M2, M3, M4);

// поведение каждого монстра

if (!pacman->getWay().empty()) {

if (!M1->isHarmless()) {

M1->setLocation(pacman->getChankX(), pacman->getChankY());

}

if (!M2->isHarmless()) {

switch (pacman->getWay().front()) {

case Way::Up:M2->setLocation(pacman->getChankX(), pacman->getChankY() - 4); break;

case Way::Down:M2->setLocation(pacman->getChankX(), pacman->getChankY() + 4); break;

case Way::Left:M2->setLocation(pacman->getChankX() - 4, pacman->getChankY()); break;

case Way::Right:M2->setLocation(pacman->getChankX() + 4, pacman->getChankY()); break;

}

}

if (!M3->isHarmless()) {

M3->setLocation(pacman->getChankX() + (M2->getChankX() - pacman->getChankX()), pacman->getChankY() + (M2->getChankY() - pacman->getChankY()));

}

if (!M4->isHarmless()) {

if (sqrt(pow((M4->getChankX() - (pacman->getChankX())), 2) + pow((M4->getChankY() - (pacman->getChankY())), 2)) < 9) {

M4->setLocation(pacman->getChankX(), pacman->getChankY());

}

else {

M4->setLocation(1, 32);

}

}

}

handleMonsterMovement(M1);

handleMonsterMovement(M2);

handleMonsterMovement(M3);

handleMonsterMovement(M4);

if (pacman->getDotsEaten() == 5) {

M2->tp(13, 14);

}

if (pacman->getDotsEaten() == 50) {

M3->tp(13, 14);

}

if (pacman->getDotsEaten() == 100) {

M4->tp(13, 14);

}

tpTunnels(pacman);

tpTunnels(M1);

tpTunnels(M2);

tpTunnels(M3);

tpTunnels(M4);

handleMonsterFrightened(M1);

handleMonsterFrightened(M2);

handleMonsterFrightened(M3);

handleMonsterFrightened(M4);

if (pacman->getDotsEaten() == 240) {

M1->tp(-2, -2);

M2->tp(-2, -2);

M3->tp(-2, -2);

M4->tp(-2, -2);

waitTime++;

}

if (pacman->isDead())

waitTime++;

if (waitTime == 200) {

if (pacman->isDead()) {

if (M1->outsideCell()) M1->tp(13, 14);

if (M2->outsideCell()) M2->tp(13, 14);

if (M3->outsideCell()) M3->tp(13, 14);

if (M4->outsideCell()) M4->tp(13, 14);

pacman->tp(13, 26);

pacman->setDead(false);

waitTime = 0;

}

else {

Behavior::initialization();

waitTime = 0;

}

}

}

// способ управления PacMan

void Behavior::keyPress(int key) {

switch (key) {

case Keyboard::W:pacman->queueWay(Way::Up); break;

case Keyboard::Up:pacman->queueWay(Way::Up); break;

case Keyboard::S:pacman->queueWay(Way::Down); break;

case Keyboard::Down:pacman->queueWay(Way::Down); break;

case Keyboard::A:pacman->queueWay(Way::Left); break;

case Keyboard::Left:pacman->queueWay(Way::Left); break;

case Keyboard::D:pacman->queueWay(Way::Right); break;

case Keyboard::Right:pacman->queueWay(Way::Right); break;

}

}

// отжатие кнопок

void Behavior::keyUnPress(int key) {}

// осуществление движения PacMan по карте

bool Behavior::pacmanCanMove() {

if (!pacman->getWay().empty()) {

switch (pacman->getWay().front()) {

case Way::Up:return !map->BlockChankPlayer(pacman->getChankX(), pacman->getChankY() - 1); break;

case Way::Down:return !map->BlockChankPlayer(pacman->getChankX(), pacman->getChankY() + 1); break;

case Way::Left:return !map->BlockChankPlayer(pacman->getChankX() - 1, pacman->getChankY()); break;

case Way::Right:return !map->BlockChankPlayer(pacman->getChankX() + 1, pacman->getChankY()); break;

}

}

return true;

}

// движение монстров по карте

bool Behavior::monsterCanMove(Monster\* monster) {

switch (monster->getWay()) {

case Way::Up:return !map->BlockChankPlayer(monster->getChankX(), monster->getChankY() - 1); break;

case Way::Down:return !map->BlockChankPlayer(monster->getChankX(), monster->getChankY() + 1); break;

case Way::Left:return !map->BlockChankPlayer(monster->getChankX() - 1, monster->getChankY()); break;

case Way::Right:return !map->BlockChankPlayer(monster->getChankX() + 1, monster->getChankY()); break;

default:return false;

}

}

// осуществление движения мостров (искуственный интелект)

void Behavior::handleMonsterMovement(Monster\* monster) {

if (monster->isHarmless()) {

if (monster->getChankX() == monster->getLocX() && monster->getChankY() == monster->getLocY()) {

monster->setHarmless(false);

}

}

if (map->isCross(monster->getChankX(), monster->getChankY())) {

if (monster->shouldChoice()) {

float dRight = calcDist(monster, 1, 0);

float dLeft = calcDist(monster, -1, 0);

float dUp = calcDist(monster, 0, -1);

float dDown = calcDist(monster, 0, 1);

if (dRight < dLeft && dRight < dUp && dRight < dDown) {

monster->setWay(Way::Right);

}

else if (dLeft < dRight && dLeft < dUp && dLeft < dDown) {

monster->setWay(Way::Left);

}

else if (dUp < dLeft && dUp < dRight && dUp < dDown) {

monster->setWay(Way::Up);

}

else if (dDown < dLeft && dDown < dUp && dDown < dRight) {

monster->setWay(Way::Down);

}

}

monster->setChoice(false);

}

else {

monster->setChoice(true);

}

if (monsterCanMove(monster) && monster->outsideCell()) {

monster->move();

}

else {

monster->setChoice(true);

}

}

// дистанция на которой монстры замечабт на PacMan

float Behavior::calcDist(Monster\* monster, int addX, int addY) {

float distance = 1000000.0f;

if (!map->BlockChankPlayer(monster->getChankX() + addX, monster->getChankY() + addY)) {

distance = (float)sqrt(pow((monster->getLocX() - (monster->getChankX() + addX)), 2) + pow((monster->getLocY() - (monster->getChankY() + addY)), 2));

}

return distance;

}

// поедание замороженного монстра

void Behavior::handleMonsterFrightened(Monster\* monster) {

if (pacman->getChankX() == monster->getChankX() && pacman->getChankY() == monster->getChankY()) {

if (monster->isHarmless()) {

monster->tp(13, 14);

monster->setHarmless(false);

}

else {

pacman->setDead(true);

M1->tp(-2, -2);

M2->tp(-2, -2);

M3->tp(-2, -2);

M4->tp(-2, -2);

}

}

}

// проход через тунель

void Behavior::tpTunnels(Player\* player) {

if (player->getChankX() == 0 && player->getChankY() == 17) {

player->tp(26, 17);

}

else if (player->getChankX() == 27 && player->getChankY() == 17) {

player->tp(1, 17);

}

}

Файл «Condition.h»

#pragma once

#include "SFML/Graphics.hpp"

#include "GameCondition.h"

#include <stack>

using namespace sf;

using namespace std;

class Condition {

private:

stack<GameCondition\*> conditions;

public:

void addCondition(GameCondition\* cond);

void cycle();

void render(RenderWindow\* window);

void keyPress(int key);

void keyUnPress(int key);

};

Файл « Condition.cpp»

#include "Condition.h"

// методы для коректной работы и рендеринга окна

void Condition::addCondition(GameCondition\* cond) {

conditions.push(cond);

conditions.top()->initialization();

}

void Condition::cycle() {

conditions.top()->cycle();

}

void Condition::render(RenderWindow\* window) {

conditions.top()->render(window);

}

void Condition::keyPress(int key) {

conditions.top()->keyPress(key);

}

void Condition::keyUnPress(int key) {

conditions.top()->keyUnPress(key);

}

Файл «Main.cpp»

#include "Motor.h"

int main() {

Motor::initialization();

while (Motor::isRunning()) {

Motor::events();

Motor::render();

}

system("pause");

return 0;

}

Файл «GameCondition.h»

#pragma once

#include "SFML/Graphics.hpp"

using namespace sf;

class GameCondition {

public:

virtual void initialization() {}

virtual void cycle() = 0;

virtual void render(RenderWindow\* window) = 0;

virtual void keyPress(int key) = 0;

virtual void keyUnPress(int key) = 0;

};

Файл «Map.h»

#pragma once

#include "PacMan.h"

#include "Monster.h"

class Map {

public:

static const int SizeX = 28;

static const int SizeY = 36;

Map();

int getChank(int x, int y);

bool BlockChankPlayer(int x, int y);

bool isCross(int x, int y);

void takeDot(PacMan\* pacman, Monster\* monster1, Monster\* monster2, Monster\* monster3, Monster\* monster4);

private:

int chank[SizeX][SizeY];

};

Файл «Map.cpp»

#include "Map.h"

// карта

Map::Map() : chank{

{30, 30, 30, 0, 13, 13, 13, 13, 13, 13, 13, 13, 1, 30, 30, 30, 8, 30, 9, 30, 30, 30, 0, 13, 13, 13, 13, 17, 28, 13, 13, 13, 13, 1, 30, 30},

{30, 30, 30, 8, 26, 26, 27, 26, 26, 26, 26, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 26, 26, 27, 11, 10, 26, 26, 26, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 2, 14, 3, 26, 2, 3, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 2, 3, 26, 6, 7, 26, 2, 3, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 11, 10, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 11, 10, 26, 26, 26, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 11, 10, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 11, 6, 14, 14, 3, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 6, 15, 7, 26, 6, 7, 26, 6, 13, 13, 13, 7, 30, 6, 13, 13, 13, 7, 26, 6, 15, 15, 15, 7, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 2, 14, 3, 26, 2, 14, 14, 14, 14, 14, 14, 3, 30, 2, 14, 14, 14, 3, 26, 2, 3, 26, 2, 14, 14, 7, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 6, 15, 15, 3, 2, 15, 15, 7, 30, 6, 15, 15, 15, 7, 26, 11, 10, 26, 6, 15, 15, 3, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 26, 26, 26, 11, 10, 30, 30, 30, 30, 30, 30, 30, 30, 30, 26, 11, 10, 26, 26, 26, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 2, 3, 26, 11, 10, 30, 18, 12, 12, 12, 19, 30, 2, 3, 26, 11, 10, 26, 2, 3, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 6, 15, 7, 26, 11, 10, 26, 6, 7, 30, 9, 30, 30, 30, 8, 30, 11, 10, 26, 6, 7, 26, 11, 10, 26, 6, 7, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 26, 26, 26, 26, 11, 10, 26, 30, 30, 30, 9, 30, 30, 30, 8, 30, 11, 10, 26, 26, 26, 26, 11, 10, 26, 26, 26, 26, 9, 30, 30},

{30, 30, 30, 24, 14, 14, 14, 3, 26, 11, 6, 14, 14, 3, 30, 31, 30, 30, 30, 8, 30, 11, 6, 14, 14, 3, 30, 11, 6, 14, 14, 3, 26, 9, 30, 30},

{30, 30, 30, 16, 15, 15, 15, 7, 26, 11, 2, 15, 15, 7, 30, 31, 30, 30, 30, 8, 30, 11, 2, 15, 15, 7, 30, 11, 2, 15, 15, 7, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 26, 26, 26, 26, 11, 10, 26, 30, 30, 30, 9, 30, 30, 30, 8, 30, 11, 10, 26, 26, 26, 26, 11, 10, 26, 26, 26, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 2, 14, 3, 26, 11, 10, 26, 2, 3, 30, 9, 30, 30, 30, 8, 30, 11, 10, 26, 2, 3, 26, 11, 10, 26, 2, 3, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 6, 7, 26, 11, 10, 30, 22, 13, 13, 13, 23, 30, 6, 7, 26, 11, 10, 26, 6, 7, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 26, 26, 26, 11, 10, 30, 30, 30, 30, 30, 30, 30, 30, 30, 26, 11, 10, 26, 26, 26, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 2, 14, 14, 7, 6, 14, 14, 3, 30, 2, 14, 14, 14, 3, 26, 11, 10, 26, 2, 14, 14, 7, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 6, 15, 7, 26, 6, 15, 15, 15, 15, 15, 15, 7, 30, 6, 15, 15, 15, 7, 26, 6, 7, 26, 6, 15, 15, 3, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 2, 14, 3, 26, 2, 3, 26, 2, 12, 12, 12, 3, 30, 2, 12, 12, 12, 3, 26, 2, 14, 14, 14, 3, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 11, 10, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 11, 2, 15, 15, 7, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 11, 30, 10, 26, 11, 10, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 11, 10, 26, 26, 26, 26, 11, 10, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 6, 15, 7, 26, 6, 7, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 6, 7, 26, 2, 3, 26, 6, 7, 26, 9, 30, 30},

{30, 30, 30, 8, 26, 26, 27, 26, 26, 26, 26, 26, 9, 30, 30, 30, 8, 30, 9, 30, 30, 30, 8, 26, 26, 26, 27, 11, 10, 26, 26, 26, 26, 9, 30, 30},

{30, 30, 30, 4, 12, 12, 12, 12, 12, 12, 12, 12, 5, 30, 30, 30, 8, 30, 9, 30, 30, 30, 4, 12, 12, 12, 12, 25, 20, 12, 12, 12, 12, 5, 30, 30} }

{

}

// места стычек pacman со стенками карты

bool Map::isCross(int x, int y) {

return (x == 1 && y == 4) || (x == 6 && y == 4) || (x == 12 && y == 4) || (x == 15 && y == 4) || (x == 21 && y == 4) || (x == 26 && y == 4)

|| (x == 1 && y == 8) || (x == 6 && y == 8) || (x == 9 && y == 8) || (x == 12 && y == 8) || (x == 15 && y == 8) || (x == 18 && y == 8) || (x == 21 && y == 8) || (x == 26 && y == 8)

|| (x == 1 && y == 11) || (x == 6 && y == 11) || (x == 9 && y == 11) || (x == 12 && y == 11) || (x == 15 && y == 11) || (x == 18 && y == 11) || (x == 21 && y == 11) || (x == 26 && y == 11)

|| (x == 9 && y == 14) || (x == 12 && y == 14) || (x == 15 && y == 14) || (x == 18 && y == 14)

|| (x == 6 && y == 17) || (x == 9 && y == 17) || (x == 12 && y == 17) || (x == 15 && y == 17) || (x == 18 && y == 17) || (x == 21 && y == 17) || (x == 26 && y == 17)

|| (x == 9 && y == 20) || (x == 18 && y == 20)

|| (x == 1 && y == 23) || (x == 9 && y == 23) || (x == 6 && y == 23) || (x == 12 && y == 23) || (x == 15 && y == 23) || (x == 18 && y == 23) || (x == 21 && y == 23) || (x == 26 && y == 23)

|| (x == 1 && y == 26) || (x == 3 && y == 26) || (x == 6 && y == 26) || (x == 9 && y == 26) || (x == 12 && y == 26) || (x == 15 && y == 26) || (x == 18 && y == 26) || (x == 21 && y == 26) || (x == 24 && y == 26) || (x == 26 && y == 26)

|| (x == 1 && y == 29) || (x == 3 && y == 29) || (x == 6 && y == 29) || (x == 9 && y == 29) || (x == 12 && y == 29) || (x == 15 && y == 29) || (x == 18 && y == 29) || (x == 21 && y == 29) || (x == 24 && y == 29) || (x == 26 && y == 29)

|| (x == 1 && y == 32) || (x == 12 && y == 32) || (x == 15 && y == 32) || (x == 26 && y == 32)

|| (x == 13 && y == 14) || (x == 14 && y == 15);

}

// поедание точек

void Map::takeDot(PacMan\* pacman, Monster\* m1, Monster\* m2, Monster\* m3, Monster\* m4) {

if (chank[pacman->getChankX()][pacman->getChankY()] == 26) {

chank[pacman->getChankX()][pacman->getChankY()] = 30;

pacman->eatDot();

}

else if (chank[pacman->getChankX()][pacman->getChankY()] == 27) {

chank[pacman->getChankX()][pacman->getChankY()] = 30;

m1->setHarmless(true);

m2->setHarmless(true);

m3->setHarmless(true);

m4->setHarmless(true);

}

}

// места где игроки не могут пройти

bool Map::BlockChankPlayer(int x, int y) { return chank[x][y] != 30 && chank[x][y] != 26 && chank[x][y] != 27; }

int Map::getChank(int x, int y) { return chank[x][y]; }

Файл «Monster.h»

#pragma once

#include "Player.h"

#include "Way.h"

class Monster : public Player {

private:

int locChankX;

int locChankY;

Way moving;

bool scattering;

bool outCell;

bool choice;

int harmless;

public:

Monster(int tilePosX, int tilePosY, int destinationX, int destinationY);

void setLocation(int x, int y);

int getLocX();

int getLocY();

void setWay(Way dir);

Way getWay();

void move();

bool isScattering();

void setScattering(bool s);

void setHarmless(bool f);

bool isHarmless();

void tp(int x, int y);

bool outsideCell();

bool shouldChoice();

void setChoice(bool d);

};

Файл «Monster.cpp»

#include "Monster.h"

Monster::Monster(int chankPosX, int chankPosY, int locationX, int locationY) : Player(chankPosX, chankPosY) {

setLocation(locationX, locationY);

setWay(Way::Unset);

scattering = true;

outCell = false;

choice = true;

harmless = false;

}

// определение позиции

void Monster::setLocation(int x, int y) { locChankX = x; locChankY = y; }

int Monster::getLocX() { return locChankX; }

int Monster::getLocY() { return locChankY; }

// движение

void Monster::setWay(Way w) { moving = w; }

Way Monster::getWay() { return moving; }

// скорость движение монстров

void Monster::move() {

switch (moving) {

case Way::Up:Player::move(0, -0.2f); break;

case Way::Down:Player::move(0, 0.2f); break;

case Way::Left:Player::move(-0.2f, 0); break;

case Way::Right:Player::move(0.2f, 0); break;

}

}

// рассеянность монстров

bool Monster::isScattering() { return scattering; }

void Monster::setScattering(bool s) { scattering = s; }

// монстры в коробке

bool Monster::outsideCell() { return outCell; }

// принятие решение монстров

bool Monster::shouldChoice() { return choice; }

void Monster::setChoice(bool d) { choice = d; }

// телепортация монстров

void Monster::tp(int x, int y) {

Player::tp(x, y);

outCell = true;

}

// условие заморозки монстра

void Monster::setHarmless(bool f) {

if (f) { harmless = 2500; }

else { harmless = 0; }

}

// замороженный монстер

bool Monster::isHarmless() {

if (harmless > 0) harmless--; {

return harmless > 0;

}

}

Файл «Motor.h»

#pragma once

#include "SFML/Graphics.hpp"

#include "Condition.h"

class Motor {

private:

static RenderWindow window;

static Condition gamestates;

public:

static void initialization();

static bool isRunning();

static void events();

static void render();

};

Файл «Motor.cpp»

#include "Behavior.h"

#include "Resource.h"

#include "Motor.h"

RenderWindow Motor::window;

Condition Motor::gamestates;

// открытие окна

bool Motor::isRunning() { return window.isOpen(); }

// создание окна

void Motor::initialization() {

Resource::load();

window.create(VideoMode(448, 586), "PacMan", Style::Close | Style::Titlebar);

gamestates.addCondition(new Behavior);

Image icon;

icon.loadFromFile("img/icon.png");

window.setIcon(32, 32, icon.getPixelsPtr());

}

// визуализация окна

void Motor::render() {

window.clear(Color::Black);

gamestates.render(&window);

window.display();

}

// взаимодействие с окном

void Motor::events() {

Event event;

while (window.pollEvent(event)) {

switch (event.type) {

case Event::Closed:window.close(); break;

case Event::KeyPressed:gamestates.keyPress(event.key.code); break;

case Event::KeyReleased:gamestates.keyUnPress(event.key.code); break;

}

}

gamestates.cycle();

}

Файл «PacMan.h»

#pragma once

#include "Player.h"

#include "Way.h"

#include <queue>

using namespace std;

class PacMan : public Player {

private:

queue<Way> way;

int eatDots;

bool dead;

public:

PacMan();

queue<Way> getWay();

void queueWay(Way dir);

void move();

void stop();

void eatDot();

int getDotsEaten();

void setDead(bool d);

bool isDead();

};

Файл « PacMan.cpp»

#include "PacMan.h"

// начальный спавн PacMan

PacMan::PacMan() : Player(13, 26) { eatDots = 0; }

// проверка на проходимость

void PacMan::queueWay(Way w) {

if (!way.empty()) {

if (w == -way.front()) {

queue<Way> clear;

swap(way, clear);

}

}

if (way.size() < 2) {

way.push(w);

}

}

// скорость перемещение PacMan

void PacMan::move() {

if (!way.empty()) {

switch (way.front()) {

case Way::Up:Player::move(0, -0.4f); break;

case Way::Down:Player::move(0, 0.4f); break;

case Way::Left:Player::move(-0.4f, 0); break;

case Way::Right:Player::move(0.4f, 0); break;

}

}

}

// условия остановки PacMan

void PacMan::stop() {

if (way.size() > 1) {

if ((int)(screenPosX + 8) % 16 == 0 && (int)(screenPosY + 8) % 16 == 0) {

switch (way.front()) {

case Way::Up:way.pop(); break;

case Way::Down:way.pop(); break;

case Way::Left:way.pop(); break;

case Way::Right: way.pop(); break;

}

}

}

}

// смотрит на состояние движения PacMan

queue<Way> PacMan::getWay() { return way; }

// PacMan сьел точку

void PacMan::eatDot() { eatDots++; }

int PacMan::getDotsEaten() { return eatDots; }

// PacMan умер

void PacMan::setDead(bool d) { dead = d; }

bool PacMan::isDead() { return dead; }

Файл «Player.h»

#pragma once

class Player {

protected:

float screenPosX;

float screenPosY;

int chankX;

int chankY;

public:

Player(int chankPosX, int chankPosY);

float getScreenPosX();

float getScreenPosY();

int getChankX();

int getChankY();

void move(float x, float y);

void tp(int x, int y);

};

Файл « Player.cpp»

#include "Player.h"

#include <cmath>

// местоположения игроков

Player::Player(int chankPosX, int chankPosY) {

chankX = chankPosX;

chankY = chankPosY;

screenPosX = chankPosX \* 16.0f + 16.0f;

screenPosY = chankPosY \* 16.0f + 8.0f;

}

float Player::getScreenPosX() { return screenPosX; }

float Player::getScreenPosY() { return screenPosY; }

int Player::getChankX() { return chankX; }

int Player::getChankY() { return chankY; }

// осущкствление перемещение игроков

void Player::move(float x, float y) {

screenPosX += x;

screenPosY += y;

if ((int)(screenPosX + 8) % 16 == 0 && (int)(screenPosY + 8) % 16 == 0) {

chankX = (int)round((screenPosX - 8) / 16);

chankY = (int)round((screenPosY - 8) / 16);

}

}

// осущкствление телепортация игроков

void Player::tp(int x, int y) {

chankX = x;

chankY = y;

screenPosX = x \* 16.0f + 8.0f;

screenPosY = y \* 16.0f + 8.0f;

}

Файл «Resource.h»

#pragma once

#include "SFML/Graphics.hpp"

#include "Animation.h"

#include <iostream>

#include "Way.h"

#include <map>

using namespace std;

using namespace sf;

class Resource {

private:

static Texture Items;

static Texture Map;

static map<int, Sprite\*> sprites;

static map<int, Animation> animations;

static void loadSprite(int value, int rect1, int rect2, int animationframes);

static const int PacManDown;

static const int PacManLeft;

static const int PacManRight;

static const int M1Down;

static const int M1Left;

static const int M1Right;

static const int M2Down;

static const int M2Left;

static const int M2Right;

static const int M3Down;

static const int M3Left;

static const int M3Right;

static const int M4Down;

static const int M4Left;

static const int M4Right;

public:

static void load();

static Sprite\* get(int value, bool animated, Way facing);

static Sprite\* MapParts[];

static const int PacManUp;

static const int DeadPacMan;

static const int M1Up;

static const int M2Up;

static const int M3Up;

static const int M4Up;

static const int HarmlessMonster;

};

Файл «Resource.cpp»

#include "Resource.h"

map<int, Sprite\*> Resource::sprites;

map<int, Animation> Resource::animations;

Texture Resource::Items;

Texture Resource::Map;

Sprite\* Resource::MapParts[32];

// анимации PacMan и монстров

const int Resource::PacManUp = 0;

const int Resource::PacManDown = 1;

const int Resource::PacManLeft = 2;

const int Resource::PacManRight = 3;

const int Resource::M1Up = 4;

const int Resource::M1Down = 5;

const int Resource::M1Left = 6;

const int Resource::M1Right = 7;

const int Resource::M2Up = 8;

const int Resource::M2Down = 9;

const int Resource::M2Left = 10;

const int Resource::M2Right = 11;

const int Resource::M3Up = 12;

const int Resource::M3Down = 13;

const int Resource::M3Left = 14;

const int Resource::M3Right = 15;

const int Resource::M4Up = 16;

const int Resource::M4Down = 17;

const int Resource::M4Left = 18;

const int Resource::M4Right = 19;

const int Resource::HarmlessMonster = 20;

const int Resource::DeadPacMan = 21;

// загрузка текстур для анимаций игроков

void Resource::load() {

Map.loadFromFile("img/map.png");

int index = 0;

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 4; j++) {

MapParts[index] = new Sprite(Map, IntRect(i \* 8, j \* 8, 8, 8));

MapParts[index]->setScale(2.0f, 2.0f);

index++;

}

}

Items.loadFromFile("img/things.png");

loadSprite(Resource::PacManUp, 0, 0, 3);

loadSprite(Resource::PacManDown, 45, 0, 3);

loadSprite(Resource::PacManLeft, 90, 0, 3);

loadSprite(Resource::PacManRight, 135, 0, 3);

loadSprite(Resource::DeadPacMan, 0, 75, 12);

loadSprite(Resource::M1Up, 0, 15, 2);

loadSprite(Resource::M1Down, 30, 15, 2);

loadSprite(Resource::M1Left, 60, 15, 2);

loadSprite(Resource::M1Right, 90, 15, 2);

loadSprite(Resource::M2Up, 0, 30, 2);

loadSprite(Resource::M2Down, 30, 30, 2);

loadSprite(Resource::M2Left, 60, 30, 2);

loadSprite(Resource::M2Right, 90, 30, 2);

loadSprite(Resource::M3Up, 0, 45, 2);

loadSprite(Resource::M3Down, 30, 45, 2);

loadSprite(Resource::M3Left, 60, 45, 2);

loadSprite(Resource::M3Right, 90, 45, 2);

loadSprite(Resource::M4Up, 0, 60, 2);

loadSprite(Resource::M4Down, 30, 60, 2);

loadSprite(Resource::M4Left, 60, 60, 2);

loadSprite(Resource::M4Right, 90, 60, 2);

loadSprite(Resource::HarmlessMonster, 120, 15, 2);

}

// смена кадров

Sprite\* Resource::get(int value, bool animated, Way facing) {

if (value != Resource::HarmlessMonster) {

switch (facing) {

case Way::Down: value += 1; break;

case Way::Left: value += 2; break;

case Way::Right:value += 3; break;

}

}

if (animated) {

animations.at(value).changeFrame();

sprites.at(value)->setTextureRect(animations.at(value).getBounds());

}

return sprites.at(value);

}

// загрузка спрайтов

void Resource::loadSprite(int value, int rect1, int rect2, int animationframes) {

IntRect\* rect = new IntRect(rect1, rect2, 15, 15);

Sprite\* sprite = new Sprite(Items, \*rect);

sprite->setScale(2.0f, 2.0f);

sprite->setOrigin(7.5f, 7.5f);

sprites.insert(pair<int, Sprite\*>(value, sprite));

animations.insert(pair<int, Animation>(value, Animation(rect, animationframes)));

}

Файл «Way.h»

#pragma once

enum Way {

Up = 1,

Down = -1,

Left = 2,

Right = -2,

Unset = 0

};