МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ Кафедра інформатики та прикладної математики

Реєстраційний №\_\_\_\_\_\_\_\_\_\_ «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_2016р.

**КУРСОВА РОБОТА З ПРОГРАМУВАННЯ**

**РОЗРОБКА ІГРОВОЇ ПРОГРАМИ «Пожежники» В ПРОГРАМІ ІНТЕРПРИТАТОР**

студента І курсу групи І-15

фізико-математичного факультету

напрям підготовки

6.040302 Інформатика

**Пірогов Владислава Миколайовича**

Керівник: старший викладач, к.т.н.

Дубан Роман Миколайович

Оцінка:

Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Шкала ECTS\_\_\_ Кількість балів\_\_\_

Члени комісії \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кривий Ріг 2016

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ**

Кафедра інформатики та прикладної математики

Завдання

на курсову роботу

студенту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ група\_\_\_\_\_\_\_\_

*(прізвище, ім’я, по-батькові)*

|  |
| --- |
| Термін здачі роботи \_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ |

|  |
| --- |
| **Тема: РОЗРОБКА ІГРОВОЇ ПРОГРАМИ «МОРСЬКИЙ БІЙ» В ПРОГРАМІ ІНТЕРПРИТАТОР** |
| 1. Аналіз предметної області й постановка задачі. |
| 1.1 Мотивація |
| 1.2 Аналіз аналогів та предметної області. |
| 1.3 Постановка задач |
| 2. Проектування задачі. |
| 2.1 User Story. |
| 2.2 Use Case. |
| 2.3 Алгоритми задачі. |
| 2.4 Інтерфейси. |
| 3. Архітектура задачі. |
| 3.1 Дані. |
| 3.2 Алгоритми реалізації. |
| 3.3 Функції модулі. |
| 4. Тестування. |
| 4.1 Unit Test. |
| 4.2 User Test |
| 4.3 Інструкція, Презентація. |
|  |

**Реферат**

* кількість частин записки: 4
* кількість ілюстрацій:18
* кількість таблиць : 1
* кількість додатків: 3
* кількість джерел: 10

перелік ключових слів: об’єктно-оріентоване програмування; Python; PyQt; Pygame;аркадні ігри

Зміст

[Вступ 4](#_Toc452664465)

[1. Аналіз предметної області й постановка задачі. 5](#_Toc452664466)

[1.1 Мотивація. 5](#_Toc452664467)

[1.2 Аналіз аналогів та предметної області. 7](#_Toc452664468)

[1.3 Постановка задач. 10](#_Toc452664469)

[2. Проектування задачі. 13](#_Toc452664470)

[2.1 User Story. 13](#_Toc452664471)

[2.2 Use Case. 14](#_Toc452664472)

[2.3 Алгоритми Задач 16](#_Toc452664473)

[3.4 Інтерфейси 18](#_Toc452664474)

[3. Архітектура задачі 20](#_Toc452664475)

[3.1 Дані 20](#_Toc452664476)

[3.2 Алгоритми реалізації 21](#_Toc452664477)

[3.3 Функції модулі. 29](#_Toc452664478)

[4. Тестування 32](#_Toc452664479)

[4.1 User Test 32](#_Toc452664480)

[4.2 Інструкція 32](#_Toc452664481)

[Висновки 33](#_Toc452664482)

[Джерела 33](#_Toc452664483)

[Додатки 33](#_Toc452664484)

# Вступ

Дана програма слугує для розвитку реакцій та розваги користувача та гарного проведення часу .

Оскільки в даній програмі було використано об'єктно орієнтоване програмування , через те що даний метод програмування використовує об’єкти - це дозволяє використовувати інкапсуляцію та наслідування класів. Як наслідок ми здатні використати структури даних та інші характеристики створені в батьківському класі об’єктів в інших класах шо утворені від батьківського. В такий спосіб ми зменшуємо кількість рядків , та покращимо сам код програми та його загальний об’єм.

Сама програма представляє з себе гру «Пожежники» . Програма створює меню в якому користувач запускає гру. За умовою гравець повинен встигати ловити усіх падаючих людей . Інакше, якщо він не впіймає хоч одного чоловіка то ,користувач програє. Також користувач не повинен ловити жодного камінця , у випадку якщо гравець впіймає хоч один камінь то він втрачає бали . Так у випадку втрати всіх балів гравець програє .Гравець виграє тільки у випадку набору 100 балів .В графічному інтерфейсі гри є індикатор шо показує кількість балів.

# Аналіз предметної області й постановка задачі.

## Мотивація.

Все з чогось починається, і навіть важко уявити, що всього лише чверть століття тому не було ніяких комп'ютерних ігор, безкоштовних ігрових автоматів та Інтернету. Але люди старшого віку напевно пам'ятають цю першу в Радянському союзі портативну ігрову консоль під назвою «Електроніка», прозвану «Вовк ловить яйця» або «Ну, постривай!». У цьому ігровому апараті гравець, керуючи мультяшним вовком, натискаючи на відповідні кнопочки, повинен був ловити яйця, які падають з лотків. Спочатку все «ловиться» легко, але швидкість і складність гри поступово збільшувалася і доходила до нереальності. Заєць в грі участі не брав, хоча і з'являвся іноді на екрані. Ця іграшка на початку 90-х наробила багато шуму. Про цей портативному ігровому автоматі ходили легенди, мовляв, якщо пройти гру до кінця, тобто не допустити розбитих яєць, то в нагороду покажуть мультик з вовком. Правда, ніхто цього мультика не бачив, але всі в нього вірили. «Ну, постривай!» практично весь час були «в роботі» і переходили з рук в руки. Тоді ніхто й не здогадувався, що популярна радянська іграшка «Електроніка» під назвою «Вовк ловить яйця», була ні чим іншим як копією американського портативного гаджета Nintendo Game & Watch.

Перший такий ігровий апарат був випущений 28 квітня 1980 року. Сенс концепції полягав у тому, що на рідко-кристалічні дисплеї крім самої гри були ще і годинник. Були створені подібні ігрові автомати з різними персонажами і сюжетами. Лінія портативних ігрових апаратів Game & Watch (гра і годинник) мала дуже велику популярність і проводилася компанією Nintendo по 1991 рік. Ну а далі естафету підхопили радіозавод СРСР. До речі, перші радянські ігрові апарати «Електроніка» були не з вовком, а яйця ловив Міккі Маус. Зараз, звичайно, багато що змінилося. Але для тих, хто відчуває ностальгію за тим часу, можна порекомендувати абсолютно безкоштовно скачати гру «Ну, постривай!» (Є багато варіантів, в тому числі і осучаснена версія) або «ловити разом з вовком яйця» он-лайн прямо в Інтернеті.

Але не тільки ця гра э прабатьком даного проекту так механіка руху багатьох об’єктів взята з гри тетріс .

Тетріс (похідне від «тетраміно» і «теніс» [2]) - комп'ютерна гра, винайдена в СРСР Олексієм Пажитновим і представлена громадськості 6 червня 1984 года [1]. Ідею «Тетріс» йому підказала куплена ним гра в пентаміно.

Назва гри походить від кількості клітин, з яких складається кожна фігура.

Правила :Випадкові фігурки тетраміно падають зверху в прямокутний стакан шириною 10 і висотою 20 клітин. У польоті гравець може повертати фігурку на 90 ° і рухати її по горизонталі. Також можна «скидати» фігурку, тобто прискорювати її падіння, коли вже вирішено, куди фігурка повинна впасти. Фігурка летить до тих пір, поки не наткнеться на іншу фігурку або на дно склянки. Якщо при цьому заповнився горизонтальний ряд з 10 клітин, він пропадає і все, що вище нього, опускається на одну клітку. Додатково показується фігурка, яка буде слідувати після поточної - це підказка, яка дозволяє планувати гравцеві дії. Темп гри поступово збільшується. Гра закінчується, коли нова фігурка не може поміститися в стакан. Гравець отримує очки за кожен заповнений ряд, тому його завдання - заповнювати ряди, не заповнюючи сам стакан (по вертикалі) якомога довше, щоб таким чином отримати якомога більше очок.

Даний проект є ідейним наслідувачем обох цих ігор , як з точки зору логіки виконання коду , так и фактичним нащадком.

## Аналіз аналогів та предметної області.

1) MICO MACO.

Офіційний сайт : <http://www.igames.com.ua/FlashGame.aspx?name=mico_maco>

Користувач повинен грати проти комп’ютера. В самій грі присутня велика кількість ігрових об’єктів та їх ефектів , як , наприклад , при падінні на землю так і у випадку якщо гравець зловить об’єкт. Наприклад об’єкт банан: в випадку якщо користувач піймає банан він отримує бали , але якщо користувач не зловить об’єкт банан , то банан впаде на землю и у випадку коли гравець наступить на банан гравця перемістить за ігрове поле і він втратить одну з життів . Так в грі присутня велика кількість анімацій пов’язаних з взаємодією гравця та ігрових об’єктів . Управління в самій грі на відміну від проекту є окрема кнопка шо виконує функцію переходу до режиму «ловіння». В цьому режими персонаж не здатен рухатись. На відміну від моєї програми. Графічний інтерфейс досить гарно оформлений в ньому присутній індикатор життів та показник балів набрати гравцем . Перед запуском гри , користувач попаде на головне меню , з нього можливо запустити гру , переглянути лист рекордів , та присутня інструкція . Та сама гра не є зрозумілою, адже правила та ефекти після взаємодії з об’єктами , ні де пояснюється . Також в грі є звукові ефекти , так постійно головному меню грає музика та при натисненні на кнопки грає звуковий ефект . В самій грі грає вже інша музика . Але нема можливості її налаштувати , зробити тихіше , або взагалі вимкнути і з часом ці звуки починають відволікати та злити . Управління зв’язано з стрілками ( в ліво , в право ) та з кнопкою ( пробіл ) . Нижче приклад ігрового процесу на рис 1.1



Рис.1.1 приклад ігрового процесу MICO MACO

2) СНІГОВИК

Офіційний сайт :

[**https://scratch.mit.edu/projects/62494276/**](https://scratch.mit.edu/projects/62494276/)

Досить проста гра. Присутній досить простий ігровий інтерфейс в якому є відображення кількості набраних балів гравця . Управління пов’язано з стрілками ( в ліво , в право ) . До її недоліків можна віднести те що вона нескінчена, в ній неможливо програти або виграти.

Нижче приклад ігрового процесу на рис 1.2

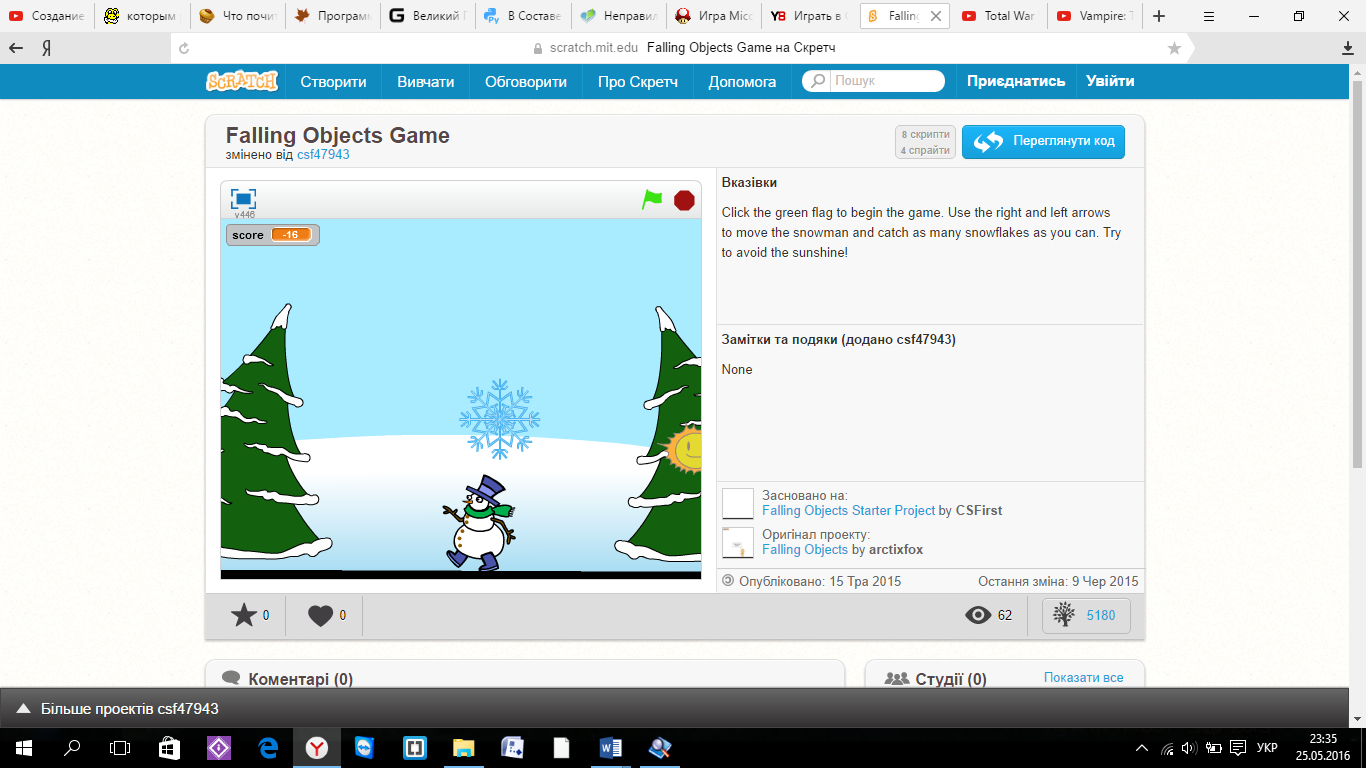


Рис 1.2 приклад ігрового процесу СНІГОВИК

3) GARFIELD : LASAGNA FROM HEAVEN

Офіційний сайт :

<http://ru.y8.com/games/garfield_lasagna_from_heaven>

Ця гра досить гарно оформлена . В самій грі присутня велика кількість ігрових об’єктів , сам інтерфейс представлений дуже просто , в ньому є показник часу до кінця рівня . Та в грі присутня анімація , так кіт одкриває рот при взаємодій з об’єктом . Але управління в грі є незручним , тому що воно пов’язано з мишкою та не завжди реагує на дії користувача .

Нижче приклад ігрового процесу на рис 1.3

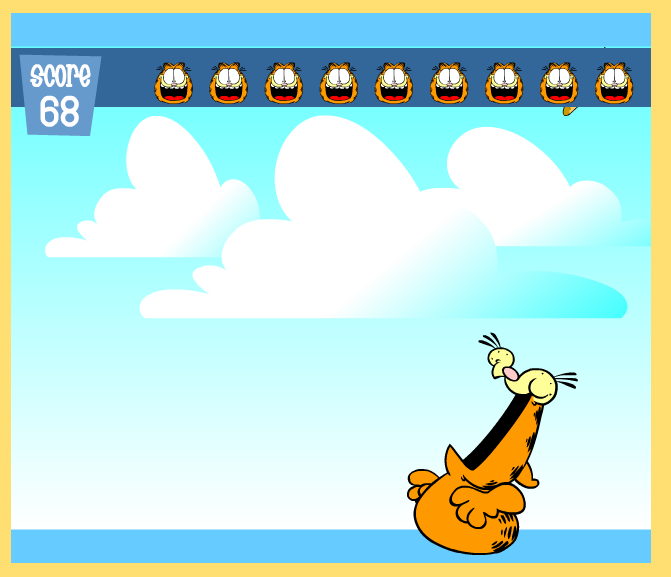


Рис 1.3 приклад ігрового процесу GARFIELD : LASAGNA FROM HEAVEN

## Постановка задач.

Опис

Ця курсова робота студента першого курсу , що є результатом самостійного опрацювання матеріалу ООП.Данна гра має безліч аналогів , та багато версій гри , але дана робота матиме найпростіший вигляд.

Дана програма служить для розваги користувача.

Якщо користувач запускає гру , йому відкриється меню звідки і буде запушена гра .

Основний опис умови гри представляє з себе наступне . З верхньої частини вікна рухається вниз певна кількість об’єктів Y, гравець повинен "зловити" всі об'єкти. Інакше гравець програє . Також в грі присутні об’єкти пастки , які ловити заборонено , кожен пійманий об’єкт пастка віднімає певну кількість балів , якщо кількість балів менша за нуль або рівне – гравець програє . Управління в грі здійснюється за допомогою . Далі йде обсяг проекту .

Обсяг проекту:

* Ігрове меню з трьома кнопками :

“Start”

“Record List”

“Quit”

* Вікно з фоном
* Лічильник часу
* Об'єкт (X): Графічно відображається: батут розміром 1/8 вікна. Рухається по осі.Ох
* Об'єкт (Y): Графічно відображається: падаюча людина розміром дорівнює батуту. Рухається зверху в низ
* Об'єкт пастка: Графічно відображається: камінь розміром 1/8 вікна. Рухається зверху в низ.

Дії:

Натиснути на кнопку “Start” – Почнеться гра

Натиснути на кнопку “Record List” – Покаже список набраних балів

Натиснути на кнопку “Quit” – Закриває меню

Спіймати - нижні координати Об’єкта Y або Об'єкта пастка співпадають з верхніми координатами Об’єкта Х.

Наслідок: якщо зловити Об’єкта Y він знищитися, а гравець отримує один бал . Якщо координати співпадають з кінцем вікна гра закінчитися. Якщо зловити Об'єкт пастка гравець втрачає 15 балів . У випадку втрати всіх балів гравець програє .Умови перемоги: гравець повинен набрати 100 балів.

Як видно цей проект хоч і є досить простим , але досить поширений . Але для точнішого розбору та створення проекту були побудовані такі User Story та Use Case а на основі їх створити алгоритми виконання проекту .

# Проектування задачі.

## User Story.

Призначені для користувача історії (англ. User Story) - спосіб опису вимог до розроблюваної системі, сформульованих як одне або більше пропозицій на повсякденному або діловою мовою користувача. Призначені для користувача історії використовуються гнучкими методологіями розробки програмного забезпечення для специфікації вимог. Кожна призначена для користувача історія обмежена в розмірі та складності. Часто історія пишеться на маленькій паперовій картці. Це гарантує, що вона не стане занадто великий. В екстремальних програмуванні призначені для користувача історії пишуться користувачами (замовниками) системи. У методології SCRUM - пишуться або схвалюються роллю власника продукту (англ. Product Owner). Для замовників (користувачів) призначені для користувача історії є основним інструментом впливу на розробку програмного забезпечення.

Призначені для користувача історії - швидкий спосіб документувати вимоги клієнта, без необхідності розробляти великі формалізовані документи і згодом витрачати ресурси на їх підтримку. Мета користувальницьких історій полягає в тому, щоб бути в змозі оперативної і без накладних витрат реагувати на швидко змінюються вимоги реального світу.

Призначена для користувача історія залишається неофіційними визначенням вимог, поки відсутня процедура приймального тестування. Перш, ніж реалізовувати призначену для користувача історію, клієнт повинен визначити відповідну приймальню процедуру, щоб гарантувати що цілі користувальницької історії були досягнуті.

З даного визначення , для кращого розуміння програми та її більш точної розробки . Було створено наступні User Story :

1. Я як користувач хочу мати меню гри з кнопками .
2. Я як користувач хочу мати кнопку початку гри для того щоб почати гру
3. Я як користувач хочу мати кнопку для перегляду результатів гри .
4. Я як користувач хочу мати кнопку для виходу з гри .
5. Я як користувач хочу мати вікно де гра буде відображатися для того щоб почати гру.
6. Я як користувач хочу мати задній фон .
7. Я як користувач хочу мати об’єкт люди для того щоб мати можливість набирати бали .
8. Я як користувач хочу щоб об’єкт люди рухались того щоб мати можливість зловити їх .
9. Я як користувач хочу мати об’єкт батут для того щоб мати можливість ловити людей та набирати бали .
10. Я як користувач хочу мати можливість рухати батут для того щоб мати можливість ловити людей та набирати бали .
11. Я як користувач хочу мати об’єкти пастка для того щоб в грі можна було програти .
12. Я як користувач хочу мати інформаційну лінію для того щоб бачити кількість балів .
13. Я як користувач хочу мати можливість вийти з гри для того щоб завершити гру.   
    1. Use Case.

Прецедент використання (англ. use case) - в розробці програмного забезпечення та системному проектуванні це опис поведінки системи, коли вона взаємодіє з кимось (або чимось) із зовнішнього середовища. Система може відповідати на зовнішні запити Актора (англ. Actor) (може застосовуватися термін актантом), може сама виступати ініціатором взаємодії. Іншими словами, сценарій використання описує, «хто» і «що» може зробити з розглянутої системою, або що система може зробити з «ким» або «чому». Методика сценаріїв використання застосовується для виявлення вимог до поведінки системи, відомих також як призначені для користувача і функціональні вимоги.

У системному проектуванні сценарії використання застосовуються на більш високому рівні ніж при розробці програмного забезпечення, часто представляючи мети зацікавлених осіб або місії. На стадії аналізу вимог сценарії використання можуть бути перетворені в ряд детальних вимог та задокументовані за допомогою діаграм вимог SysML або інших подібних механізмів.

З даного визначення , для кращого розуміння програми та її більш точної розробки . Було створено наступні Use Case :

№1 :

Назва: Камінь, що падає

Опис: Камінь, що падає з гори екрана

Передумова: гра триває

Тригер : Користувач набрав достатню кількість очок

Основний сценарій : Камінь падає з гори екрана, користувач ухиляється від нього

Результат: Користувач залишається при своїх балах, гра продовжується.

Додатковий сценарій : Камінь падає з гори екрана, користувач не ухиляється від нього

Додатковий результат: Користувач втрачає 15 балів , якщо у користувача було менше 15 балів, гра завершується

№2 :

Назва: Людина, що падає

Опис: Людина, що падає з гори екрана

Передумова: гра почалась

Тригер : гра почалась

Основний сценарій : Людина падає з гори екрана, користувач її ловить

Результат: Користувач отримує один бал, гра продовжується.

Додатковий сценарій : Людина падає з гори екрана, користувач не ловить її

Додатковий результат: гра завершується

№3 :

Назва: Батут, що рухається

Опис: Батут, що рухається вздовж екрану

Передумова: Гра почалась

Тригер : Гра почалась

Основний сценарій : Гравець тисне на клавіши(стрілка ліво, стрілка право), батут рухається

Результат: Користувач отримує можливість ловити об’єкти .

Додатковий сценарій : Гравець не тисне на клавіши

Додатковий результат: Батут не рухається

№4 :

Назва: Меню запуску

Опис: Меню, в якому користувач обирає складність та починає гру

Передумова: Гра була запущена

Основний сценарій : Гравець обирає складність та запускає гру

Результат: Запускається гра з обраним рівнем складності

Додатковий сценарій : Гравець вимикає меню

Додатковий результат: Гра закривається

№5:

Назва: Вивід балів в інформаційному рядку

Опис: Рядок, в якому показуються бали гравця

Передумова: Гра була запущена

Тригер : Гра почалась

Основний сценарій : Користувач отримує бали

Результат: Бали відображаються на інформаційному рядку

* 1. Алгоритми Задач

Алгоритм (латінізов. Algorithmi за араб. Ім'ям узб. Математика аль-Хорезмі) - набір інструкцій, Які описують порядок Дій Виконавця, щоб досягти результату розв'язання задачі за скінченну Кількість Дій; система правил виконання дискретного процесу, яка досягає поставленої мети за скінченній годину. Для візуалізації алгоритмів часто Використовують блок-схеми.

Для комп'ютерних програм алгоритм є списком деталізованих інструкцій, що реалізують процес обчислення, Який, починаючи з початкових стану, відбувається через послідовність логічних станів, яка завершується кінцевім станом. Перехід з попереднього до наступного стану НЕ обов'язково детермінованій - деякі алгоритми можуть містити елементи випадковості.

На основі даних Use Case і User Story було розроблені такі алгоритми задач:

№1 Меню

1. Користувач запустив гру
2. Користувач натиснув одну з трьох кнопок : “Start” , “Record List”, “Quit”.
   1. Якщо натиснути “Start” то почнется гра
   2. Якщо натиснути “Record List” користувач побачить таблицю рекордів
   3. Якщо натиснути “Quit” гра закінчиться.

№2 Люди

1. Гра була запушена
2. Люди починають падати з верху
   1. У випадку якщо користувач піймав людину то він отримує бал
   2. У випадку якщо користувач не піймав людину то гра завершиться

№3 Камінь

1. Гра була запушена
2. Камінь починають падати з верху
   1. У випадку якщо користувач піймав людину то він втрачає бал
   2. У випадку якщо користувач втрачає усі бали гра завершиться

№4 Батут

1. Гра була запушена
2. Якщо натиснута одна з кнопок стрілка в право , стрілка в ліво ,то батут рухається
3. Гравець за допомогою батута може піймати об’єкти люди та каміння.

Для точнішої реалізацій програми було розроблено наступні інтерфейси

* 1. Інтерфейси

На основі даних Use Case і User Story та розроблених алгоритмів задач було створено такі інтерфейси :

1. Розроблено інтерфейс меню за допомогою бібліотеки PyQt5 , що має три кнопки : “Start”; “Record List”; “Quit”. Приклад меню на Рис 2.1

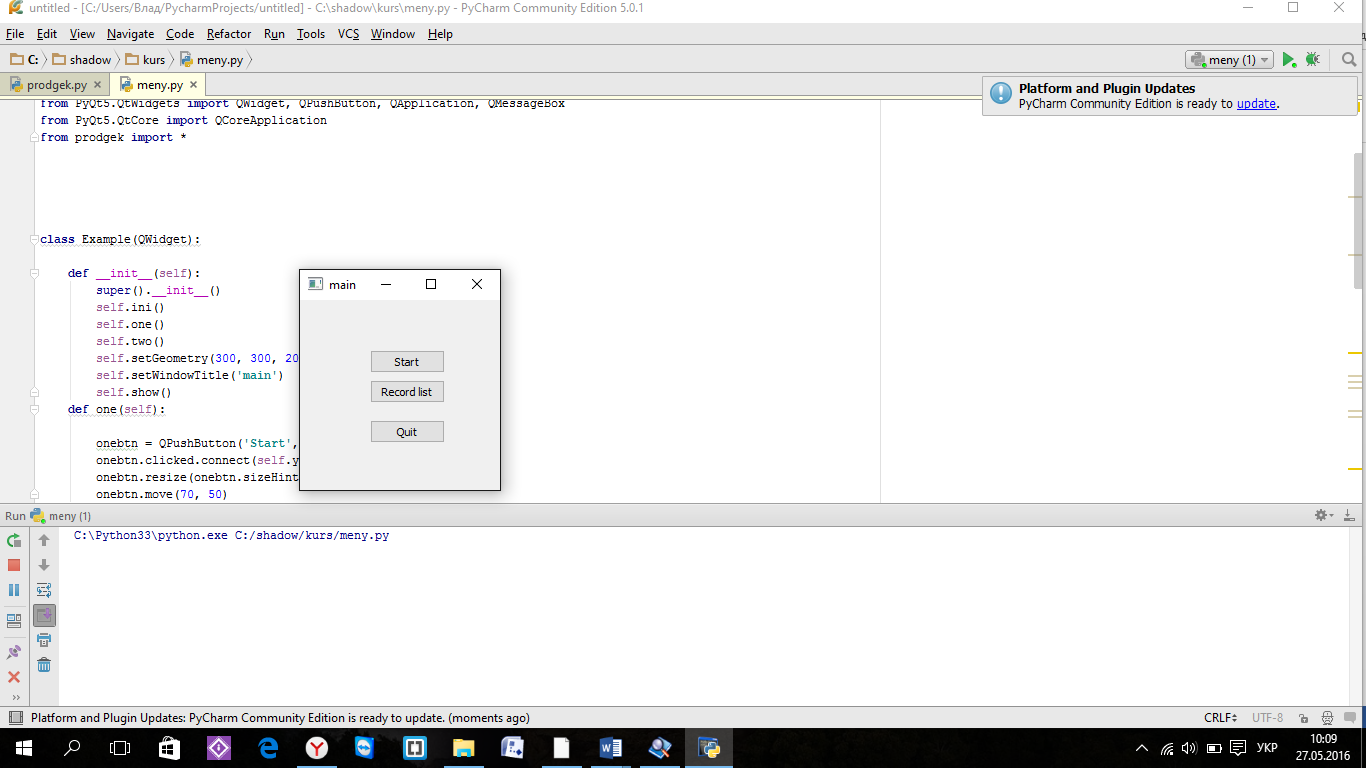


Рис 2.1 Приклад меню

2) Розроблено інтерфейс головного циклу гри , код гри було створено за допомогою бібліотеки Pygame та об’єктно-орієнтованого програмування , активується після натиснення кнопки “Start” в головному меню . Приклад інтерфейсу на Рис 2.2

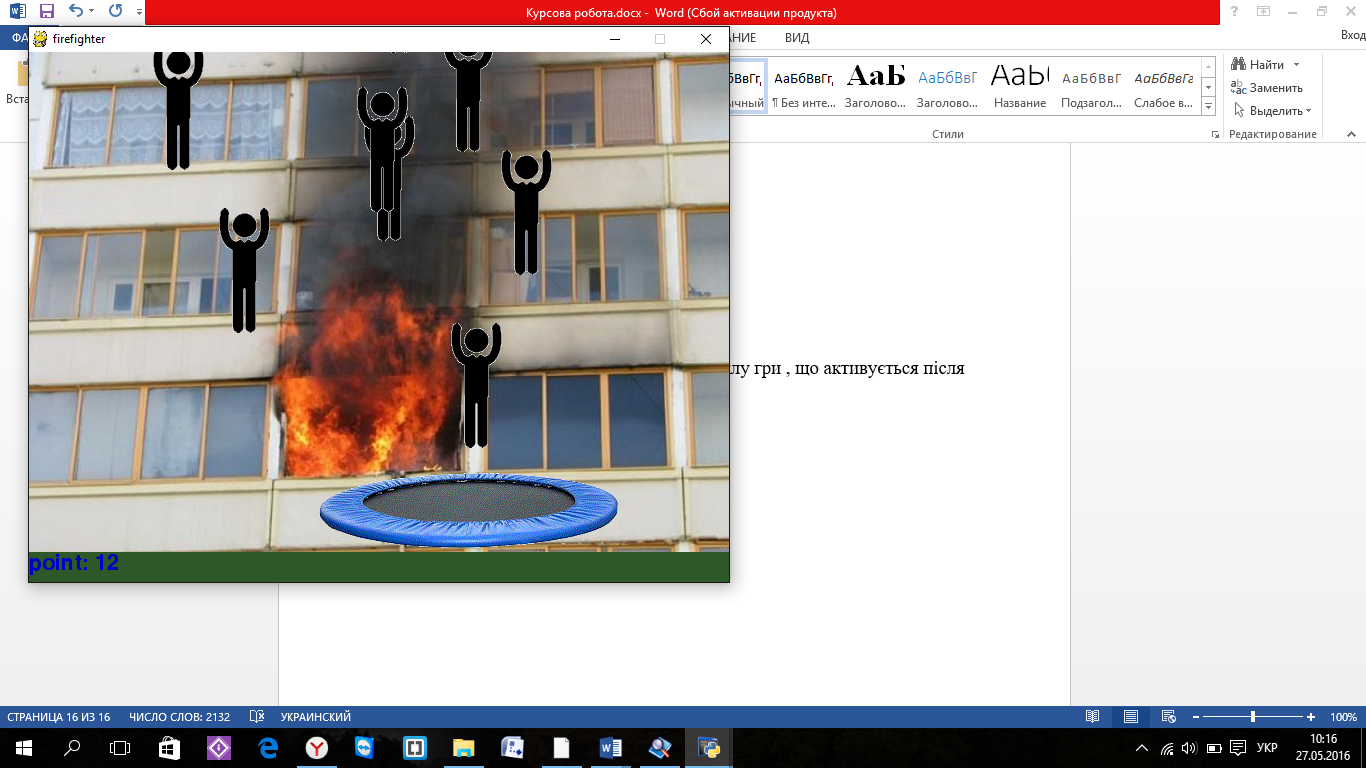


Рис 2.2 Приклад інтерфейсу

3) Розроблено інтерфейс списку результатів з використанням бібліотеки PyQt5. Приклад інтерфейсу на Рис 2.3

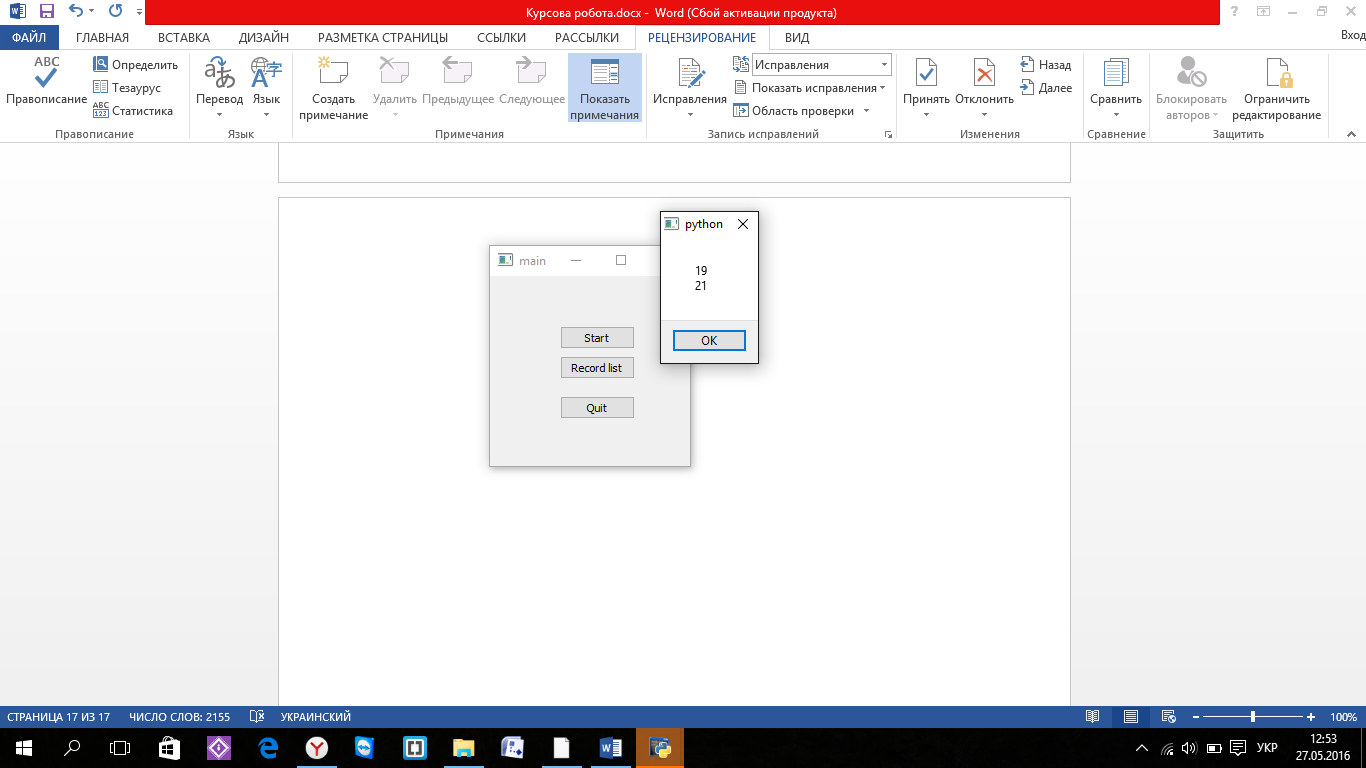


Рис. 2.4 Приклад інтерфейсу

Таким чином виходячи з усього вище викладеного було побудовано наступну архітектуру задачі

# Архітектура задачі

## Дані

В [об'єктно-орієнтованому програмуванні](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), **клас** — це спеціальна конструкція, яка використовується для групування пов'язаних змінних та [функцій](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0). При цьому, згідно з термінологією [ООП](https://uk.wikipedia.org/wiki/%D0%9E%D0%9E%D0%9F), глобальні змінні класу (члени-змінні) називаються *полями даних* (також *властивостями* або *атрибутами*), а члени-функції називають *методами* класу. Створений та ініціалізований екземпляр класу називають *об'єктом* класу. На основі одного класу можна створити безліч *об'єктів*, що відрізнятимуться один від одного своїм *станом* (значеннями полів).Для реалізацій в даному проекті було створено чотири класи : гравець (You); люди(people); камінь(stone); приклад (Example). Класи гравець(You), люди(people), камінь(stone) , що наслідують від класу sprite з бібліотеки PyGama. Класи люди(people) і камінь(stone) відразу після створення заносяться у списки people\_list та stone\_list відповідно . Клас You задається перед головним циклом гри один раз . Клас Example унаслідується від класу QWidget з PyQt бібліотеки . Далі більш наглядно в таблиці 1

Таблиця 1 Класи в програмі

|  |  |  |  |
| --- | --- | --- | --- |
| Клас | Метод/Функція | Графічне зображення | Унаслідуються |
| Example | “Start”;“Record List”; “Quit” | Вікно меню | QWidget |
| You | “left”; “right” | Батут | Sprite |
| people | “update” | Падаючи люди | Sprite |
| stone | “update” | Падаюче каміння | Sprite |

Далі на основі даних та з пункту Проектування задач створимо алгоритми реалізації

## Алгоритми реалізації

1. Алгоритм головний цикл гри на Рис3.1 :



Рис 3.1 головний цикл

1. Алгоритм головного циклу меню на Рис 3.2



Рис 3.2 головного циклу меню

1. Алгоритм циклу гри на Рис 3.3



Рис 3.3 Алгоритм циклу гри

1. Функція руху людей на Рис 3.4



Рис 3.4 Функція руху людей

1. Функція руху каміння на Рис 3.5



Рис 3.5 Функція руху каміння

1. Рух батуту на Рис 3.6



Рис 3.6 Рух батуту

1. Функція right на Рис 3.7



Рис 3.7 Функція right

1. Функція left на Рис 3.8



Рис 3.8 Функція left

Як видно в описі алгоритмів руху відсутні цикли це пов’язано с тим що кожна функція і так знаходиться в головному циклі гри , тому вже виконується циклічно.

В наступному пункті буде показано один з варіантів як все вище показане реалізувати за допомогою коду .a

## Функції модулі.

Для реалізацій даного проекту було необхідно підключити бібліотеки PyGame та PyQt. Далі буде показано можливі методи реалізацій з використанням елементів коду по кожному з об’єктів .

1. Меню представлено у вигляді об’єкту класу Example , що унаслідуються від класу QWidget , що в свою чергу належеть бібліотеці PyQt . В класі задано три кнопки з функціями .Додаток з кодом цього класу А.

Далі задається об’єкт та викликається на Рис 3.9 :

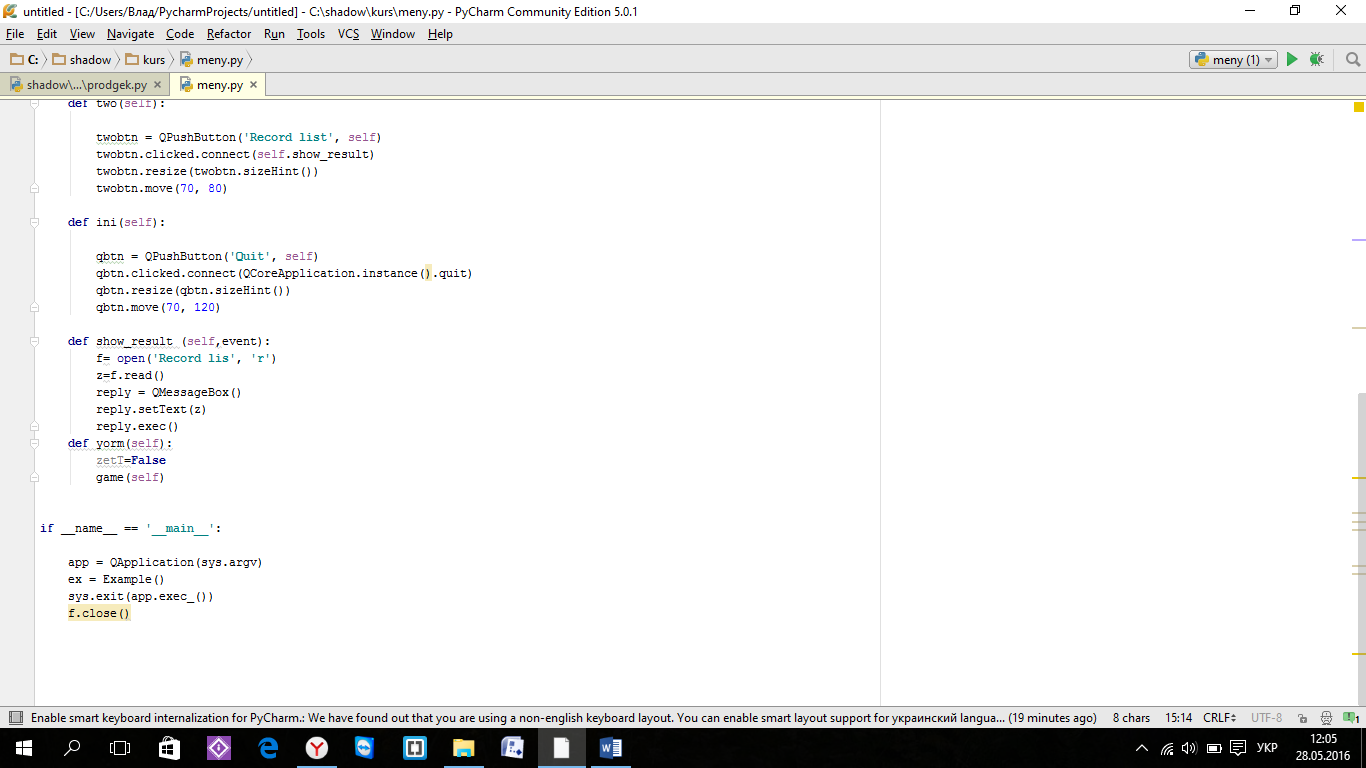


Рис 3.9 задається об’єкт

1. Сам проект був створений за допомогою бібліотеки PyGame та прив’язаний до кнопки меню Start(дивіться додаток №1).Та сам проект в меню передається як функція на Рис 3.10 .

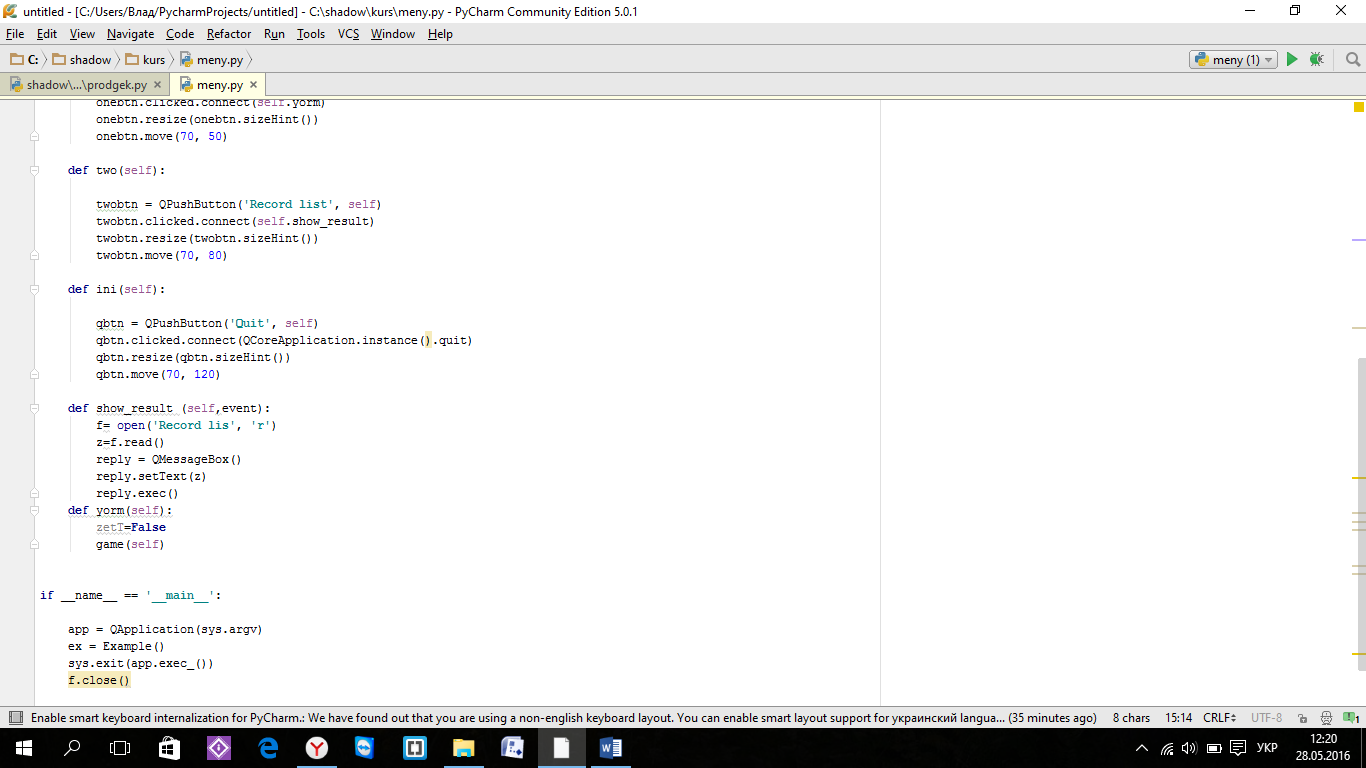


Рис 3.10 функція передається в меню

1. Головний цикл гри представлену додатках Б та В, необхідною до згадки є рядок, що дає змогу не натискати на клавіші по декілька разів, а один раз натиснути і утримувати клавішу для руху : pygame.key.set\_repeat(1,1)

В грі використано три класи що унаслідуються від класу Sprite , який належить до бібліотеки PyGame:

1. Клас “You” описує об’єкт яким буде грати користувач . Методи призначені для цього класу : left ( рух в ліво) ; right (рух в право) на Рис 3.11

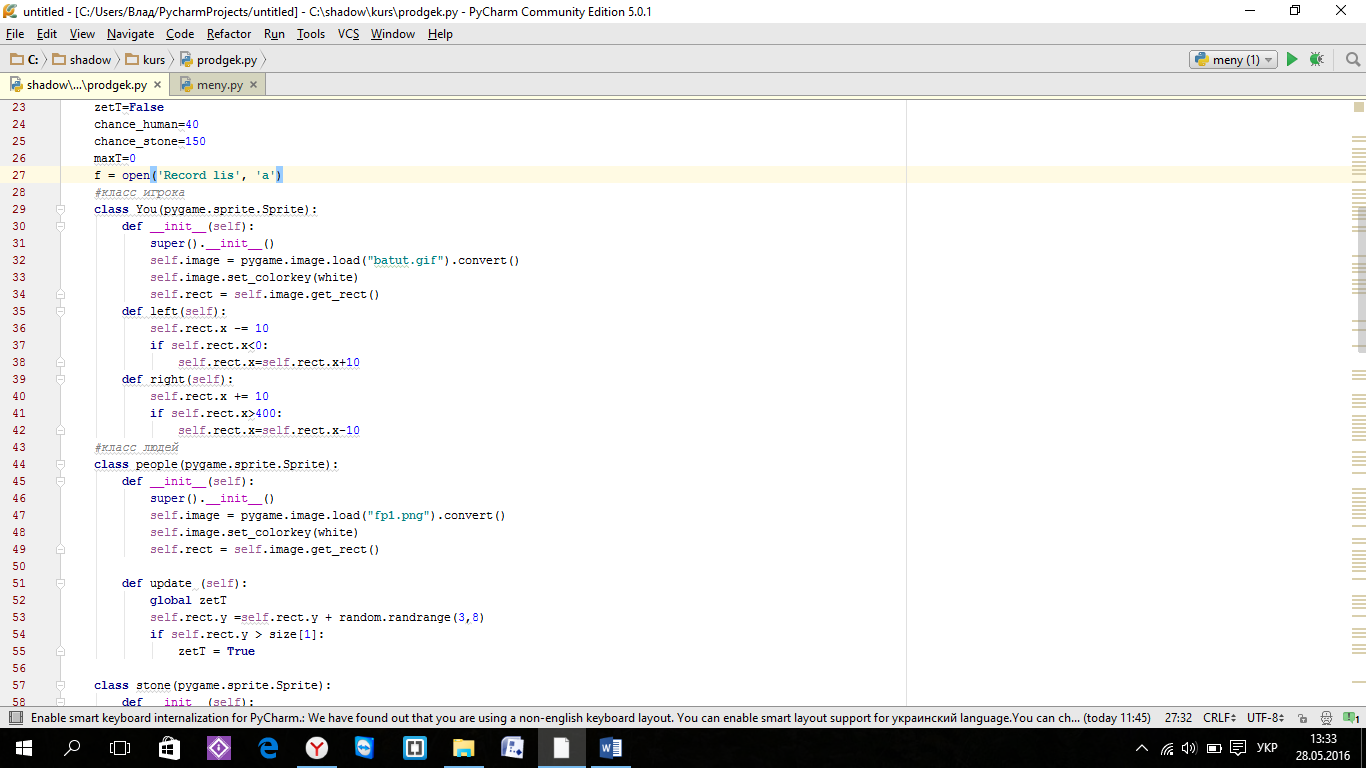


Рис 3.11 Функція руху

1. Клас “people” описує об’єкт який користувач буде ловити. Методи призначені для цього класу : update ( рух об’єкта в низ ) на Рис 3.12

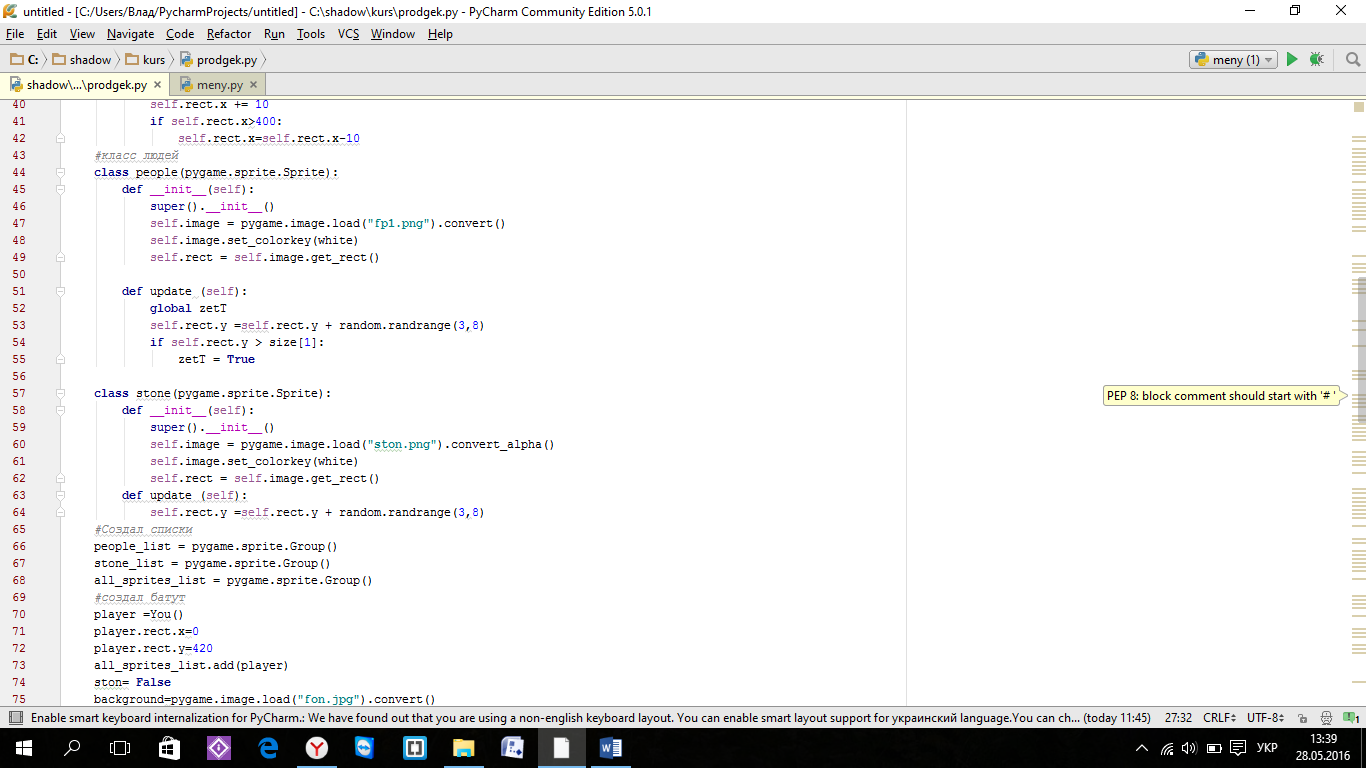


Рис 3.12 клас людей

1. Клас “stone” описує об’єкт який користувач буде ловити. Методи призначені для цього класу : update ( рух об’єкта в низ ) на Рис 3.13

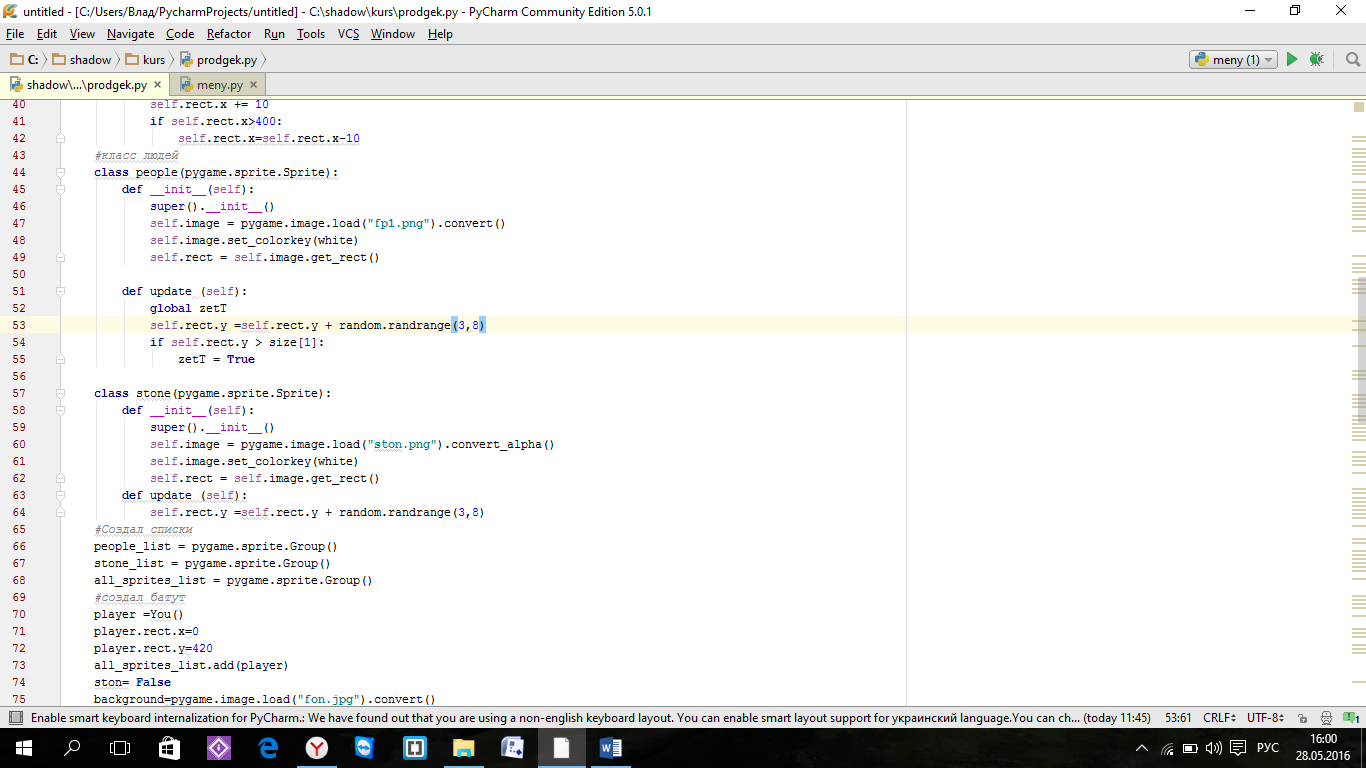


Рис 3.13 клас камні

Загальна розробка гри майже завершена , необхідно тільки провести тестування.

# Тестування

## User Test

Завдання user test полягає в тому щоб виявити помилки в роботи програми , визначити чи досить зрозумілий та зручний інтерфейс програми , перевірити чи виконує програма тещо від було треба , чи виконує те що не треба . Було проведено наступні тести:

1. Стабільна робота програми протягом п’ятнадцяти хвилин . Помилок не виявлено , обсяг використаної пам’яті , навантажень не змінився.
2. Стабільна робота програми протягом години . Виявлено що відбувається незначне зростання пам’яті . Помилку виправлено. Причиною помилки було накопичення об’єктів каміння у списку stone\_list .
3. Припинення роботи , програма нормально закривається як з вікна меню так і с режиму гри.
4. Тестування припинення гри за основним сценарієм програшу (не піймана людина ) . Помилок не виявлено .
5. Тестування припинення гри при додатковому сценарій програшу . Виявлено помилку: гра припинилась але не було відображено відповідного тексту на екрані . Помилка полягала в порушені логіки гри .рис 3.14

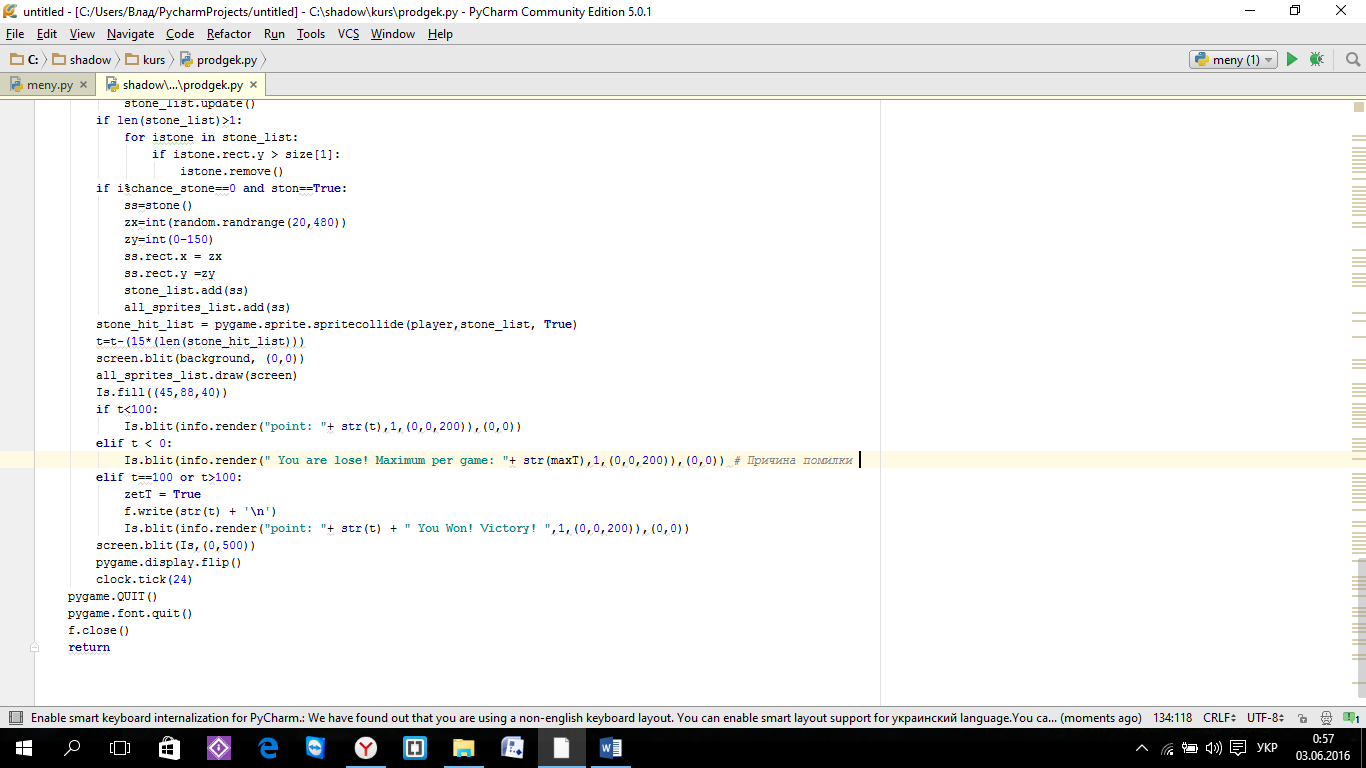


Рис 3.14 помилковий код

Помилку виправлено .рис 3.15

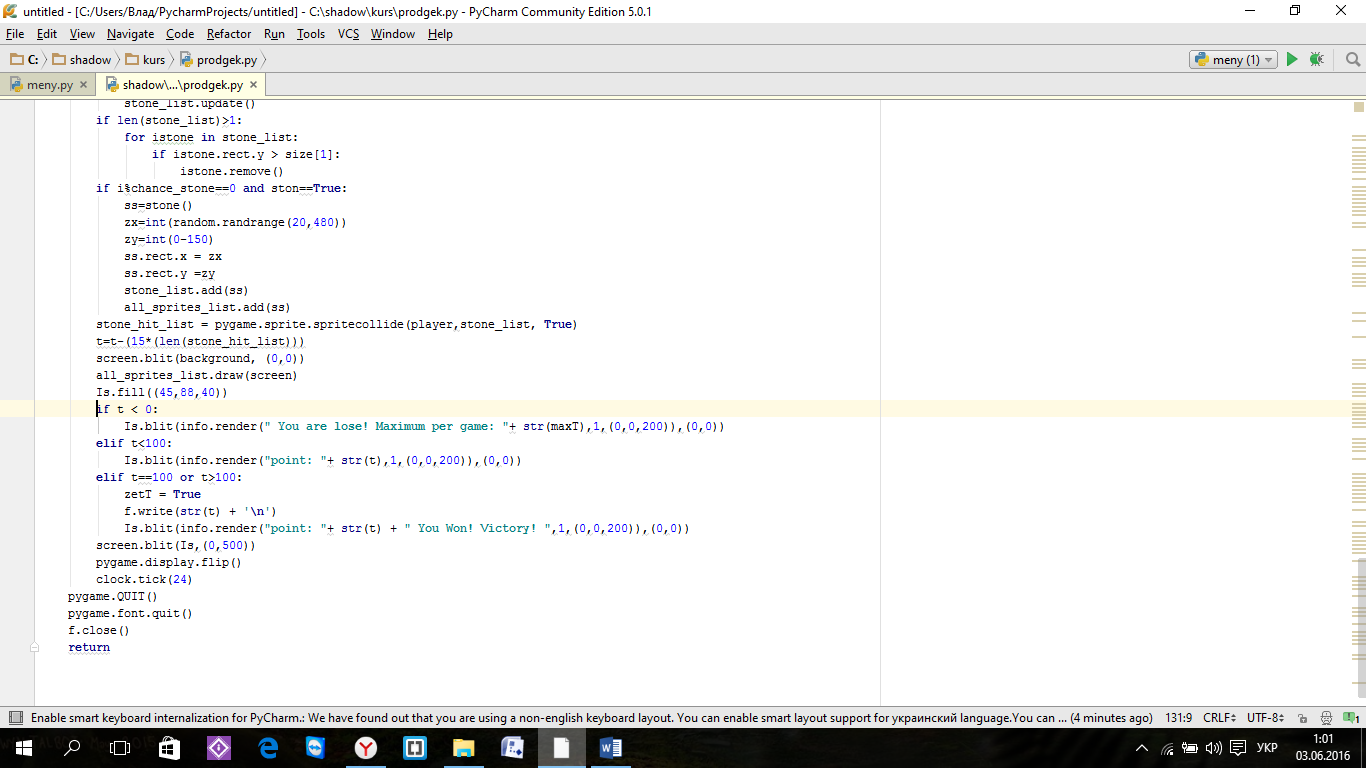


Рис 3.15 правильний код

1. Тестування припинення гри за основним сценарієм перемоги не виявило помилок.

## Інструкція

Використовуючи стрілки ліво – право управляйте батутом та ловіть усіх людей, якщо хоч один впаде ви програєте . Ухиляйтесь від каміння щоб зберегти бали , якщо їх буде менше нуля ви програли . Ви виграйте тільки якщо наберете сто балів.

# Висновки

Даний проект гарно демонструє роботу з об’єктами та класами , розуміння мови Python та взагалі основних принципів програмування . В ньому використовується різні бібліотеки та способи програмування. Програма дуже гарно демонструє навички програмування набуті за перший курс . Що тільки підтверджує цінність даного проекту .

Реалізована гра Пожежники в даному проекті була виконана в найбільш простий, але достатньо раціональний спосіб завдяки використанню інформації що була взята з досліджених аналогів. Конкретніше у створені більш зручного та зрозумілого інтерфейсу.

Для більш точної реалізацій програмного коду та зовнішнього виду програми були розроблені : User Story, Use Case, Алгоритми задачі, Інтерфейси. Це допомогло розробити: Алгоритми реалізацій , функцій та модулі та дані, які вони будуть опрацьовувати. З чого і складалася програма.

Для остаточного завершення програми було проведено декілька User Test-тів що виявили декілька помилок та допомогло виправити їх.

Далі для зручності користувача була написана інструкція .

# 

# Джерела

# 

# Программирование аркадных игр и обучение информатике [електронний ресурс] <http://programarcadegames.com/>

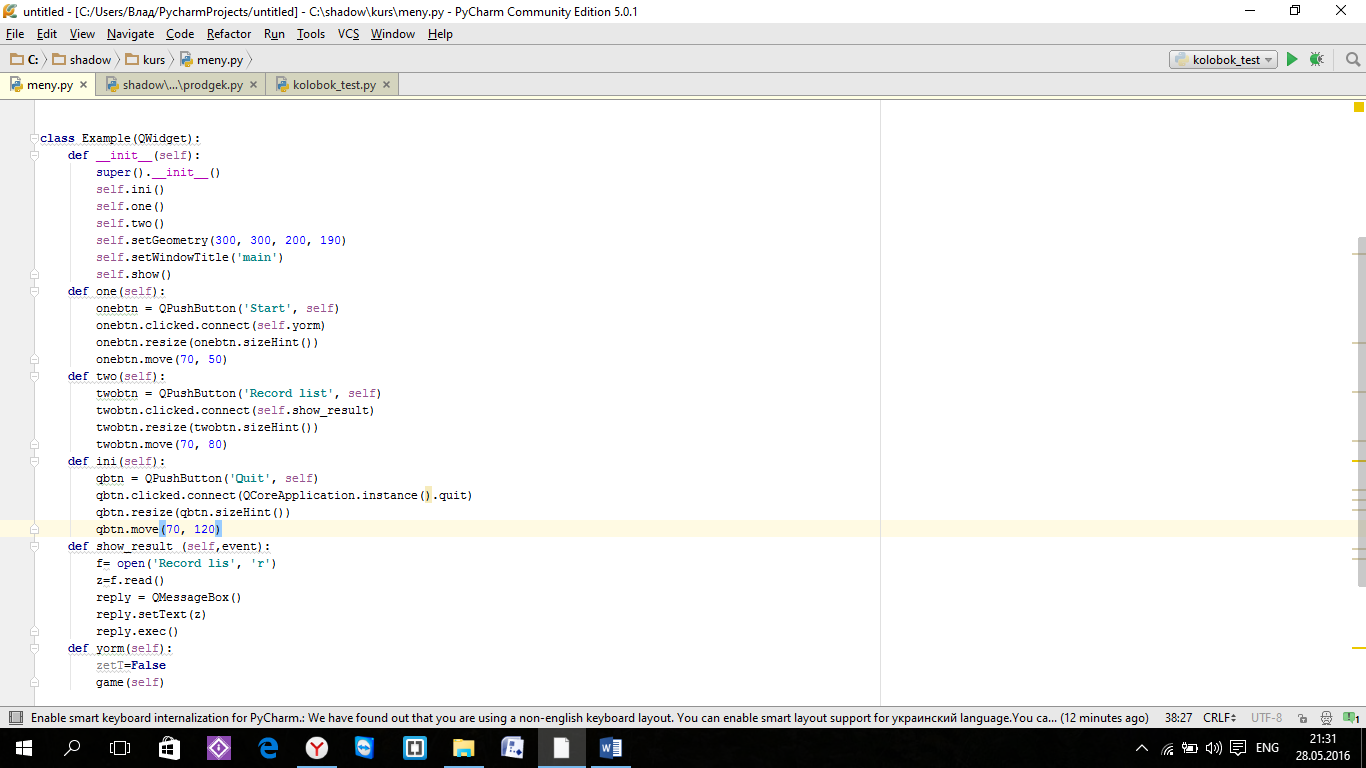
##### pygame documentation [електронний ресурс] <http://www.pygame.org/>

1. Python алгоритмы [електронний ресурс] <http://py-algorithm.blogspot.nl/>
2. Доусон М. Праграммируем на Python – СПб.:Питер,2014-416с.ил.
3. TM Python [електронний ресурс] <https://www.python.org/>
4. Язык программирования Python 3 для начинающих и чайников[електронний ресурс] <http://pythonworld.ru/>
5. Python [електронний ресурс] <http://www.cyberforum.ru/>
6. Прохорецок Н.А. Python 3 и PyQt. Разработка приложений : СПб.:БХВ-Петербург,2012.-704 с.:
7. SUBSCRIBE.RU[електронний ресурс] <http://old.subscribe.ru/>
8. Python -3[електронний ресурс] http://python-3.ru/

# Додатки

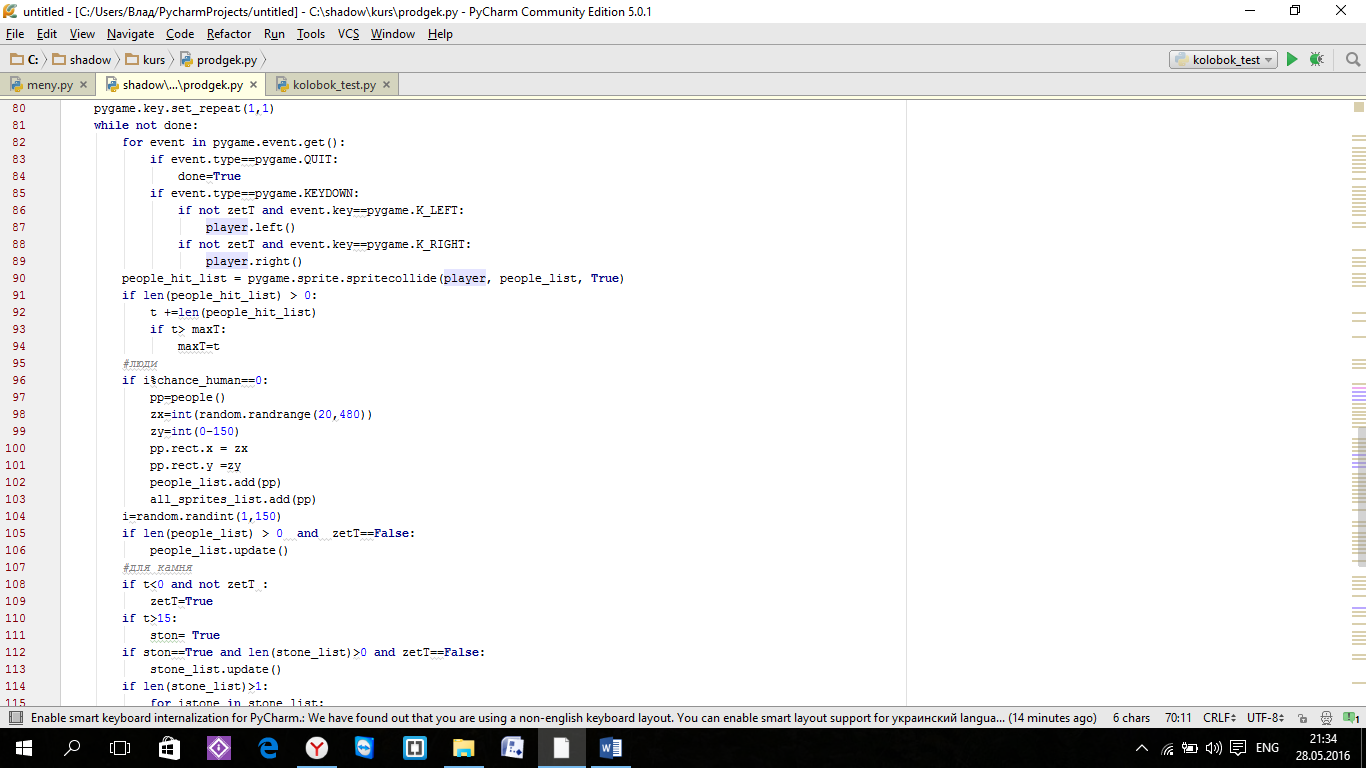
Додаток А

код меню



Додаток Б

Перша частина коду гри



Додаток В

Друга частина коду гри

