*Міністерство освіти і науки України*

*Коломийський політехнічний коледж*

*Національного університету «Львівська політехніка»*

***Спеціальність*** *121 «Інженерія програмного забезпечення»*

*Дипломний проєкт*

*на тему: «Візуальна новела на движку Ren`Py» у ТзОВ «БЕСТ-РАН КОНСАЛТИНГ», м. Коломия*

*Пояснювальна записка на 69 стор.*

*Виконала: студентка 4 курсу, групи П-41*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Сайко Г.Т.)*

*Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_(Дуб Т.Т.)*

*Консультант економічної*

*частини \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Смиковчук Т.В. )*

*Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ляшеник А.В.)*

*Заступник директора*

*з навчальної роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Смиковчук Т.В.)*

*Коломия – 2020 року*

**КОЛОМИЙСЬКИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ**

**НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Відділення | | |  | |
| Циклова комісія | | | *інженерії програмного забезпечення* | |
| Освітньо-кваліфікаційний рівень | | | | *молодший спеціаліст* |
| Галузь знань | *12 «Інформаційні технології»* | | | |
| Спеціальність | | *121 «Інженерія програмного забезпечення»* | | |

**ЗАТВЕРДЖУЮ**

Голова циклової комісії  
інженерії програмного забезпечення

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**/ \_\_\_\_\_\_\_\_/

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_року

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **З А В Д А Н Н Я** | | | | | | | | |
| **НА ДИПЛОМНИЙ ПРОЕКТ** | | | | | | | | |
| студенту | *Сайко Галині Тарасівній* | | | | | | | |
| 1. Тема проекту: | | | *«Візуальна новела на движку Ren`Py» у ТзОВ «БЕСТ-РАН КОНСАЛТИНГ», м. Коломия* | | | | | |
|  | | |  | | | | | |
| керівник | | Дуб Тетяна Теодозіївна | | | | | | |
| Затверджено наказом по коледжу від | | | |  |  | | 20\_ р. | № |
| 1. Термін подання студентом закінченої роботи | | | | | |  | | |
| 1. Зміст розрахунково-пояснювальної записки: | | | | | |  | | |

1. Опис предметного середовища. 2. Огляд наявних аналогів.

3. Постановка задачі. 4. Аналіз предметної області. 5. Проектування системи. 6. Проектування UML-діаграми. 7. Побудова об’єктно-орієнтрованої моделі. 8. Засоби розробки. 9. Опис програмної реалізації. 10. Маркетинговий аналіз розробки та впровадження програмного продукту. 11. Економічне обґрунтування доцільності розробки та впровадження програмного продукту.

4. Консультанти розділів проекту

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| 7.Дата видачі завдання | 17.04.2020 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів  дипломного проекту | Строк  виконання етапів проекту | Примітка |
| 1 | Вибір теми, подання заяви на затвердження теми дипломного проекту | 03.04.2020 |  |
| 2 | Отримання завдання, аналіз літературних джерел за метою роботи, робота над планом | 17.04.2020 |  |
| 3 | Аналіз отриманої задачі, розробка алгоритму розв’язку | 25.04.2020 |  |
| 4 | Розробка та налагодження програмного продукту, проведення тестових розрахунків | 25.05.2020 |  |
| 5 | Розрахунок економічних показників | 13.06.2020 |  |
| 6 | Оформлення пояснювальної записки | 13.06.2020 |  |
| 7 | Допуск до захисту дипломного проекту | 20.06.2020 |  |
| 8 | Попередній захист дипломного проекту | 21.06.2020 |  |
| 9 | Доопрацювання та оформлення документів для захисту дипломного проекту | 24.06.2020 |  |
| 10 | Захист дипломного проекту | 25.06.2020 |  |

**Студент**  \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

( підпис ) (прізвище та ініціали)

**Керівник проекту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_

( підпис ) (прізвище та ініціали)

**ЗМІСТ**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

412 ДП 12.121.05.01

Розроб.

Сайко Г.Т.

Перевір.

Реценз.

Н. Контр.

Затверд.

*«Візуальна новела на движку Ren’Py» у ТзОВ «БЕСТ-РАН КОНСАЛТИНГ», м. Коломия*

Літ.

Аркушів

*69*

КПК НУ «ЛП»

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**……………………………….………..…5

**ВСТУП**…………………………………………………………………..………..…6

**1 ЗАГАЛЬНІ ПОЛОЖЕННЯ**

1.1 Опис предметного середовища…………………………………………7

1.2 Огляд наявних аналогів…………………………………………………8

1.3 Постановка задачі……………………………………………………….10

Висновки до розділу……………………………………………………..…11

**2 ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ**

2.1 Аналіз предметної області………………………………………………12

2.2 Проектування системи…………………………………………………14

2.2.1 Проектування UML-діаграми…………………………………17

2.2.2 Побудова об’єктно-орієнтрованої моделі……………………21

Висновки до розділу…………………………………………………….…23

**3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ**

3.1 Засоби розробки…………………………………………………………24

3.2 Опис програмної реалізації……………………………………………25

Висновки до розділу……………………………………………….………38

**4 ЕКОНОМІЧНА ЧАСТИНА**

4.1 Маркетинговий аналіз розробки та впровадження програмного продукту…………………………………………………………………..…39

4.2 Економічне обґрунтування доцільності розробки та впровадження програмного продукту. ………………………………………………..……43

Висновки до розділу……………….……………………………………..…49

**ЗАГАЛЬНІ ВИСНОВКИ**…………………………….…………………………..50

**ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА**….……………………………………51

**ДОДАТКИ**…………………………………………………………………………52

**Додаток А**………………………………………………………………..…52

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

БД – база даних

СУБД – система управління базою даних

ПЗ – програмне забезпечення

ПО – предметна область

ПП – програмний продукт

ER – Entity Relation

ООП – об’єктно-орієнтоване програмування

**ВСТУП**

Можна не любити відеоігри та не розуміти їх популярності, але заперечувати, що це одна з найбільш успішних індустрій у час технологічного розвитку — вже просто неможливо.

Унікальний досвід, який можуть подарувати лише ігри, — це пряма взаємодія з вашим ціннісним компасом. Після хорошої книги ви можете задуматися про те, яка ви людина, і як би ви вчинили в тій чи іншій ситуації. Хороші ігри змушують не лише думати, а й буквально вирішувати це постійно.

Незважаючи на те, що ринок цифрових розваг впритул наблизився, а деінде навіть випередив ринок кіно. Що технології, створені для геймерів, почали використовувати в Голлівуді. Що відеоігри нагороджують Гільдія сценаристів США та BAFTA. Масова аудиторія все одно продовжує вважати, що ігри — це лише одноклітинні стрілялки.

І хоча пройшло вже двадцять років, велика кількість людей, особливо в пострадянських країнах, продовжує вірити, що це правда. І що відеоігри — це пуста трата часу.

Відеоігри створюють із тією ж якістю та продуманістю, що й інші медіа. Вони можуть висловлювати думки тієї ж глибини. І вони дарують унікальний досвід, співмірний хіба що з участю у перформансах. Відеоігри — це мистецтво.

**1 ЗАГАЛЬНІ ПОЛОЖЕННЯ**

* 1. **Опис предметного середовища**

Предметною областю даної роботи є розробка візуальної новели на движку Ren'Py.

Візуальна новела – жанр відеоігр, в якому гравцеві демонструється історія за допомогою виводу на екран статичних зображень, тексту і звуків.

Велика частина візуальних новел має розгалужений сюжет і кілька варіантів кінцівки, у такому випадку гравцеві в певні моменти пропонується зробити вибір між різними варіантами дій чи відповідей у діалозі, таким чином визначаючи подальші події у грі.

Стиль оповіді у візуальних новелах дещо відрізняється від друкованих книг. Здебільшого розповідь ведеться від першої особи, і спостерігачем всіх подій є головний герой.

Багато людей, не залежно від віку, люблять читати. А ще така ж кількість, люблять грати в відеоігри. І жанр ігор Візуальна новела, містить в собі і те і інше, що не може не радувати.

Напевно не раз людина читаючи якусь книгу, задумувалась над тим, що в якійсь певній ситуації вчинила б інакше, не так як головний герой, і даний жанр ігор дозволяє це.

**1.2 Огляд наявних аналогів**

Перш ніж почати розробляти певну систему, потрібно спочатку здійснити пошук аналогів, дослідити, проаналізувати їх переваги та недоліки, щоб зрозуміти, яких помилок потрібно уникнути та які їх переваги доцільно реалізувати. Було проаналізовано візуальну новелу «Doki Doki Literature Club!».

Doki Doki Literature Club! — психологічний хоррор,візуальна новелла, розроблений командою «Team Salvato». Випущена 22 вересня 2017 року для Windows, MacOS і Linux, а пізніше, 6 жовтня, і для Steam. Гра має, в основному, лінійний сюжет, альтернативні сцени і кінцівки, що залежать від вибору гравця. Хоча на перший погляд гра здається безтурботною, насправді це психологічний хорор, який передбачає інтенсивне використання техніки руйнування четвертої стіни.

У перші три місяці після випуску, Doki Doki Literature Club! було завантажено більше мільйона разів.

За аналогією з іншими представниками жанру візуальних романів ігровий процес Doki Doki Literature Club! відрізняється низьким рівнем інтерактивності і складається зі сцен зі статичними двовимірними зображеннями персонажів в перспективі від першої особи. Оповідання та діалоги подаються у вигляді супутнього персонажам тексту. Оповідання ведеться від імені головного героя твору - члена літературного клубу, в який його запросила подруга дитинства Сайорі.

У певні розробниками моменти відтворення тексту припиняється, і гравцеві пропонується зробити вибір з кількох варіантів дій, що визначають подальший перебіг подій, - подібні рішення впливають на розвиток відносин головного героя з ключовими жіночими персонажами (Сайорі, Юрі, Нацукі і Монікою).

Також вплив на симпатію персонажів до протагоністу надають результати міні-ігри, в якій гравцеві пропонується скласти вірш з окремих слів. Кожне з ужитих виразів збільшує прихильність певної дівчини до головного героя в залежності від її переваг, а також може викликати негативну реакцію у інших дівчат, яка відображається на поведінці тібі-зображень героїнь в екрані міні-ігри. За результатами міні-ігор гравцеві можуть відкриватися додаткові сцени з тієї героїнею, якою було присвячено вірш.

Візуальна новела розділена на три стадії проходження, в ході яких гравцеві необхідно тричі почати гру заново, при цьому кожна зі стадій гри має власний сюжет. В ході ігрового процесу на вимогу гри гравець змушений звертатися до файлів візуального роману для подальшого просування оповіді.

На рисунках 1.1 і 1.2 демонструються геловне меню гри та вигляд гри в процесі проходження.



Рис. 1.1 - Головне меню гри



Рис. 1.2 - Вигляд гри в процесі проходження

**1.3 Постановка задачі**

Були складені вимоги до створення програмного продукту:

1. Автоматичне збереження гри;

2. Відкат, для повернення раніше показаного екрану;

3. Можливість пропускати текст при відтворенні, в тому числі можливість показувати лише текст, який не був показаний раніше;

4. Автоматичну прокрутку тексту без використання клавіатури, що може бути зручно при великих обсягах тексту;

5. Можливість приховувати текст так, що користувач може бачити зображення за ним;

6. Запис вибору гравця і вихід на певну кінцівку;

7. Введення ім’я гравця на початку гри;

8. Можливість використання бази даних;

9. Вибір дії за допомогою кліку (клікабельна карта);

10. Міні-ігри.

**Висновки до розділу**

У даному розділі був проведений опис предметного середовища, який мав на меті пояснити доцільність розробки програмного забезпечення, розробленого за час дипломного проекту.

Також в розділі був наведений огляд наявних аналогів, що дозволяє проаналізувати і порівняти аналоги з дипломним програмним продуктом.

В кінці розділу була наведена постановка задачі з основними вимогами до програмного продукту.

**2 ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ**

**2.1 Аналіз предметної області**

Предметною областю даної роботи є розробка візуальної новели на движку Ren'Py.

Основною вимогою до створення є написання програми на мові програмування Python. Дана візуальна новела повинна бути досить простою та легкою в розумінні.

Візуальна новела – жанр відеоігр, в якому гравцеві демонструється історія за допомогою виводу на екран статичних зображень, тексту і звуків.

Велика частина візуальних новел має розгалужений сюжет і кілька варіантів кінцівки, у такому випадку гравцеві в певні моменти пропонується зробити вибір між різними варіантами дій чи відповідей у діалозі, таким чином визначаючи подальші події у грі.

Стиль оповіді у візуальних новелах дещо відрізняється від друкованих книг. Здебільшого розповідь ведеться від першої особи, і спостерігачем всіх подій є головний герой.

Ren'Py - безкоштовна, вільна і відкрита платформа для створення візуальних новел.

В можливості двигуна входить створення розгалужених діалогових потоків, збереження та відкати до потрібних точок сценарію, різні варіації переходів між сценами, створення різноманітних меню вибору, DLC та інше. Є можливість відігравати відео як в повноекранному режимі, так і в формі анімованих спрайтів. Для опису потрібних анімацій використовується описова мова ATL (Animation and Translation Language). Також є можливість анімувати керівні елементи графічного інтерфейсу.

Скриптова система в Ren'Py максимально спрощена, та використовується як послідовний опис подій в грі. Є можливість вставляти блоки Python-коду в сценарний потік, що значно збільшує потенційні можливості в ігровій сцені.

Стандартно, рушій підтримує Windows, останні версії Mac OS X та Linux, а також може бути запущений на Android та iOS 7+.

Але через свою відкритість та мінімальне використання сторонніх бібліотек може бути портований практично на будь-яку платформу, яка підтримує Python та SDL.

**2.2 Проектування системи**

Багато людей, не залежно від віку, люблять читати. А ще така ж кількість, люблять грати в відеоігри. І жанр ігор Візуальна новела, містить в собі і те і інше, що не може не радувати.

Напевно не раз людина читаючи якусь книгу, задумувалась над тим, що в якійсь певній ситуації вчинила б інакше, не так як головний герой, і даний жанр ігор дозволяє це.

Модель методології розробки програмної системи

При розробці дипломної роботи була використана спіральна модель життєвого циклу.

На рисунку 2.1 демонструється спіральна модель життєвого циклу

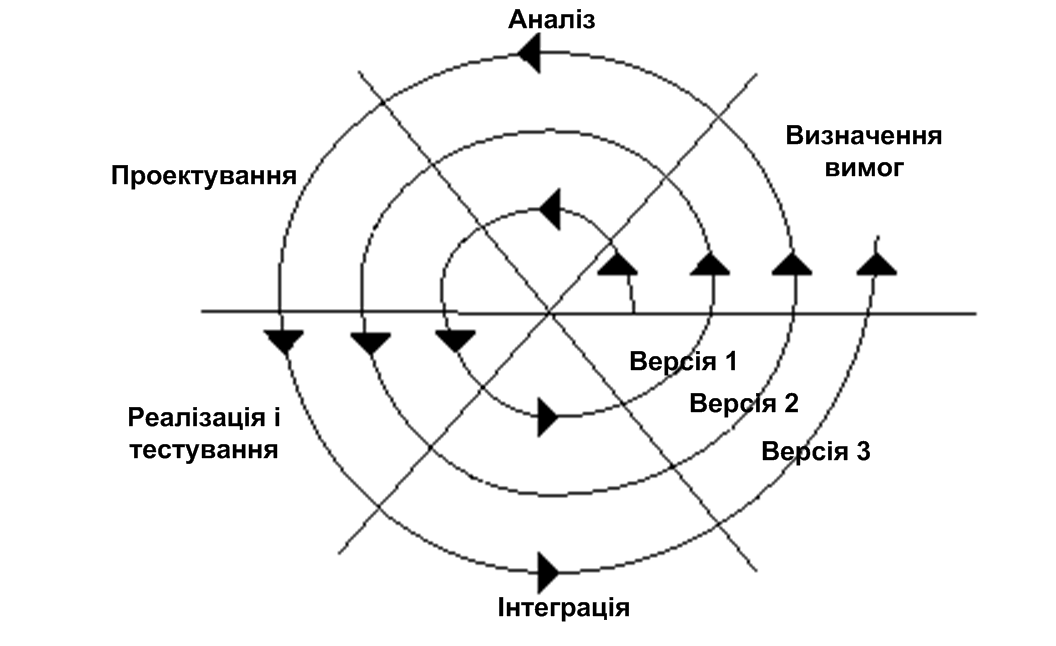


Рис. 2.1 - Спіральна модель життєвого циклу

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При ітеративному способі розробки відсутню роботу можна буде виконати на наступній ітерації. Головне ж завдання - щонайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

Виходячи з можливості внесення змін, як в процес, так і в проміжний продукт було створено спіральну модель ЖЦ.

Внесення змін орієнтоване на задоволення потреби користувачів одразу, як тільки буде встановлено, що створені артефакти або елементи документації не відповідають дійсному стану розробки.

Дана модель ЖЦ допускає аналіз продукту на витку розробки, його перевірку, оцінку правильності та прийняття рішення про перехід на наступний виток або повернення на попередній виток для доопрацювання на ньому проміжного продукту.

В спіральній моделі є свої переваги і недоліки:

Переваги спіральної моделі

• Cпрощується внесення змін в проект при зміні вимог замовника;

• Ітераційний підхід спрощує повторне використання компонентів дозволяє використовувати компонентний підхід до програмування. Це обумовлено тим. що набагато простіше виявити (ідентифікувати) загальні частини проекту, коли вони вже частково розроблені, ніж намагатися виділити їх на самому початку проекту;

• Cпіральна модель дозволяє отримати більш надійну і стійку систему;

• Іераційний підхід дозволяє удосконалювати процес розробки - аналіз, проведений в кінці кожної ітерації, дозволяє проводити оцінку того, що має бути змінено в організації розробки, і поліпшити її на наступній ітерації.

• Окремі елементи інформаційної системи інтегруються в єдине ціле поступово. Інтеграція проводиться фактично безперервно. Оскільки інтеграція Опочинається з меншої кількості елементів, то виникає набагато менше проблем при її проведенні (за деякими оцінками, при використанні каскадної моделі розробки інтеграція займає до 40% всіх витрат в кінці проекту);

• Зменшення рівня ризиків. Дана перевага є наслідком попереднього, так як ризики виявляються саме під час інтеграції. Тому рівень ризиків максимальний на початку розробки проекту. У міру просування розробки очікуваний ризик зменшується;

• Ітераційна розробка забезпечує більшу гнучкість в управлінні проектом, даючи можливість внесення тактичних змін до розроблювальне виріб.

Проблеми, що виникають при використанні спіральної моделі:

Основна проблема спірального циклу - визначення моменту переходу на наступний етап. Для її вирішення необхідно ввести тимчасові обмеження на кожен з етапів життєвого циклу. Інакше процес розробки може припиниться в нескінченне вдосконалення вже зробленого. При ітераційному підході корисно слідувати принципу "краще - ворог хорошого". Тому завершення ітерації повинно проводитися строго відповідно до плану, навіть якщо не вся запланована робота закінчена. Планування робіт зазвичай проводиться на основі статистичних даних, отриманих у попередніх проектах, і особистого досвіду розробників.

**2.2.1 Проектування UML-діаграми**

Метод UML (Unified Modeling Language — уніфікована мова моделювання) є рідкісним прикладом плідної кооперації групи (G. Booch, I. Jacobson, J. Rumbaugh) провідних спеціалістів з програмної інженерії і авторів відповідних методів інженерії вимог, що набули значного визнання і широко застосовуються. Дослідивши всі переваги власних пропозицій і широкий спектр конкуруючих, вони інтегрували свої зусилля, створивши новий метод моделювання, якому дали цитовану вище назву UML. UML став базовим для багатьох провідних розробників програмного забезпечення, і тепер експерти прогнозують, що він набуде статусу міжнародного стандарту як метод моделювання продуктів усіх стадій життєвого циклу розробки програмних систем.

Автори визначають свій метод як мову для специфікації, візуалізації, конструювання й документування *артефактів (artifacts)* програмних систем, а також для моделювання бізнесу.

В основу методу покладено парадигму об’єктного підходу, за якою концептуальне моделювання проблеми відбувається в термінах взаємодії об’єктів:

- онтологія домену визначає склад класів об’єктів домену, їхніх атрибутів та взаємовідношень, а також послуг (операцій), які можуть виконувати об’єкти класів;

- модель поведінки визначає можливі стани об’єктів, інциденти, що ініціюють переходи з одного стану в інший, повідомлення, які об’єкти надсилають одне одному;

- модель процесів визначає дії, які виконують об’єкти.

Як і в попередньо розглянутих методах, автори UML декларують, що, враховуючи складність проблеми концептуального моделювання, її не можна розв’язати єдиною нотацією. Концептуальна модель вимог пропонується як сукупність нотацій, переважно діаграм, котрі є візуалізацією подання основних елементів системи в моделі. Кожна з діаграм демонструє певну підмножину інформації, яка деталізує елементи, що являють собою певний аспект опису моделі та його семантику.

В комплексі сукупність включених до методу діаграм відображає найважливіші випадки функціонування системи.

Елементи UML

Діаграма випадків використання

Діаграми випадків використання описують взаємозв’язки і залежності між групою випадків використання і акторами, що беруть участь у процесі Важливо зауважити, що діаграми випадків використання не призначено для показу компонування, вони не можуть описати внутрішню структуру системи. Діаграми випадків використання призначено для полегшення обміну інформацією між майбутніми користувачами системи і замовником, вони особливо корисні для визначення переліку можливостей, які повинна мати система. За діаграмами випадків використання можна сказати, що система має робити, але не те, як вона досягає потрібних результатів, для останнього ці діаграми просто не придатні.

Випадок використання визначає, з точки зору акторів (користувачів), групу дій у системі, які призводять до конкретного видимого результату. Випадки використання є описом типових елементів взаємодії користувачів системи з самою системою. Вони відповідають зовнішньому інтерфейсу системи і визначають форму вимог до того, що має робити система (зауважте, лише «що», а не «як»).

Під час роботи з випадками використання важливо пам’ятати декілька простих правил:

* Кожен випадок використання має бути пов’язано принаймні з одним актором
* У кожного з випадків використання має бути ініціатор (актор)
* Кожен з випадків використання має призводити до відповідного результату (результату з «комерційним значенням»)

Випадки використання можуть мати зв’язки з іншими випадками використання. Ось три найпоширеніших зв’язки між випадками використання:

* «включення», яке вказує на те, що випадок використання відбувається всередині іншого випадку використання
* «розширення», яке означає, що у певних випадках або у певній точці (яку називають точкою розширення) випадок використання буде розширено іншим випадком використання.
* Узагальнення, за якого випадок використання успадковує характеристики випадку використання “вищого рангу”, при цьому можливе перевизначення деяких з характеристик у спосіб, подібний до успадкування між класами.

Актор — це зовнішній чинник (поза межами системи), який взаємодіє з системою шляхом участі (і часто ініціювання) у випадку використання. Акторами, на практиці, можуть бути звичайні люди (наприклад, користувачі системи), інші комп’ютерні системи або зовнішні події.

Акторам відповідають не *реальні* люди або системи, а лише їх *ролі*. Це означає, що коли особа у різний спосіб взаємодіє з системою (виконуючи різні ролі), їй відповідають декілька акторів. Наприклад, особа, яка виконує підтримку користувачів телефоном і приймає замовлення від користувачів до системи, може бути показано актором «Персонал служби підтримки» і актором «Відповідальний за продажі».

Опис випадків використання

Описи випадків використання — це текстові примітки до випадків використання. Зазвичай, вони мають форму нотаток або документа, який певним чином пов’язано з випадком використання, і який пояснює процеси або дії, які відбуваються під час випадку використання.

На діаграмі використання, на рисунку 2.2 в зрозумілій формі демонструється робота додатку.

Діаграма прецедентів (варіантів-використання) UseCase

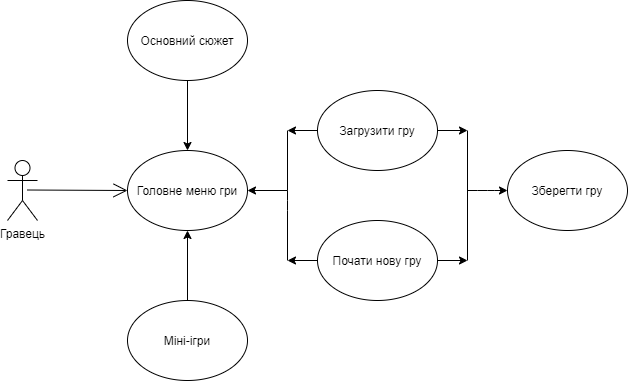


Рис. 2.2 – Діаграма варіантів використання

**2.2.2 Побудова об’єктно-орієнтрованої моделі**

**Діаграма класів**

На діаграмах класів буде показано різноманітні класи, які утворюють систему і їх взаємозв’язки. Діаграми класів називають «статичними діаграмами», оскільки на них показано класи разом з методами і атрибутами, а також статичний взаємозв’язок між ними: те, яким класам «відомо» про існування яких класів, і те, які класи є «частиною» інших класів, — але не показано методи, які при цьому викликаються.

Діаграма класів:

Клас Section предомонстрований в таблиці 2.1.

|  |
| --- |
| **Section** |
| def \_\_init\_\_(self, title) |
| Section(\_("Сюжет"))  Syuzhet("osn\_syuzhet", \_("Основний сюжет"))    Section(\_("Міні-ігри"))  MiniIgry("minigameknp", \_("Камінь-ножиці-папір"))  MiniIgry("minigame2", \_("Швидко натискай на потрібні кнопки")) |

Таблиця 2.1 – структура класу Section

Клас Syuzhet продемонстрований в таблиці 2.2.

|  |
| --- |
| **Syuzhet** |
| def \_\_init\_\_(self, label, title, move=True) |

Таблиця 2.2 – структура класу Syuzhet

Клас MiniIgry продемонстрований в таблиці 2.3

|  |
| --- |
| **MiniIgry** |
| def \_\_init\_\_(self, label, title, move=True) |

Таблиця 2.3 – структура класу MiniIgry

Клас Sections продемонстрований в таблиці 2.4

|  |
| --- |
| **Sections** |
| def \_\_init\_\_(self, title) |
| Sections(\_("Меню")) |

Таблиця 2.4 – структура класу Sections

Клас Stolova продемонстрований в таблиці 2.5

|  |
| --- |
| **Stolova** |
| def \_\_init\_\_(self, label, title, move = True) |
| Stolova("menu\_makarony", \_("Макарони - 8грн"))  Stolova("menu\_pure", \_("Картошка пюре - 8грн"))  Stolova("menu\_vidbyvna", \_("Відбивна - 10грн"))  Stolova("menu\_kotleta", \_("Котлета - 10грн"))  Stolova("menu\_hlib", \_("Хліб - 0.50грн"))  Stolova("exit\_stolova", \_("Вихід")) |

Таблиця 2.5 – структура класу Stolova

**Висновки до розділу**

У другому розділі був проведений аналіз предметної області, в якому було проаналізовано та пояснено що таке візуальна новела, розказано про рушій Ren'Py. Наведено модель методології розробки програмної системи.

Також в розділі було розказано про діаграму варіантів використання, опис випадків використання та проведено проектування UML-діаграми.

В кінці розділу було виконано побудову об’єктно-орієнтованої моделі показано різноманітні класи, які утворюють систему і їх взаємозв’язки.

**3 Програмне та технічне забезпечення**

**3.1 Засоби розробки**

Для розробки поставленого завдання використовуються такі технології:

Ren'Py - безкоштовна, вільна і відкрита платформа для створення візуальних новел.

В можливості двигуна входить створення розгалужених діалогових потоків, збереження та відкати до потрібних точок сценарію, різні варіації переходів між сценами, створення різноманітних меню вибору, DLC та інше. Є можливість відігравати відео як в повноекранному режимі, так і в формі анімованих спрайтів. Для опису потрібних анімацій використовується описова мова ATL (Animation and Translation Language). Також є можливість анімувати керівні елементи графічного інтерфейсу.

Скриптова система в Ren'Py максимально спрощена, та використовується як послідовний опис подій в грі. Є можливість вставляти блоки Python-коду в сценарний потік, що значно збільшує потенційні можливості в ігровій сцені.

Стандартно, рушій підтримує Windows, останні версії Mac OS X та Linux, а також може бути запущений на Android та iOS 7+.

Але через свою відкритість та мінімальне використання сторонніх бібліотек може бути портований практично на будь-яку платформу, яка підтримує Python та SDL.

Програмування відбувається на мові програмування Python. Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

**3.2 Опис програмної реалізації**

Так як я тема дипломної це візуальна новела, а вона в свою чергу є грою, в неї є яскраво виражений інтерфейс.

При запуску візуальної новели відкривається головне меню, що продемонстровано на рисунку 3.1



Рис. 3.1 – Головне меню

При виборі «Начать», гра вітається з користувачем, що продемонстровано на рисунку 3.2

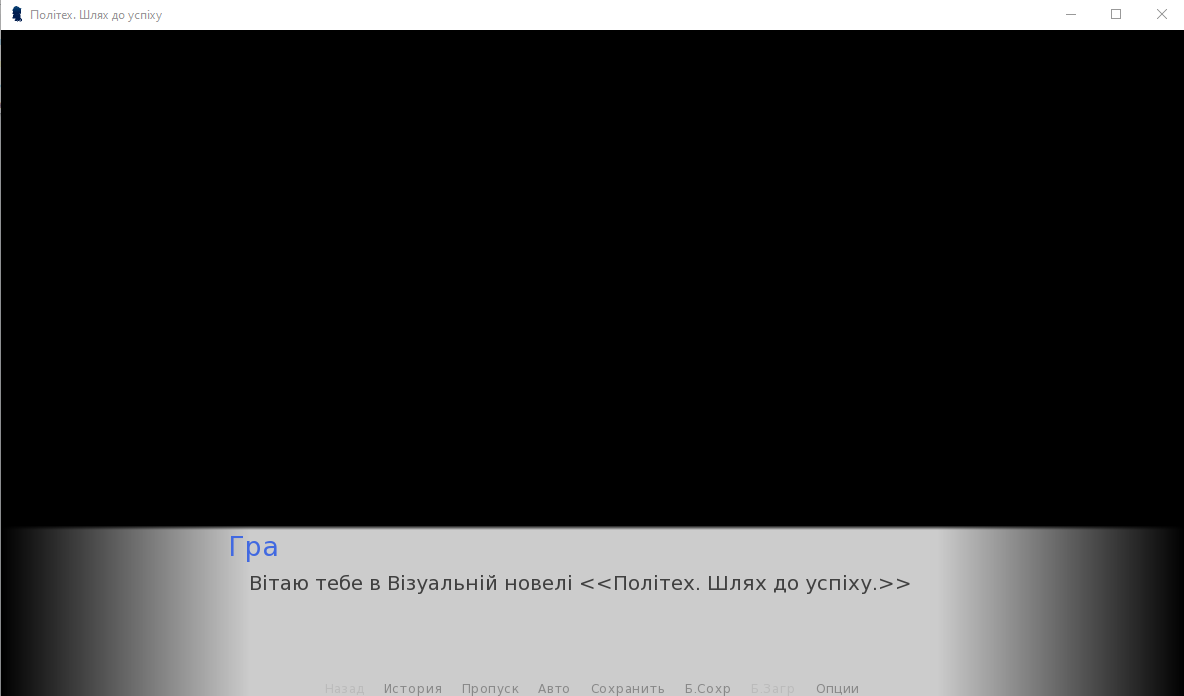


Рис. 3.2 – Пункт «Начать»

І далі питає, що ви хочете обрати, що продемонстровано на рисунку 3.3:

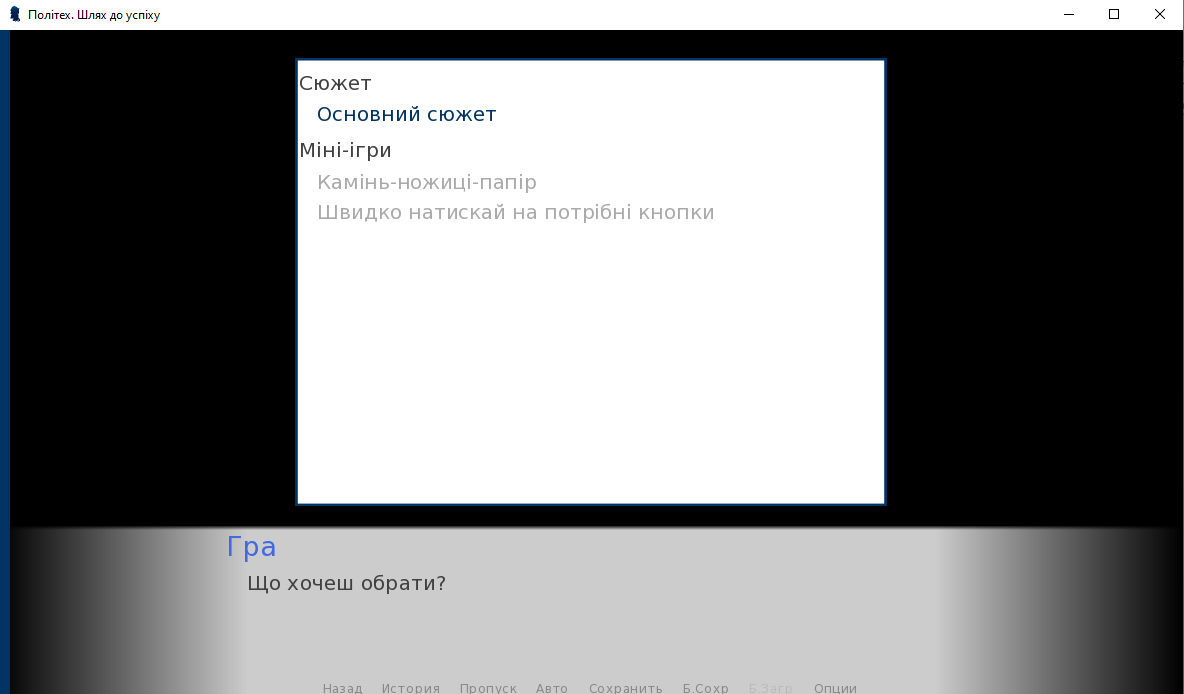
****

Рис. 3.3

Реалізація:

#мітка початку гри

label start:

window show

#початкове вікно

G "Вітаю тебе в Візуальній новелі <<Політех. Шлях до успіху.>> "

#мітка початку новелли

label novella:

#якщо ви лише зайшли в гру, вас запитують що ви хочете обрати

if novella\_first\_time:

$ G (\_("Що хочеш обрати?"), interact=False)

else:

#якщо ви зайшли в гру раніше, вас запитують чи це все що ви хотіли

$ G (\_("Це все що ти хотів?"), interact=False)

$ novella\_first\_time = False

$ renpy.choice\_for\_skipping()

call screen novella(adj=novella\_adjustment)

$ novel = \_return

call expression novel.label from \_call\_expression

jump novella

Вигляд Головного меню реалізований наступним чином:

init python:

novella = [ ]

class Section(object):

def \_\_init\_\_(self, title):

self.kind = "section"

self.title = title

novella.append(self)

class Syuzhet(object):

def \_\_init\_\_(self, label, title, move=True):

self.kind = "syuzhet"

self.label = label

self.title = title

novella.append(self)

class MiniIgry(object):

def \_\_init\_\_(self, label, title, move=True):

self.kind = "miniigry"

self.label = label

self.title = title

novella.append(self)

Section(\_("Сюжет"))

Syuzhet("osn\_syuzhet", \_("Основний сюжет"))

Section(\_("Міні-ігри"))

MiniIgry("minigameknp", \_("Камінь-ножиці-папір"))

MiniIgry("minigame2", \_("Швидко натискай на потрібні кнопки"))

screen novella(adj):

frame:

xsize 640

xalign .5

ysize 485

ypos 30

has side "c r b"

viewport:

yadjustment adj

mousewheel True

vbox:

for i in novella:

if i.kind == "syuzhet":

textbutton i.title:

action Return(i)

left\_padding 20

xfill True

elif i.kind == "miniigry":

textbutton i.title:

action Return(i)

left\_padding 20

xfill True

else:

null height 10

text i.title alt ""

null height 5

bar adjustment adj style "vscrollbar"

default novella\_adjustment = ui.adjustment()

default novella\_first\_time = True

Якщо ви натиснете на «Основний сюжет», почнеться сюжет, спочатку вас запитають ваше власне ім’я, якщо ви не хочете вводити, можна просто натиснути «Enter», і вас будуть називати іменем заданим в коді, що продемонстровано на рисунку 3.4.



Рис 3.4 – «Основний сюжет»

Реалізовано це так:

label start:

$ player\_name = renpy.input("Твоє імя? (Якщо не хочеш вибирати, натисни Enter)")

$ player\_name = player\_name.strip()

# .strip() - даний параметр видалить лишні або випадково набрані гравцем пробіли.

# Якщо гравець не хоче придумувати ім'я, то буде використано по замовчуванні:

if player\_name == "":

$ player\_name = "Михайло"

# Тепер інші персонажі можуть звертатись вибраним ім'ям.

Код всього сюжету, ви можете переглянути в Додатку А.

Меню вибору в грі виглядає наступним чином, що продемонстровано на рисунку 3.5



Рис 3.5 – Меню вибору

Код:

# При виборі, відбувається перехід до певної мітки.

menu:

"{cps=0}Цей хлопець здається хорошим, але якийсь дивний.. Чи варто мені з ним дружити? Адже в мене є друзі в групі.. Хоча я і хочу справжнього найкращого друга.{/cps}"

# Якщо гравець вибирає Так, відбувається перехід до мітки choice1\_yes

"Так. Чому б і ні?":

jump choice1\_yes

# Якщо гравець вибирає Ні, відбувається перехід до мітки choice1\_no

"Ні. В мене вже є друзі.":

jump choice1\_no

label choice1\_yes:

# Якщо гравець вибирає так, додається один бал.

$ ppoints += 1

$ menu\_flag = True

Y "Так, давай дружити!"

jump choice1\_done

label choice1\_no:

# Якщо гравець вибирає ні, бал віднімається.

$ ppoints -= 1

$ menu\_flag = False

Y "Ми ж одногрупники! Ми вже друзі."

jump choice1\_done

jump choice1\_done

label choice1\_done:

# ... гра продовжується тут.

Сумування балів і вихід на певну з кінцівок виглядає наступним чином:

$ question\_asked = False

# Змінна для збереження балів вибору кінцівки.

$ ppoints = 0

label choice1\_yes:

# Якщо гравець вибирає так, додається один бал.

$ ppoints += 1

$ menu\_flag = True

Y "Так, давай дружити!"

jump choice1\_done

label choice1\_no:

# Якщо гравець вибирає ні, бал віднімається.

$ ppoints -= 1

$ menu\_flag = False

Y "Ми ж одногрупники! Ми вже друзі."

jump choice1\_done

if (ppoints==2) :

Y "Я закінчив коледж добре і вступив в університет."

elif (ppoints==1) :

Y "Я нормально закінчив коледж і зміг вступити до університету на платне."

else:

Y "Мене забрали до армії."

Також, в грі реалізована клікабельна карта, вибір на якій, також впливає на сюжет, що продемонстровано на рисунку 3.6

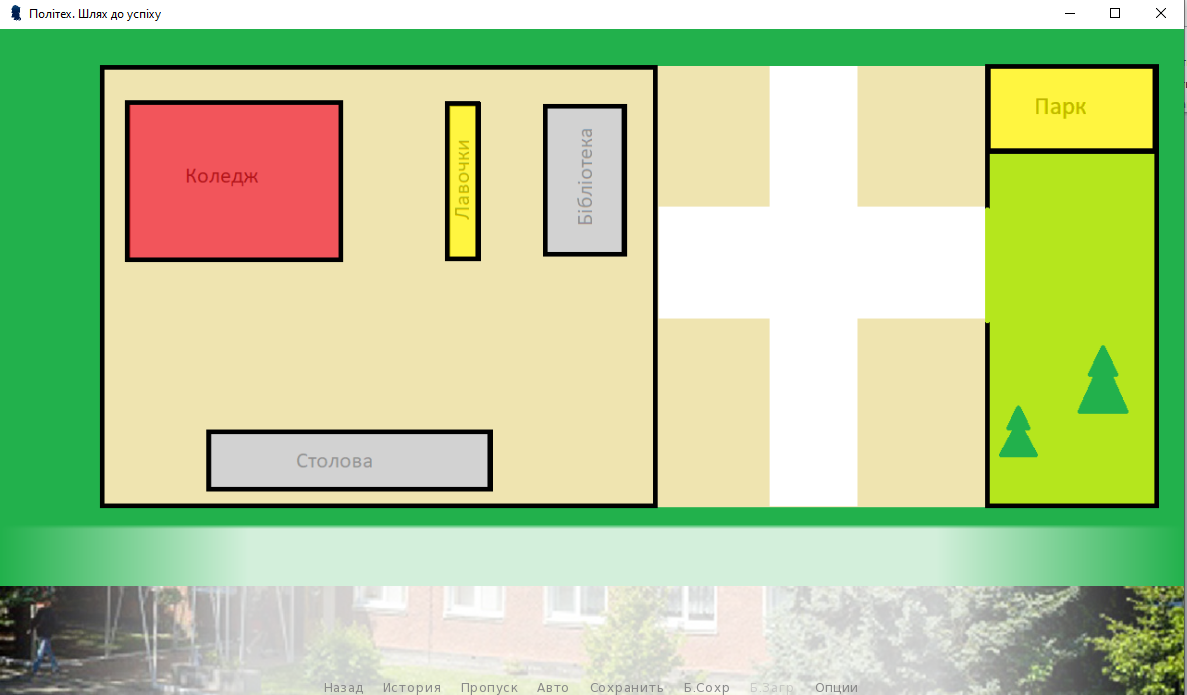


Рис 3.6 – Клікабельна карта

Код:

#карта

screen map:

imagemap:

ground "images/map/map\_ground.png"

idle "images/map/map\_idle.png"

hover "images/map/map\_hover.png"

#в лапках перші два значення відображають верхній лівий кут активної області

#наступні два це довжина і ширина області

#clicked означає що якщо на область натиснути, то спрацює означает Return()

#яке прийме значення яке вказане в лапках

hotspot (116, 67, 285, 185) action Return("college") alt "college"

hotspot (485, 77, 38, 172) action Return("lavka") alt "lavka"

hotspot (1058, 40, 190, 98) action Return("park") alt "park"

label example\_label:

call screen map

#наш результат минулого вибору в меню, завдяки Return()

#був збережений в хранилищі \_return

$ result = \_return

#тепер результат дій кліку буде залежати від вибору який ми зробили раніше

if result == "college":

jump choisel\_kabinet1

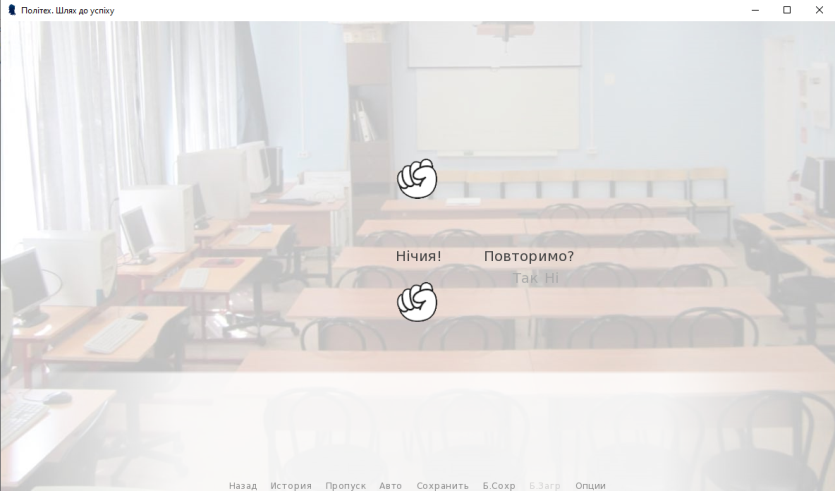
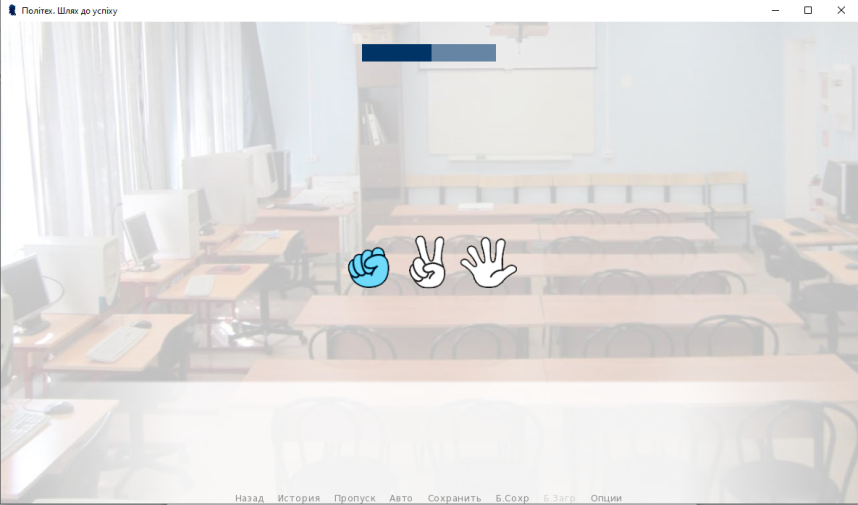
elif result == "lavka":

jump choisel\_lavka1

elif result == "park":

jump choisel\_park1

Міні-гру «Камінь-Ножиці-Папір» продемонстровано на рисунку 3.7 та 3.8.



Код:

# До початку основного сюжету гри:

Рис 3.7 Рис 3.8

init -2:

$ time = 3

$ p\_choise = 0

$ you\_play = False

$ firzt = True

screen knb\_timer:

timer time repeat True action [Hide('knb\_game\_buttons'), Show('knb\_game\_result')]

bar value AnimatedValue(0, time, time, time) xmaximum 200 xalign 0.5 yalign 0.05

screen knb\_game\_buttons:

modal True

use knb\_timer

hbox:

align .5, .5

imagebutton auto '/images/rock\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise', 'rock'), SelectedIf(p\_choise == 'rock')] hovered Play('sound', '/music/odd2.mp3')

imagebutton auto '/images/scissors\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise', 'scissors'), SelectedIf(p\_choise == 'scissors')] hovered Play('sound', '/music/odd2.mp3')

imagebutton auto '/images/paper\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise', 'paper'), SelectedIf(p\_choise == 'paper')] hovered Play('sound', '/music/odd2.mp3')

screen knb\_game\_result:

modal True

python:

enemy\_ch\_list = ["rock", "scissors", "paper"]

random\_lndx = renpy.random.randrange(3)

enemy\_ch = enemy\_ch\_list[random\_lndx]

if p\_choise <> 0:

add '/images/' + enemy\_ch + '\_idle.png' align .5, .3

if enemy\_ch == p\_choise:

text 'Нічия!' align .5, .5

elif enemy\_ch == 'rock' and p\_choise == 'scissors':

text 'Програш!' align .5, .5

elif enemy\_ch == 'rock' and p\_choise == 'paper':

text 'Перемога!' align .5, .5

elif enemy\_ch == 'paper' and p\_choise == 'scissors':

text 'Перемога!' align .5, .5

elif enemy\_ch == 'paper' and p\_choise == 'rock':

text 'Програш!' align .5, .5

elif enemy\_ch == 'scissors' and p\_choise == 'paper':

text 'Програш!' align .5, .5

elif enemy\_ch == 'scissors' and p\_choise == 'rock':

text 'Перемога!' align .5, .5

add '/images/' + p\_choise + '\_idle.png' align .5, .6

else:

text 'Ти не зробив вибір.' align .5, .4

text 'Повторимо?' align .65, .5

hbox:

align .65, .55

textbutton 'Так' action [Hide('knb\_game\_result'), Jump('knb\_play')]

textbutton 'Ні' action [Hide('knb\_game\_result'), Jump('to\_end')]

# В головному блоці основного сюжету гри:

label knb\_start:

"Зіграємо в КНП?"

menu:

"Так":

$ you\_play = True

jump knb\_play

"Ні":

jump to\_end

label knb\_play:

if firzt == True:

S 'Граємо в "Камінь-ножиці-папір"! Правила прості. Камінь виграє Ножиці, Ножиці виграють Папір, Папір виграє Камінь!'

else:

S 'Не тягни резину! Погнали!'

$ firzt = False

$ p\_choise = 0

show screen knb\_game\_buttons

scene bg college1

pause

label to\_end:

if you\_play == True:

S 'Бачу ти награвся!'

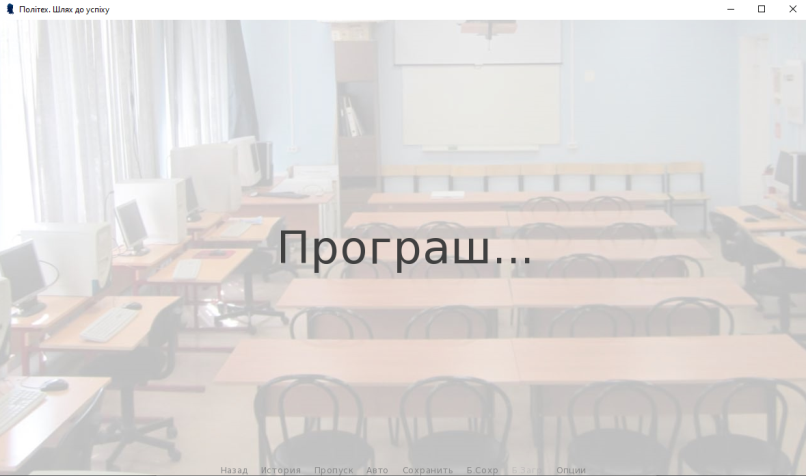
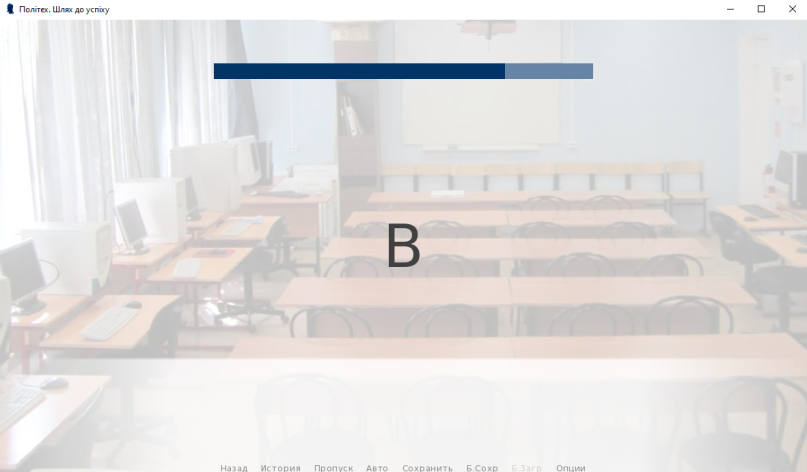
S 'До зустрічі!'

else:

S 'А чого тоді приперся?'

S 'Давай, удачі!'

Міні-гру «Швидко натискай на потрібні кнопки» що продемонстровано на рисунках 3.9 та 3.10



Код:

#міні-гра 2, натиснути на клавішу яку показано на екрані

init python:

Рис 3.9 Рис 3.10

import string

# змінні

qte\_word = ""

next\_k = ""

qteTime = .0

qteMaxTime = 5.0

abc = list(string.ascii\_lowercase)

# ініціалізація гри при запуску екрану

# параметри передаються при виклику екрана гри

# якщо слово пусте, то генерується рандомне довжиною length

# time - час, відведений на гру в секундах

def qte\_init(word="", time=5.0, length=5):

global qte\_word, next\_k, qteMaxTime, qteTime

qteMaxTime = time

qteTime = time

qte\_word = word.lower()

if word:

next\_k = qte\_word[0]

else:

for i in range(0, length):

qte\_word = qte\_word + renpy.random.choice(abc)

next\_k = qte\_word[0]

renpy.restart\_interaction()

# натиснення чергової потрібної кнопки, переходимо до іншої

def next\_key():

global qte\_word, next\_k

qte\_word = qte\_word[1:]

next\_k = ""

if qte\_word:

next\_k = qte\_word[0]

renpy.restart\_interaction()

NextKey = renpy.curry(next\_key)

qteInit = renpy.curry(qte\_init)

# сам екран гри

screen scr\_qte(word="", time=5.0, length=5):

# ініціалізація

on 'show' action qteInit(word, time, length)

modal True

if qte\_word:

# зменшуємо час виділений на гру, і провіряємо чи не вийшов він. якщо вийшов - програш

timer 0.01 repeat True action [SetVariable("qteTime", qteTime - .01), If(qteTime <= .0, true=Return(False))]

# відображаємо яку кнопку потрібно натиснути

text next\_k.upper() align(.5, .5) size 96

# якщо щось потрібно натиснути, то питає клавіатуру

if len(next\_k) == 1:

key next\_k action NextKey()

else:

# всі кнопки натиснули - перемога

timer .1 action Return(True)

# шкала часу

bar value StaticValue(qteTime, qteMaxTime) align(.5, .1) xmaximum 600

# міні-гра починається тут

label minigame2:

scene bg college\_knb

"Натискайте на кнопки на клавіатурі, які будуть показані на екрані. Потрібно натиснути 5 кнопок щоб виграти."

$ res = renpy.call\_screen("scr\_qte", word="balda", time=3.0, \_layer="master")

# $ res = renpy.call\_screen("scr\_qte", time=4.0, length=4, \_layer="master")

# $ res = renpy.call\_screen("scr\_qte", \_layer="master")

if res:

centered "{size=72}Перемога!"

else:

centered "{size=72}Програш..."

return

Базу даних продемонстровано на рисунку 3.11.

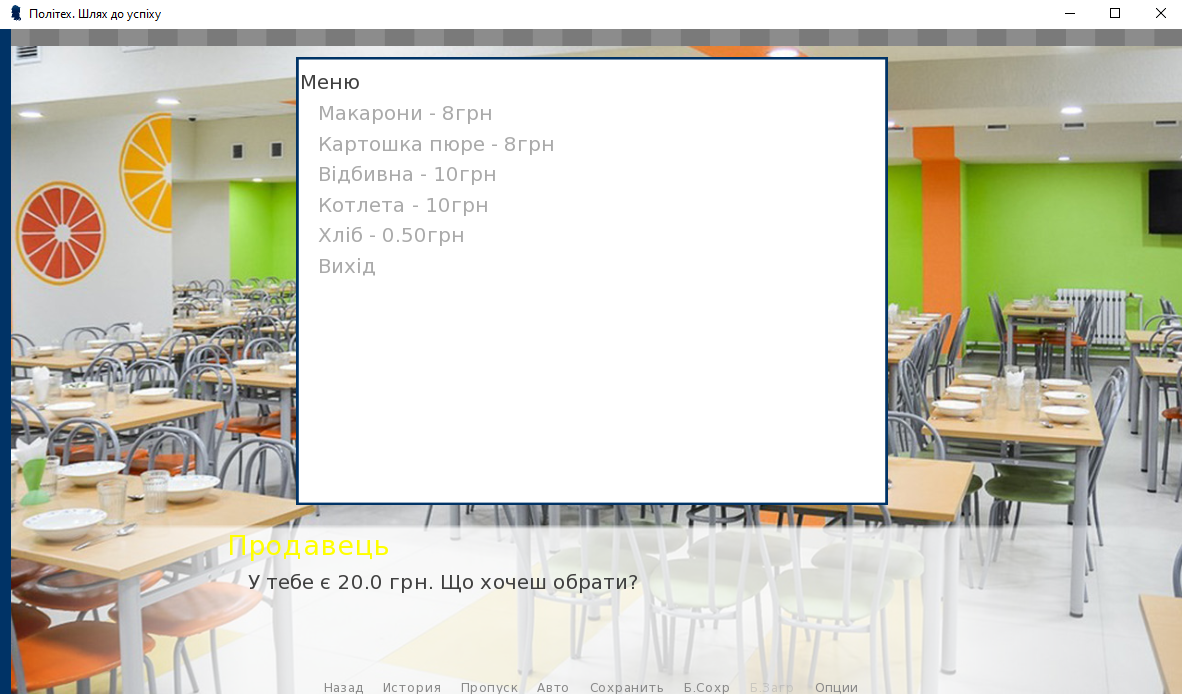


Рис. 3.11 – База даних

В базі даних зберігають дані про кількість ігрових грошей гравця. По ходу гри ця сума міняється. Також в базі зберігають дані про товари в їдальні, а саме їх назва та ціна. Це влаштована база даних в програму.

Код:

Створення бази даних:

init python:

import sqlite3

conn = sqlite3.connect('arsen.db')

c = conn.cursor()

#c.execute('''CREATE TABLE money (id int, sum float)''')

#c.execute("DELETE FROM money")

c.execute("INSERT INTO money VALUES(1, 0)")

#c.execute('''CREATE TABLE ida (id int, nazva varchar(20), cina float)''')

c.execute("INSERT INTO ida VALUES(1, 'makarony', 8)")

c.execute("INSERT INTO ida VALUES(2, 'pure', 8)")

c.execute("INSERT INTO ida VALUES(3, 'vidbyvna', 10)")

c.execute("INSERT INTO ida VALUES(4, 'kotleta', 10)")

c.execute("INSERT INTO ida VALUES(5, 'hlib', 0.50)")

c.execute("SELECT sum FROM `money`")

money = c.fetchone()[0]

Меню:

label menu\_pure:

$ selected\_tovar.append(2)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 2")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(2)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump start

Виклик під час вибору продукту:

label menu\_vidbyvna:

$ selected\_tovar.append(3)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 3")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(3)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump stolov

**Висновки до розділу**

У третьому розділі були наведені технології розробки візуальної новели, рушій Ren’Py та мову програмування Python та описано їх.

Також наведено уривки коду з поясненнями та скріншоти виконання програми.

Даний розділ дозволяє дослідити чи мета в повній мірі була досягнута, і чи відповідає гра вимогам, які були складені на початку.

**4 Економічна частина**

**1 Маркетинговий аналіз розробки та впровадження програмного продукту**

На сучасному етапі розвитку людства інформаційні технології використовуються в усіх сферах життя людини, зокрема в освіті, науці, роботі, повсякденному житті.

Процес інформатизації зачепив також ігрову індустрію, адже сьогодні відеоігри позиціонуються як дещо більше, ніж просто складова сфери розваг.

Ринок комп’ютерних ігор є одним з найбільш динамічно зростаючих ІТ-ринків у світі, оскільки прогнози на майбутнє показують стійке збільшення показників. Хоча не в усіх регіонах світу розвиток цієї індустрії проходить рівномірно, заперечувати лідерство ігрової індустрії в економіці високорозвинених країн неможливо. Це шлях у майбутнє, куди прямують розвинуті цивілізації. Такого висновку дійшли незалежні експерти відразу декількох провідних аналітичних компаній. Так, згідно з даними агентства “Newzoo”, за останні п’ять років доходи від продажу відеоігор у світі збільшились на 65%, або на 82 млрд. дол., досягнувши рекордної позначки у 122 млрд. дол. за підсумками 2017 року. Згідно з прогнозами до 2020 року капіталізація світового ринку відеоігор складе 166 млрд. дол., а до 2021 року – 180,1 млрд. дол., що буде означати приріст 30%.

Згідно з прогнозами незалежних аналітичних агентств, ігрова індустрія продовжить стрімке та стійке зростання як в коротко- так і в довгостроковій перспективі. Причому це стосується як зростання доходів компаній-учасників ринку, так і збільшення аудиторії користувачів. З точки зору інвестицій нині він є одним з найперспективніших ринків у світі.

Варто зазначити, що популяризація комп’ютерних ігор сприяє інтенсифікації розвитку суміжних продуктів, зокрема сервісів ігрового стрімінгу (безперервної передачі мультимедійних даних), таких як “Twitch” та “YouTube Gaming”, де можна дивитись їхнє відеопроходження, що для сучасної молоді є альтернативою телебачення. В країнах Європи та Америки організовуються турніри, де команди змагаються в проходженні таких трендових комп’ютерних ігор, як “Counter-Strike”, “LoL”, “Dota 2”. Крім того, на основі комп’ютерних ігор навіть знімають високобюджетні фільми (у 2016 році на екрани вийшли кінокартини “Warcraft” та “Assasin’s Creed”, зняті за мотивами однойменних відеоігор).

Представлення основних ринкових характеристик додатку за допомогою 3-рівневої моделі товару за Ф. Котлером:

1. Товар за задумом. Візуальна новела призначена для розваги. Візуальна новела пропонує приємний дизайн, та цікавий сюжет
2. Товар в реальному виконанні. В реальному часі візуальна новела відповідає всім заданим критеріям, з впевненістю можна сказати, що вона є якісною та має в собі весь задуманий функціонал
3. Товар з підкріпленням. Після установки візуальної новели, користувач має змогу повідомити розробнику про виявлені баги чи матеріально підтримати розробника. Візуальна новела і на далі буде оновлюватися і кожен користувач матиме доступ до нових версій

Конкурентне середовище даної візуальної новели не є дуже великим, на ринку досить мала кількість аналогів. Найпопулярнішим з них є «Doki Doki Literature Club!». Порівняльна характеристика основних параметрів заданої візуальної новели «Політех. Шлях до успіху» з аналогом продемонстровано в таблиці 4.1.

Таблиця 4.1 – Порівняльна характеристика основних параметрів додатку з його аналогом

|  |  |  |
| --- | --- | --- |
| Параметр | «Політех. Шлях до успіху» | «Doki Doki Literature Club!» |
| Доступність | Додаток є повністю безплатним | Додаток є повністю безплатним |
| Платформи | Windows, Android | Windows, Linux, macOS |
| Інтерфейс | Мінімалістичний | Набитий різними кнопками, приємний до сприйняття |
| Функціонал | Більший | Менший |
| Анімації | Відсутні | Присутні |

Позиціонування товару - це оптимальне розміщення товару в ринковому просторі. Результат позиціонування - це конкретні маркетингові дії з розробки, поширення та просування товару на ринок. Візуальна новела націлена на людей, які досягли віку 14+. Планується захоплювати аудиторію унікальним функціоналом додатку та цікавим сюжетом.

Одним з основних інструментів стратегічного управління, що оцінюють в комплексі внутрішні і зовнішні чинники, які впливають на розвиток компанії є SWOT-аналіз.

В таблиці 4.2 наведено приклад SWOT аналізу.

Таблиця 4.2 – SWOT аналіз.

|  |  |  |  |
| --- | --- | --- | --- |
| Strengths | Weaknesses | Opportunities | Threats |
| 1. Доступність  2. Легке пристосування  3. Гарантована якість | 1. Відсутність реклами  2. Недовіра клієнтів | 1. Реклама  2. Розширення клієнтської бази | 1. Конкуренція на ринку  2. Великі корпорації |

**4. 2 Економічне обґрунтування доцільності розробки та впровадження програмного продукту**

1. Розрахунок собівартості програмного продукту

Розробка будь-якого програмного продукту потребує певних матеріальних, часових і трудових витрат, а тому повинно відповідно окуповуватися. Собівартість продукції в економічному значенні – це сукупні витрата на її виробництво та збут.

Основою розрахунку собівартості ПП є визначення трудомісткості, яка характеризується переліком основних етапів і видів робіт, які повинні бути виконані в проєкті. Трудомісткість виконання роботи по проєкту має ймовірнісний характер і визначається в днях або годинах. Розрахунок трудомісткості представлений в таблиці 4.3.

Таблиця 4.3 – Розрахунок трудомісткості виконання проєкту

|  |  |  |
| --- | --- | --- |
| №  з/п | Види робіт | Трудомісткість, дн |
| 1 | Отримання технічного завдання | 1 |
| 2 | Збір інформації та ознайомлення із сферою дослідження | 10 |
| 3 | Вибір схеми побудови програми | 2 |
| 4 | Розробка загальної методики створення програмного продукту | 1 |
| 5 | Розробка основного алгоритму | 37 |
| 6 | Створення інтерфейсу | 1 |
| 7 | Налаштування | 2 |
| 8 | Підготовка технічної документації | 4 |
| 9 | Здача продукту | 1 |
|  | Всього | 58 |

Витрати на матеріали не передбачені.

Розрахунок витрат на електроенергію розраховуємо за формулою (4.1):

Ве = В\*П\*Ф грн. (4.1)

де, В – вартість однієї кВТ/год. електроенергії, грн./кВТ-год.;

П – встановлена потужність комп’ютера та інших пристроїв, кВт;

Ф – фактична кількість годин роботи комп’ютера та інших пристроїв при створенні ПП, год.

1 кВт енергії коштує 0.9 грн (до 100кВт)

Щоб вирахувати потужніть ком’ютера, необхідно перемножити вихідну напругу. 20V \* 3.25A = 65Вт/год.

65Вт/год : 1000 = 0.065кВт/год.

Фактична кількість годин роботи комп’ютера становить 212 годин.

Ве = 0.9 \* 0.065 \* 212 = 12.4 грн

Розрахунок амортизаційних витрат обладнання (згідно податкового кодексу України) можна проводити із застосуванням прямолінійного методу, за яким річна сума амортизації визначається діленням вартості, яка амортизується, на строк корисного використання об’єкта основних засобів. (до групи 4 «машини та обладнання» - входять електронно-обчислювальні машини, термін корисного використання яких складає не менше 2 років).

Амортизація розраховується за формулою (4.2):

 грн. (4.2)

де, Ц - балансова вартість комп’ютера, грн.;

Тк – термін корисного використання комп’ютера;

Т - термін фактичного використання комп’ютера пр. розробці ПП, місяці.

 = 1249.92 грн.

Витрати на оплату праці включають витрати на основну, додаткову заробітну плату та нарахування.

Основна заробітна плата нараховується виходячи з затверджених посадових окладів (тарифних ставок) відповідно до встановлених норм часу або норм виробітку продукції. Основна заробітна плата розраховується шляхом множення середньоденної (середньогодинної) зарплати у відповідному місяці на кількість відпрацьованих днів (годин) в цьому місяці. Для визначення середньоденної (середньогодинної) заробітної плати потрібно посадовий оклад працівника розділити на встановлену норму робочого часу.

Основна заробітна плата розробника (Зо) розраховується за формулою (4.3):

 грн. (4.3)

де, М - місячний посадовий оклад програміста-розробника, грн.;

Тр – число робочих днів в місяці, Тр = 22;

t – трудомісткість, число годин роботи розробника.

 = 35221.8 грн

Додаткова заробітна плата – це винагорода за працю в позаурочний та нічний час, в вихідні та святкові дні, надбавки за особливі умови праці, успішне виконання проекту тощо. Може встановлюватись у відсотках до основної заробітної плати.

Додаткова заробітна плата розробника (Зд) розраховується за формулою (4.4):

 грн. (4.4)

= 7044.4 грн

Нарахування на заробітну плату (Нзп) проводять у формі ЄСВ (22%) від суми основної та додаткової заробітної плати, формула (4.5):

 грн. (4.5)

= 9298.56 грн

Розрахунки витрат на оплату праці на основі даних про основну, додаткову заробітну плату та нарахування в таблиці 4.5.

Таблиця 4.5 – Витрати на оплату праці розробників ПП

|  |  |  |
| --- | --- | --- |
| №  з/п | Найменування витрат | Витрати, грн. |
| 1 | Основна заробітна плата | 35221.8 |
| 2 | Додаткова заробітна плата | 7044.4 |
| 3 | Нарахування на заробітну плату | 9298.56 |
| Всього | | 51564.76 |

Розрахунок загальної собівартості ПП проводимо на основі отриманих розрахунків витрат, а саме: витрати на матеріали, амортизаційні витрати, витрати на електроенергію, витрати на оплату праці, інші витрати (за потреби) в таблиці 4.6.

Таблиця 4.6 – Розрахунок собівартості програмного продукту

|  |  |  |
| --- | --- | --- |
| №  з/п | Найменування статей витрат | Витрати, грн. |
| 1 | Електроенергія | 12.4 |
| 2 | Амортизація | 1249.92 |
| 3 | Витрати на оплату праці | 51564.76 |
| Разом | | 52827.08 |

2. Визначення ціни реалізації програмного продукту

Однією з головних проблем господарюючих суб’єктів є вибір правильної політики встановленні цін на товари і послуги, які вони виробляють чи надають.

Може встановлюватись нижня і верхня межа ціни ПП.

Нижня межа ціни – це рівень мінімально допустимої ціни на продукт. Встановлення ціни нижче зазначеного рівня робить подальше його виробництво та продаж недоцільним, але бувають випадки, коли необхідно штучно знижувати ціну для початку або захопити певну цільову групу споживачів. Нижню межу ціни можна знайти за формулою 4.6

, грн. (4.6)

= 89806.04 грн

де S – собівартість програмного продукту, грн.;

Р – норма рентабельності, прогнозована розробником, %.

Верхня межа ціни програмного продукту – це та ціна, яку споживач готовий платити за продукт в реальних умовах ринку. Вона забезпечує заданий рівень конкурентоспроможності в порівнянні з конкуруючими продуктами. Верхня ціна програмного продукту може бути визначена на рівні середніх цін основних конкурентів на ринку або шляхом корегування собівартості програмного продукту на індекс конкурентоспроможності ПП, який розробляється. Верхня ціна заданого програмного забезпечення становить 100000 грн.

Для розрахунку економічної ефективності використаємо формулу:

Прибуток = Дохід – Витрати

89806.04 – 52827.08= 36978.96 грн

3 Соціально-економічні результати розробки та впровадження програмного продукту

Результатом роботи є візуальна новела. Впровадження програмного продукту буде відбуватись на платформах Steam та PlayMarket. Кожен зможе скачати та встановити візуальну новелу.

Просування візуальної новели планує відбуватись так:

1. Створення сайту посвяченого грі. Сайт для публікації новин, опису гри, посилання на скачування та посилання на облікові записи в соціальних мережах.
2. Блог розробника. Вкладка на сайті, де будуть вказані останні оновлення, досягнення та новини, які зв’язані з грою.
3. Облікові записи в Twitter, Facebook та Instagram. Облікові записи призначені також для публікації регулярних новин, концепт-артів та скріншотів, зв’язаних з новелою.
4. Трейлер гри. В якому буде показано можливості візуальної новели та представлення її для зацікавлення користувачів.

**Висновки до розділу**

У даному розділі був проведений маркетинговий аналіз, який мав на меті оцінку можливості реалізації програмного продукту розробленого за час дипломного проектування на ринку та отримання рівня доходу, що дав би можливість покрити витрати за проектом.

Також в розділі був наведений кошторис всіх витрат за час проектування. А саме: витрати на електроенергію, амортизація, витрати та нарахування на заробітну плату.

В кінці економічної частини проведений розрахунок економічної ефективності, підраховано, що рівень доходів перевищує рівень витрат. Також, були наведені соціально-економічні результати розробки та впровадження програмного продукту.

Отже в даному розділі було економічно обґрунтовано доцільності розробки програмного забезпечення, розробленого за час дипломного проекту.

**ЗАГАЛЬНІ ВИСНОВКИ**

Під час роботи над дипломним проектом було проведено розробку програмного забезпечення у вигляді візуальної новели. При написанні дипломної роботи було освоєну мову програмування Python та використання засобів об’єктно-орієнтованого програмування цієї мови, освоєно движок RenPy.

Було реалізовано:

* Автоматичне збереження гри;
* Відкат, для повернення раніше показаного екрану;
* Можливість пропускати текст при відтворенні, в тому числі можливість показувати лише текст, який не був показаний раніше;
* Автоматичну прокрутку тексту без використання клавіатури, що може бути зручно при великих обсягах тексту;
* Можливість приховувати текст так, що користувач може бачити зображення за ним;
* Запис вибору гравця і вихід на певну кінцівку;
* Введення ім’я гравця на початку гри;
* Можливість використання бази даних;
* Вибір дії за допомогою кліку (клікабельна карта);
* Дві міні-гри.

Мета в повній мірі досягнута, гра відповідає вимогам, які були складені на початку.

Отже, завдання дипломного проектування виконано і результат його виконання може впроваджуватись на ігрові платформи.

**ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА**

1. <https://www.renpy.org/> - про движок Ren’Py;
2. <https://www.renpy.org/doc/html/> - документація Ren’Py;
3. «Учим Python, делая крутые игры», Эл Свейгарт – книга для вивчення мови програмування Python за допомогою створення ігор.
4. ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання»
5. Глушаков С.В., Ломотько Д.В., Бази даних, К., Фолио, 2002 – 493 с.
6. Методичні вказівки до виконання дипломного проекту.

ДОДАТОК А

* Код основного сюжету візуальної новели:

# основний сюжет

init python:

import sqlite3

conn = sqlite3.connect('arsen.db')

c = conn.cursor()

#c.execute('''CREATE TABLE money (id int, sum float)''')

#c.execute("DELETE FROM money")

c.execute("INSERT INTO money VALUES(1, 0)")

#c.execute('''CREATE TABLE ida (id int, nazva varchar(20), cina float)''')

c.execute("INSERT INTO ida VALUES(1, 'makarony', 8)")

c.execute("INSERT INTO ida VALUES(2, 'pure', 8)")

c.execute("INSERT INTO ida VALUES(3, 'vidbyvna', 10)")

c.execute("INSERT INTO ida VALUES(4, 'kotleta', 10)")

c.execute("INSERT INTO ida VALUES(5, 'hlib', 0.50)")

c.execute("SELECT sum FROM `money`")

money = c.fetchone()[0]

init -2:

$ time = 5

$ p\_choise1 = 0

$ you\_play1 = False

$ firzt1 = True

screen knb\_timer1:

timer time repeat True action [Hide('knb\_game\_buttons1'), Show('knb\_game\_result1')]

bar value AnimatedValue(0, time, time, time) xmaximum 200 xalign 0.5 yalign 0.05

screen knb\_game\_buttons1:

modal True

use knb\_timer1

hbox:

align .5, .5

imagebutton auto '/images/knb/rock\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise1', 'rock'), SelectedIf(p\_choise1 == 'rock')] hovered Play('sound', '/music/odd2.mp3')

imagebutton auto '/images/knb/scissors\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise1', 'scissors'), SelectedIf(p\_choise1 == 'scissors')] hovered Play('sound', '/music/odd2.mp3')

imagebutton auto '/images/knb/paper\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise1', 'paper'), SelectedIf(p\_choise1 == 'paper')] hovered Play('sound', '/music/odd2.mp3')

screen knb\_game\_result1:

modal True

python:

enemy\_ch\_list1 = ["rock", "scissors", "paper"]

random\_lndx = renpy.random.randrange(3)

enemy\_ch1 = enemy\_ch\_list1[random\_lndx]

if p\_choise1 <> 0:

add '/images/knb/' + enemy\_ch1 + '\_idle.png' align .5, .3

if enemy\_ch1 == p\_choise1:

text 'Нічия!' align .5, .5

elif enemy\_ch1 == 'rock' and p\_choise1 == 'scissors':

text 'Програш!' align .5, .5

elif enemy\_ch1 == 'rock' and p\_choise1 == 'paper':

text 'Перемога!' align .5, .5

elif enemy\_ch1 == 'paper' and p\_choise1 == 'scissors':

text 'Перемога!' align .5, .5

elif enemy\_ch1 == 'paper' and p\_choise1 == 'rock':

text 'Програш!' align .5, .5

elif enemy\_ch1 == 'scissors' and p\_choise1 == 'paper':

text 'Програш!' align .5, .5

elif enemy\_ch1 == 'scissors' and p\_choise1 == 'rock':

text 'Перемога!' align .5, .5

add '/images/knb/' + p\_choise1 + '\_idle.png' align .5, .6

else:

text 'Ти не зробив вибір.' align .5, .4

text 'Повторимо?' align .65, .5

hbox:

align .65, .55

textbutton 'Так' action [Hide('knb\_game\_result1'), Jump('knb\_play1')]

textbutton 'Ні' action [Hide('knb\_game\_result1'), Jump('choice1\_done2')]

#наступний день надпис

screen text\_example():

frame:

xalign 0.5 ypos 300

text \_("Наступний день."):

size 30

# Початок гри:

label osn\_syuzhet:

$ question\_asked = False

# Змінна для збереження балів вибору кінцівки.

$ ppoints = 0

"{b}ДІСКЛЕЙМЕР! Всі персонажі придумані автором гри. Всі співпадіння випадкові.{/b}"

$ player\_name = renpy.input("Твоє імя? (Якщо не хочеш вибирати, натисни Enter)")

$ player\_name = player\_name.strip()

# .strip() - даний параметр видалить лишні або випадково набрані гравцем пробіли.

# Якщо гравець не хоче придумувати ім'я, то буде використано по замовчуванні:

if player\_name == "":

$ player\_name = "Михайло"

# Тепер інші персонажі можуть звертатись вибраним ім'ям.

scene bg college2

# Задній фон.

# Початок сюжету.

G "в тебе є [money] грн"

"{i}Сьогодні перший навчальний день після канікул. Другий семестр. Я вчусь на програміста, адже це так цікаво, а ще, я обожнюю математику!!{/i}"

"{i}Зараз я навчаюсь на 4 курсі і скоро закінчу коледж. Від мене залежить моє майбутнє після нього.{/i}"

"{i}На вході в коледж висить вивіска: <<Політех - це успіх>> Я точно стану успішним програмістом!!{/i}"

"{i}Мені дуже подобається навчатись в групі П-41. Це точно найкраща група, не те що ті П-42.{/i}"

"Люди почали збиратись біля входу. Там стояв якийсь дивний хлопець, який ковиряв ногою землю. Здається він новенький."

show lucy slava

"Дивний хлопець" "Привіт! Мене звати Слава. Я новенький, але я дуже шарю в програмуванні. І я Анімешник! А ти?"

show lucy slava at right

Y "А мене звати [player\_name]. Приємно познайомитись! Я навчаюсь тут з першого курсу."

show lucy slava at center

S "Ваау! Так круто! Давай дружити?"

"Цей хлопець здається хорошим, але якийсь дивний.. Чи варто мені з ним дружити? Адже в мене є друзі в групі.. Хоча я і хочу справжнього найкращого друга."

hide lucy slava

# Меню вибору.

# При виборі, відбувається перехід до певної мітки.

menu:

"{cps=0}Цей хлопець здається хорошим, але якийсь дивний.. Чи варто мені з ним дружити? Адже в мене є друзі в групі.. Хоча я і хочу справжнього найкращого друга.{/cps}"

# Якщо гравець вибирає Так, відбувається перехід до мітки choice1\_yes

"Так. Чому б і ні?":

jump choice1\_yes

# Якщо гравець вибирає Ні, відбувається перехід до мітки choice1\_no

"Ні. В мене вже є друзі.":

jump choice1\_no

label choice1\_yes:

# Якщо гравець вибирає так, додаються бали.

$ ppoints += 2

$ menu\_flag = True

Y "Так, давай дружити!"

jump choice1\_done0

label choice1\_no:

# Якщо гравець вибирає ні, бал віднімається.

$ ppoints -= 1

$ menu\_flag = False

Y "Ми ж одногрупники! Ми вже друзі."

jump choice1\_done0

label choice1\_done0:

# ... гра продовжується тут.

Y "Добре, я йду, мені потрібно ще.. ем.. ну треба мені."

"Він звісно і новенький, але я не хочу з ним зараз тусуватись. Є справи важливіші."

"{cps=0}Куди підемо зараз?{/cps}"

jump example\_label

label choisel\_kabinet1:

$ppoints +=2

$ menu\_flag = True

"Я йду до кабінету. Не хочу спізнитися на пару. Та й буду мати час щоб підготуватись, якщо вчитель щось спитає."

jump vubir\_kabinet

label choisel\_lavka1:

$ppoints +=1

$ menu\_flag = True

"О, бачу своїх знайомих на лавці, підійду до них. Давно не бачились. Думаю встигну на пару."

jump vubir\_lavka

label choisel\_park1:

$ppoints -=1

$ menu\_flag = False

"Йду покурю в парк. Якраз з своїми побачусь, стрільну в когось сігу. Пофіг якщо спізнюсь."

jump vubir\_park

# ... гра продовжується тут.

label vubir\_kabinet:

show bg college1

with slowdissolve

Y "Я вже в кабінеті."

Y "Канікули промайнули швидко. Скоро закінчення коледжу. Потрібно буде поступати. Так хочу бути як Біл Гейтс чи Гейб Ньюелл. Вони мої кумири!"

Y "Так, я повторю краще завдання."

jump choice1\_done1

label vubir\_lavka:

show bg college2

with slowdissolve

Y "Я сиджу на лавочці буля своїх. З деякими гуляли на канікулах. А з деякими не бачились. Класно зараз так сидіти з ними і слухати історії хто як провів канікули."

Y "Домовились сьогодні вечором піти в гру. Хоча я не знаю точно чи підемо, залежно від того скільки домашки зададуть."

Y "Ми розмовляли, поки не продзвенів дзвінок. Ми трішки спізнились на пару. Але не критично."

jump choice1\_done1

label vubir\_park:

show bg park

with slowdissolve

Y "Я в парку. Слухаю історію про те як Грубий вчора виграв матч в кс, тащив."

Y "Заздрю. Я тоже хочу тащити. Треба більше задротити щоб прокачати свій скілл. Сьогодні треба буде зайти в гру."

Y "Ми спізнились на пару і на нас кричали. Пофіг."

jump choice1\_done1

label choice1\_done1:

show bg college1

with slowdissolve

"{i}Сиджу роздивляюсь одногрупників.{/i}"

"{i}Кожен займається хто чим.{/i}"

"{i}Хтось слухає викладача. Хтось малює. Хтось в щось грає. Можливо і мені пограти?{/i}"

"Підемо в Камінь-ножиці-папір?"

$ you\_play1 = False

$ firzt1 = True

menu:

'Так':

$ you\_play1 = True

jump knb\_play1

'Ні':

jump choice1\_done2

# Міні-гра "Камінь-ножиці-папір"

label knb\_play1:

if firzt1 == True:

S 'Граємо в "Камінь-ножиці-папір"! Правила прості. Камінь виграє Ножиці, Ножиці виграють Папір, Папір виграє Камінь!'

else:

S 'Не тягни резину! Погнали!'

$ firzt1 = False

$ p\_choise1 = 0

show screen knb\_game\_buttons1

scene bg college\_knb

pause

label to\_end1:

if you\_play1 == True:

'Бачу ти награвся!'

'До зустрічі!'

else:

'Ну, як хочеш.'

# Кінець міні-гри

label choice1\_done2:

"{i}Хоча ні. Я не хочу грати.{/i}"

show bg college1

"Вирішую все-таки послухати що каже вчитель."

scene bg room

with slowdissolve

"Час пройшов незамітно. Перший навчальний день підійшов до завершення."

"{i}Я вже вдома. Домашньої роботи сьогодні немає і я можу зайнятись чимось своїм.{/i}"

scene expression "#000000"

show screen text\_example

with Dissolve (1.0)

pause .5

hide screen text\_example

# Кишенькові гроші

python:

c.execute("SELECT sum FROM money")

money = c.fetchone()[0] + 20

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

G "в тебе є [money] грн"

scene bg roadcollege

with slowdissolve

"{i}Іду до коледжу.{/i}"

"Весна. Навколо розцвітають дерева. Все озеленіло. Зима закінчилась. Така насолода для очей і для душі."

"{i}Сьогодні мені приснився гарний сон.{/i}"

"{i}Мені снилось що я..{/i}"

"{i}Стоп.{/i}"

"{i}Хто це там попереду..{/i}"

"{i}Здається це новенький, Слава..{/i}"

"Піти з ним до коледжу?"

menu:

"{cps=0}Піти з ним до коледжу?{/cps}"

"Так, він здається хорошим.":

jump choice1\_yes3

"Можливо мені получиться пройти так, щоб він мене не помітив..":

jump choice1\_no3

"Ой ні, я краще піду довшою дорогою.":

jump choise1\_no4

label choice1\_yes3:

$ppoints +=1

$ menu\_flag = True

"{i}Я залюбки піду разом з ним! Він мені подобається. Можливо ми подружимось.{/i}"

Y "Слава, зачекай!"

"Він помітив мене і зачекав, пока я наздожену."

show lucy slava at center

S "Привіт, [player\_name]!"

show lucy slava at right

Y "Привіт, ти теж недалеко живеш?"

show lucy slava at center

S "Так, в будинку на куті. Нещодавно переїхали."

show lucy slava at right

Y "Круто, зможемо разом ходити в коледж."

hide lucy slava

"Ми дійшли до коледжу, всю дорогу ми розказували про себе і різні історії з життя. Було цікаво і не скучно йти разом."

jump choisel\_done3

label choice1\_no3:

$ppoints +=0

$ menu\_flag = False

"{i}Я просто надіюсь що він мене не помітить..{/i}"

"Опустивши голову і намагаючись пройти швидко повз Славу, він мене помітив."

"{i}Блін.. Ну що ж, прийдеться йти разом з ним.{/i}"

show lucy slava at center

S "Привіт, [player\_name]. Ти також недалеко живеш?"

show lucy slava at right

Y "Так.."

show lucy slava at center

S "Круто! Ми можемо ходити разом до коледжу! Я дуже радий цьому, хех."

show lucy slava at right

"Слава усміхнувся."

hide lucy slava

"{i}Можливо він і не такий поганий..{/i}"

"Ми дійшли до коледжу, всю дорогу Слава розказував про себе і історії з життя. Інколи було навіть цікаво."

jump choisel\_done3

label choise1\_no4:

$ppoints -=1

$ menu\_flag = False

"{i}Та ну, я краще дійсно обійду. Йти ще з цим душним. Краще до Грубого подзвоню.{/i}"

"Діставши телефон з карману й набравши Влада, ми почали говорити."

"{i}Жаль що він живе далеко, так би разом ходили..{/i}"

Y "Дороу, грубий, шо ти?"

V "Прив. Та таке, вже в шаразі. Ти лиш йдеш?"

Y "Да. Мушу йти довшою дорогою, виявляється Слава, новенький, живе десь недалеко. Не хочу з ним йти."

V "Ха-ха, та могли б подружитись, чо ти."

Y "Ой, ні, дякую. В мене є ти, бро."

V "Броо."

"Дорога до коледжу подолалась незамітно, завдяки розмові з Грубим."

jump choisel\_done3

scene bg college1

with slowdissolve

label choisel\_done3:

"{i}Знову пари.{/i}"

"{i}Було трішки скучно на канікулах і хотілось до коледжу. Але важко привикати знову до такого режиму.{/i}"

"{i}Хоча і класно бачити всіх своїх.{/i}"

"Перерва. Піти мені зараз в столову перекусити?"

menu:

"{cps=0}Перерва. Піти мені зараз в столову перекусити?{/cps}"

"Так. Кишенькові ж для цього мені дали.":

jump stolova\_yes

"Ні. Я краще зберу на гру, щоб грати з Грубим.":

jump stolova\_no

label stolova\_yes:

$ppoints +=1

$ menu\_flag = True

jump vubir\_stolova

label stolova\_no:

"Хочу гру. Будемо грати з Грубим і я підніму собі очки крутості!"

jump kinec

label vubir\_stolova:

show bg stolova

with slowdissolve

Y "Я в столовій. Хочу щось перекусити."

jump stolov

label kinec:

G "в тебе є [money] грн"

if (ppoints==6 or ppoints==5):

G "Далі сюжету нажаль немає. Але ти рухаєшся у напрямку найкращої з кінцівок!"

elif (ppoints==4 or ppoints==3):

G "Далі сюжету нажаль немає. Але ти рухаєшся у напрямку хорошої кінцівки!"

else:

G "Далі сюжету нажаль немає. Але ти рухаєшся у напрямку найгіршої з кінцівок!"

return

* Код реалізації столової:

init python:

selected\_tovar = [ ]

stolov = [ ]

class Sections(object):

def \_\_init\_\_(self, title):

self.kind = "sections"

self.title = title

stolov.append(self)

class Stolova(object):

def \_\_init\_\_(self, label, title, move = True):

self.kind = "stolova"

self.label = label

self.title = title

stolov.append(self)

Sections(\_("Меню"))

Stolova("menu\_makarony", \_("Макарони - 8грн"))

Stolova("menu\_pure", \_("Картошка пюре - 8грн"))

Stolova("menu\_vidbyvna", \_("Відбивна - 10грн"))

Stolova("menu\_kotleta", \_("Котлета - 10грн"))

Stolova("menu\_hlib", \_("Хліб - 0.50грн"))

Stolova("exit\_stolova", \_("Вихід"))

screen stolov(adj):

frame:

xsize 640

xalign .5

ysize 485

ypos 30

has side "c r b"

viewport:

yadjustment adj

mousewheel True

vbox:

for i in stolov:

if i.kind == "stolova":

textbutton i.title:

action Return(i)

left\_padding 20

xfill True

else:

null height 10

text i.title alt ""

null height 5

bar adjustment adj style "vscrollbar"

default stolov\_adjustment = ui.adjustment()

default stolov\_first\_time = True

label stolov:

if stolov\_first\_time:

$ P (\_("У тебе є [money] грн. Що хочеш обрати?"), interact=False)

else:

$ P (\_("Це все що ти хотів? В тебе залишилось [money] грн."), interact=False)

$ stolov\_first\_time = False

$ renpy.choice\_for\_skipping()

call screen stolov(adj=stolov\_adjustment)

$ stol = \_return

call expression stol.label from \_call\_expression\_

jump start

label menu\_makarony:

$ selected\_tovar.append(1)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 1")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(1)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump stolov

label menu\_pure:

$ selected\_tovar.append(2)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 2")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(2)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump start

label menu\_vidbyvna:

$ selected\_tovar.append(3)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 3")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(3)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump stolov

label menu\_kotleta:

$ selected\_tovar.append(4)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 4")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(4)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump stolov

label menu\_hlib:

$ selected\_tovar.append(5)

$ deneg\_net = 0

$ hvatae = 100

python:

c.execute("SELECT sum FROM money WHERE id = 1")

temp\_money = c.fetchone()[0]

c.execute ("SELECT cina FROM ida WHERE id = 5")

deneg\_net = temp\_money - c.fetchone()[0]

if deneg\_net < 0:

deneg\_net = temp\_money

selected\_tovar.remove(5)

hvatae = 0

money = deneg\_net

c.execute ("UPDATE money SET sum = (?) WHERE id = 1", (money,))

if hvatae == 0:

P "В тебе не вистачає коштів!"

jump stolov

label exit\_stolova:

$ nazvy\_tovariv = "Ти взяв: "

if selected\_tovar.count(1) > 0:

$ nazvy\_tovariv += "макарони | "

if selected\_tovar.count(2) > 0:

$ nazvy\_tovariv += "картошку пюре | "

if selected\_tovar.count(3) > 0:

$ nazvy\_tovariv += "котлету | "

if selected\_tovar.count(4) > 0:

$ nazvy\_tovariv += "відбивну | "

if selected\_tovar.count(5) > 0:

$ nazvy\_tovariv += "хліб | "

P "Смачного! [nazvy\_tovariv]"

jump kinec

* Головний скріпт:

# Визначення персонажів гри.

define G = Character(\_("Гра"), color="#4169E1")

define Y = Character(\_("Я"), color="#4169E1")

define S = Character(\_("Слава Анімушник"), color="#000080")

define V = Character(\_("Влад Грубий"), color="#008000")

define P = Character(\_("Продавець"), color="#FFFF00")

define slowdissolve = Dissolve(1.0)

init python:

novella = [ ]

class Section(object):

def \_\_init\_\_(self, title):

self.kind = "section"

self.title = title

novella.append(self)

class Syuzhet(object):

def \_\_init\_\_(self, label, title, move=True):

self.kind = "syuzhet"

self.label = label

self.title = title

novella.append(self)

class MiniIgry(object):

def \_\_init\_\_(self, label, title, move=True):

self.kind = "miniigry"

self.label = label

self.title = title

novella.append(self)

Section(\_("Сюжет"))

Syuzhet("osn\_syuzhet", \_("Основний сюжет"))

Section(\_("Міні-ігри"))

MiniIgry("minigameknp", \_("Камінь-ножиці-папір"))

MiniIgry("minigame2", \_("Швидко натискай на потрібні кнопки"))

screen novella(adj):

frame:

xsize 640

xalign .5

ysize 485

ypos 30

has side "c r b"

viewport:

yadjustment adj

mousewheel True

vbox:

for i in novella:

if i.kind == "syuzhet":

textbutton i.title:

action Return(i)

left\_padding 20

xfill True

elif i.kind == "miniigry":

textbutton i.title:

action Return(i)

left\_padding 20

xfill True

else:

null height 10

text i.title alt ""

null height 5

bar adjustment adj style "vscrollbar"

default novella\_adjustment = ui.adjustment()

default novella\_first\_time = True

label start:

window show

G "Вітаю тебе в Візуальній новелі <<Політех. Шлях до успіху.>> "

label novella:

if novella\_first\_time:

$ G (\_("Що хочеш обрати?"), interact=False)

else:

$ G (\_("Це все що ти хотів?"), interact=False)

$ novella\_first\_time = False

$ renpy.choice\_for\_skipping()

call screen novella(adj=novella\_adjustment)

$ novel = \_return

call expression novel.label from \_call\_expression

jump novella

* Міні-гра «Камінь-ножиці-папір»:

#міні-гра 1, камінь-ножиці-папір

init -2:

$ time = 5

$ p\_choise = 0

$ you\_play = False

$ firzt = True

screen knb\_timer:

timer time repeat True action [Hide('knb\_game\_buttons'), Show('knb\_game\_result')]

bar value AnimatedValue(0, time, time, time) xmaximum 200 xalign 0.5 yalign 0.05

screen knb\_game\_buttons:

modal True

use knb\_timer

hbox:

align .5, .5

imagebutton auto '/images/knb/rock\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise', 'rock'), SelectedIf(p\_choise == 'rock')] hovered Play('sound', '/music/odd2.mp3')

imagebutton auto '/images/knb/scissors\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise', 'scissors'), SelectedIf(p\_choise == 'scissors')] hovered Play('sound', '/music/odd2.mp3')

imagebutton auto '/images/knb/paper\_%s.png' focus\_mask True action [Play('sound', '/music/done\_1.mp3'), SetVariable('p\_choise', 'paper'), SelectedIf(p\_choise == 'paper')] hovered Play('sound', '/music/odd2.mp3')

screen knb\_game\_result:

modal True

python:

enemy\_ch\_list = ["rock", "scissors", "paper"]

random\_lndx = renpy.random.randrange(3)

enemy\_ch = enemy\_ch\_list[random\_lndx]

if p\_choise <> 0:

add '/images/knb/' + enemy\_ch + '\_idle.png' align .5, .3

if enemy\_ch == p\_choise:

text 'Нічия!' align .5, .5

elif enemy\_ch == 'rock' and p\_choise == 'scissors':

text 'Програш!' align .5, .5

elif enemy\_ch == 'rock' and p\_choise == 'paper':

text 'Перемога!' align .5, .5

elif enemy\_ch == 'paper' and p\_choise == 'scissors':

text 'Перемога!' align .5, .5

elif enemy\_ch == 'paper' and p\_choise == 'rock':

text 'Програш!' align .5, .5

elif enemy\_ch == 'scissors' and p\_choise == 'paper':

text 'Програш!' align .5, .5

elif enemy\_ch == 'scissors' and p\_choise == 'rock':

text 'Перемога!' align .5, .5

add '/images/knb/' + p\_choise + '\_idle.png' align .5, .6

else:

text 'Ти не зробив вибір.' align .5, .4

text 'Повторимо?' align .65, .5

hbox:

align .65, .55

textbutton 'Так' action [Hide('knb\_game\_result'), Jump('knb\_play')]

textbutton 'Ні' action [Hide('knb\_game\_result'), Jump('to\_end')]

#тут в лебел старт було:

# Міні-гра "Камінь-ножиці-папір"

label minigameknp:

label knb\_play:

if firzt == True:

S 'Граємо в "Камінь-ножиці-папір"! Правила прості. Камінь виграє Ножиці, Ножиці виграють Папір, Папір виграє Камінь!'

else:

S 'Не тягни резину! Погнали!'

$ firzt = False

$ p\_choise = 0

show screen knb\_game\_buttons

scene bg college\_knb

pause

label to\_end:

if you\_play == True:

'Бачу ти награвся!'

'До зустрічі!'

else:

'Ну, як хочеш.'

# Кінець міні-гри

* Міні-гра «Швидко натискай на потрібні кнопки»:

#міні-гра 2, натиснути на клавішу яку показано на екрані

init python:

import string

# змінні

qte\_word = ""

next\_k = ""

qteTime = .0

qteMaxTime = 5.0

abc = list(string.ascii\_lowercase)

# ініціалізація гри при запуску екрану

# параметри передаються при виклику екрана гри

# якщо слово пусте, то генерується рандомне довжиною length

# time - час, відведений на гру в секундах

def qte\_init(word="", time=5.0, length=5):

global qte\_word, next\_k, qteMaxTime, qteTime

qteMaxTime = time

qteTime = time

qte\_word = word.lower()

if word:

next\_k = qte\_word[0]

else:

for i in range(0, length):

qte\_word = qte\_word + renpy.random.choice(abc)

next\_k = qte\_word[0]

renpy.restart\_interaction()

# натиснення чергової потрібної кнопки, переходимо до іншої

def next\_key():

global qte\_word, next\_k

qte\_word = qte\_word[1:]

next\_k = ""

if qte\_word:

next\_k = qte\_word[0]

renpy.restart\_interaction()

NextKey = renpy.curry(next\_key)

qteInit = renpy.curry(qte\_init)

# сам екран гри

screen scr\_qte(word="", time=5.0, length=5):

# ініціалізація

on 'show' action qteInit(word, time, length)

modal True

if qte\_word:

# зменшуємо час виділений на гру, і провіряємо чи не вийшов він. якщо вийшов - програш

timer 0.01 repeat True action [SetVariable("qteTime", qteTime - .01), If(qteTime <= .0, true=Return(False))]

# відображаємо яку кнопку потрібно натиснути

text next\_k.upper() align(.5, .5) size 96

# якщо щось потрібно натиснути, то питає клавіатуру

if len(next\_k) == 1:

key next\_k action NextKey()

else:

# всі кнопки натиснули - перемога

timer .1 action Return(True)

# шкала часу

bar value StaticValue(qteTime, qteMaxTime) align(.5, .1) xmaximum 600

# міні-гра починається тут

label minigame2:

scene bg college\_knb

"Натискайте на кнопки на клавіатурі, які будуть показані на екрані. Потрібно натиснути 5 кнопок щоб виграти."

$ res = renpy.call\_screen("scr\_qte", word="balda", time=3.0, \_layer="master")

# $ res = renpy.call\_screen("scr\_qte", time=4.0, length=4, \_layer="master")

# $ res = renpy.call\_screen("scr\_qte", \_layer="master")

if res:

centered "{size=72}Перемога!"

else:

centered "{size=72}Програш..."

return

* Карта:

#карта

screen map1:

imagemap:

ground "images/map/map\_ground.png"

idle "images/map/map\_idle.png"

hover "images/map/map\_hover.png"

#в лапках перші два значення відображають верхній лівий кут активної області

#наступні два це довжина і ширина області

#clicked означає що якщо на область натиснути, то спрацює означает Return()

#яке прийме значення яке вказане в лапках

hotspot (116, 67, 285, 185) action Return("college") alt "college"

hotspot (485, 77, 38, 172) action Return("lavka") alt "lavka"

hotspot (1058, 40, 190, 98) action Return("park") alt "park"

screen map2:

imagemap:

ground "images/map/map\_ground.png"

idle "images/map/map\_idle.png"

hover "images/map/map\_hover.png"

hotspot (218, 433, 533, 500) action Return("stolova") alt "stolova"

label example\_label:

call screen map1

#наш результат минулого вибору в меню, завдяки Return()

#був збережений в хранилищі \_return

$ result = \_return

#тепер результат дій кліку буде залежати від вибору який ми зробили раніше

if result == "college":

jump choisel\_kabinet1

elif result == "lavka":

jump choisel\_lavka1

elif result == "park":

jump choisel\_park1

elif result == "stolova":

call screen map2

jump stolova\_yes