КН-19-2
Голяк Д.В.

ЛАБОРАТОРНА РОБОТА 3
Аналіз і формалізація вимог до програмного продукту. Розробка технічного завдання на створення програмного продукту

Мета роботи: навчитися аналізувати й формалізувати вимоги замовника з використанням UML, розраховувати витрати на створення програмного продукту, виконувати планування робіт, розробляти та оформлювати технічне завдання на створення програмного продукту відповідно до ДСТУ.

Завдання:

1. Вивчити теоретичні відомості.

2. Виконати аналіз і формалізацію вимог замовника на розробку програмного продукту відповідно до індивідуального завдання.

3. Розробити діаграму прецедентів використання й виконати опис прецедентів.

4. Виконати розрахунок витрат на створення програмного продукту.

5. Виконати планування робіт зі створення програмного продукту.

6. Розробити технічне завдання на створення програмного продукту.

7. Зробити висновки про вибір моделі створення програмного продукту.

Хід роботи:

1) Сформулюємо індивідуальне завдання:

Тема: розробка однокористувацької двовимірної гри "Тетріс" з використанням принципів ООП.

"Тетріс" є головоломкою, побудованою на використанні геометричних фігур "тетраміно" - різновиду поліміно, що складаються з чотирьох квадратів.

Випадкові фігурки тетраміно падають зверху в прямокутний стакан шириною 10 і висотою 20 клітин. У польоті гравець може повертати фігурку та рухати її по горизонталі. Також можна «скидати» фігурку, тобто прискорювати її падіння, коли вже вирішено, куди фігурка повинна впасти. Фігурка летить, поки не наткнеться на іншу фігурку або на дно склянки. Якщо при цьому заповнився горизонтальний ряд з 10 кліток, він пропадає і все, що вище нього, опускається на одну клітку.

У спеціальному полі гравець бачить фігурку, яка буде слідувати після поточної — ця підказка дозволяє планувати свої дії. Темп гри поступово збільшується. Назва гри походить від кількості клітин, з яких складається кожна фігура. Гра закінчується, коли нова фігурка не може поміститися в стакан. Гравець отримує бали за кожну фігурку, тому його задача — заповнювати ряди, не заповнюючи саму склянку якомога довше, щоб таким чином отримати якомога більше балів.

2) **Розробимо діаграму прецедентів:**

Опис прецеденту «Моніторинг загальної статистики»

Основний виконавець – Гра **Зацікавлені особи** – Гравець
Передумови: • Наявність ігрових сесій (мінімум 1) • Наявність створеного акаунту (логін, пароль)
Вхідні дані: • Дані акаунту
Основний успішний сценарій (основний процес): • Гра надає можливість переглянути кількість зіграних сесій, найкращий результат, час у грі, загальний рахунок, рівень гравця
Частота виконання: • для кожного гравця
Постумови (результати): • Отримання загальної статистики гравця
Вихідні дані: • Загальна статистика гравця

Опис прецеденту «Моніторинг поточної статистики»

Основний виконавець – Гра **Зацікавлені особи** – Гравець
Передумови: • Запуск поточної гри
Вхідні дані: • Дані поточної гри

Основний успішний сценарій (основний процес): • Гра надає можливість переглянути поточний рахунок у грі, рівень складності, кількість складених ліній, наступну фігуру (залежно від налаштувань)
Частота виконання: • для кожного гравця
Постумови (результати): • Отримання поточної статистики гравця
Вихідні дані: • Поточна статистика гравця

Опис прецеденту «Геймплей»

Основний виконавець – Гра **Зацікавлені особи** – Гравець
Передумови: • Запуск поточної гри
Вхідні дані: • Дані поточної гри • Дії гравця
Основний успішний сценарій (основний процес): • Гравець має можливість виконувати певні дії для отримання результатів у грі (рухати та повертати фігури, складати лінії)
Частота виконання: • для кожного гравця
Постумови (результати): • Гравець завершує гру самостійно або у результаті виконання умов гри (кінець гри) та отримує додаткову статистику у профілі
Вихідні дані: • Результуюча статистика гри (час, рівень, рахунок)

3) Виконаємо розрахунок витрат на створення програмного продукту:

Для визначення договірної ціни розраховуються поточні витрати на виконання програмного продукту. Ці витрати розраховуються як сума прямих та накладних витрат за весь період створення програми.

Поточні витрати на створення програмного продукту, повязані з використанням ПК для виконання даної роботи, розраховуються виходячи з загального часу $T_{заг}$, собівартості людино-години $C_{люд}$ та години експлуатації ПК $C_{пк}$, а також вартості ПЗ, необхідного для розробки:

$B_{ПП} = (C_{люд} + C_{пк}) * T_{заг} + C_{пз}$

Почасова ставка $C_{люд}$ = 100 грн/год.

Вартість години роботи ПК = 10 грн/год.

Вартість використаного ПЗ = 0 грн (Безкоштовна VS Community Edition 2019).

Розрахуємо загальний час розробки $T_{заг}$:

$T_{заг} = T_{підг} + T_{код} + T_{зв}$, де $T_{підг}$ – час аналізу замовлення, створення ТЗ, пошуку необхідної інформації та ресурсів, $T_{код}$ – час написання та відладки коду, $T_{зв}$ – час створення звітів з розробки ПП.

$T_{заг} = 10 + 50 + 5 = 65$ годин.

Розрахуємо поточні витрати на розробку програмного продукту $B_{ПП}$:

$B_{ПП} = (100 + 10) * 65 + 0 = 7150$ грн.

4) Сформулюємо технічне завдання:

Додаток буде створений для платформ Windows 7,8,10 x32,x64.

Виконуваний файл програми – TetrisOOP.exe.

Гра повинна складатися з меню, налаштувань, самої гри. Гра відбувається на двовимірному полі розміром 10x20 блоків.

Меню повинно складатися з наступних пунктів:

- Старт => веде до початку гри
- Налаштування => веде до налаштувань гри
- Про гру => веде до статистики нарахованих очок
- Вихід => вихід з гри
- Бокове меню із статистикою гравця (може грати як гість, тоді статистика не зберігається, або може зареєструватися зі збереженням)

Налаштування повинні включати такі можливості:

- Налаштування геймплею (опціональна демонстрація наступної фігури)
- Налаштування графіки (наприклад кольорова/чорно-біла, віконне/повноекранне відображення)
- Налаштування керуючих клавіш (переміщення та поворот фігур)
- Зберігати користувальницькі параметри та скинути їх до базових
- Налаштування звуку

Геймплей:

У меню користувач може увійти до акаунту та розпочати гру.

У ході гри на полі з'являються двовимірні фігури, що складаються із блоків. Фігури з'являються один за одним. Фігура з'являється у верхній частині екрана в середині поля. Момент появи нової фігури: остаточне встановлення попередньої фігури на полі. Після появи фігура починає поступальний рух до кінця поля.

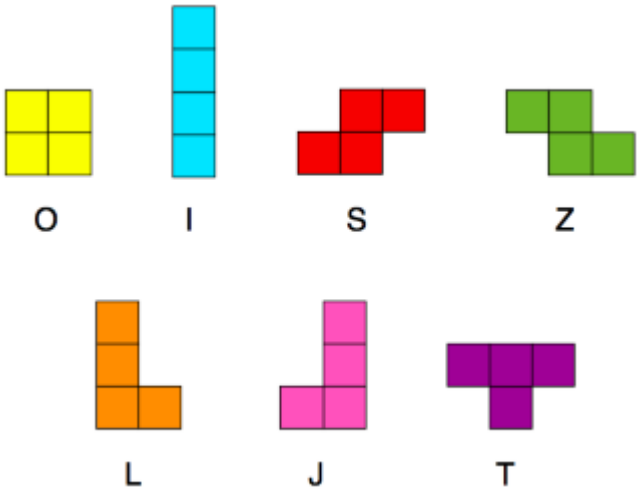
Під час налаштування встановлюється час переходу до нового блоку, який використовується для входу гри.

Момент зупинки руху фігури: досягнення кінця поля чи досягнення блоку зайнятого іншою фігурою.

При повному заповненні останнього рядка фігурами він знищується, встановлені фігури зсуваються на 1 блок вниз.

Фігури можна повертати на 90 градусів за годинниковою стрілкою за наявності відповідного місця (незайнятих іншими фігурами блоків або обмеження розміру поля). Фігури можна переміщати ліворуч-праворуч за наявності відповідного місця (незайнятих іншими фігурами блоків або обмеження розміру поля).

Типи фігур:



Нарахування очок

При знищенні одного рядка нараховується X очок.

Перехід на наступні рівні

При знищенні певної кількості рядків / отриманні очок відбувається перехід на наступний рівень.

Кінець гри

Неможливо нікуди посунути нову фігуру, що з'явилася, з’являється вікно кінця.

Також потрібно реалізувати функцію паузи під час гри.

Висновки: я навчився розраховувати витрати на створення програмного продукту, виконувати планування робіт та складати технічне завдання на створення програмного продукту.

Я вивчив теоретичні відомості, а також виконав аналіз і формалізацію вимог замовника на розробку гри Tetris у ООП виконанні.

Розробив діаграму прецедентів використання й виконав опис прецедентів додатку. Розрахував витрати на створення програмного продукту, у результаті отримав суму у 7150 грн, яку вважаю справедливою для цього проекту.

Також написав детальне технічне завдання на створення програми, завдяки якому можливо у найменші сроки розробити максимальну кількість функціоналу гри.