Звіт з практики. Гра «Diamond Rush»

Структура даних

Скорочення: екземпляри класу – е. к.

У програмі для різних цілей було використано масиви та колекції. Об’єкти, що створюються безпосередньо під час виконання програми, зберігаються в колекціях, а саме в ArrayList, перевагою якого є швидкодія під час довільного доступу до елементів, зручність видалення та додавання нових об’єктів до колекції (у порівнянні з масивами). Об’єкти, кількість яких відома заздалегідь і залишається незмінною в процесі виконання програми, розташовуються в масивах.

*Масиви:*

Кожен рівень гри – це двовимірний масив клітинок, е. к. Cell. Цей масив міститься в е.к. Level, який заповнюється в класі – карті, Maps, і використовується в багатьох класах програми. Level також має масив е. к. Checkpoint. Е. к. Checkpoint містить двовимірний масив е. к. Cell для відновлення певної ділянки карти та масив цілих чисел additionalSegment для виділення додаткової ділянки. Для створення інших об’єктів у лабіринті також використовуються масиви. Ламка стіна, е. к. BreakableWall, використовує масив точок Point для пошуку на карті інших таких самих стін. Скриня, е. к. Chest, містить масив картинок, за допомогою ітерації по якому відбувається анімація відкривання скрині. Також у скрині є масив об’єктів, які наслідуються від абстрактного класу ThingInChest і лежать в ній. Подвійні двері, е. к. DoubleDoor, містять масив змій, е. к. Snake, яких треба вбити, щоб подвійні двері відчинилися. Пастки, об’єкти, що імплементують інтерфейс Trap, мають масиви картинок, які використовуються залежно від направлення і стану об’єкту. Вогняна пастка, е. к. FireTrap, має також boolean масив станів вогню. У героя Boy є 16 масивів картинок, за допомогою ітерації по яким відбувається анімація руху хлопчика. Не тільки об’єкти в лабіринті, а й карта рівнів, е. к. MapPanel, містить масив кнопок-переходів на рівні, е. к. LevelUI, та масиви x і y рисочок між цими кнопками. У класі для збереження прогресу, ProgressStorage, використовуються масиви стрічок для зручної роботи з контентом файлу.

*Колекції ArrayList:*

Ламка стіна, е. к. BreakableWall, використовує колекції клітинок, е. к. Cell, для швидкого додавання і зберігання інших таких самих стін поблизу на карті. У е. к. Checkpoint є колекції предметів, що знаходяться на підконтрольній чекпоїнту території: колекція е. к. Chest та колекція пасток. При малюванні поточного рівня в е. к. PlayPanel створюється колекція секретних стін, е. к. SecretWall, для того, щоб малювати їх після промальовування інших об’єктів.

Опис методів та класів

* ***PlayPanel***: клас наслідується від JPanel та імплементує KeyListener. Це ігрова панель, на якій малюються та взаємодіють всі об’єкти лабіринту. В класі реалізовані такі можливості: малювання об’єктів; рух хлопчика при натисканні на кнопки клавіатури одночасно по екрану, по карті та в масиві; переміщення видимої ділянки карти; рестарт, пауза та відновлення гри; оновлення зібраних хлопчиком речей в змінних та на верхній панелі, StatusBarPanel; віднімання у хлопчика енергії; застосування чекпоїнтів; закінчення рівня. Написані boolean методи перевірок типу itIsObject() для всіх об’єктів, що можуть знаходитись в клітинці, та деякі інші перевірки простору для каменів та хлопчика.
* ***ProgressStorage***: клас для збереження прогресу в файл та його відновлення. Має приватний статичний файл Progress.txt, куди записуються дані по кожному рівню. Є статичні методи для отримання та оновлення контенту файлу; зміни прогресу та отримання даних по конкретному рівню; метод, щоб скинути прогрес.
* ***Boy***: клас дозволяє створювати хлопчика – героя, який має поля xInArray та yInArray, проте в масив клітинок Cell не додається. В класі є Images хлопчика та їхні масиви, за допомогою ітерації по яким відбувається анімація його руху. У хлопчика є методи: ходити та бити молотком вверх, вниз, вліво, вправо; штовхати камінь та відкривати двері срібним або золотим ключем вліво, вправо; тримати камінь над головою, відкривати скриню.
* ***Stone***: абстрактний клас імплементує інтерфейс Resettable для подальшої реалізації брили (каменя – пастки) та діаманта (нешкідливого каменя). Камінь має методи для падіння в різні сторони, поле змії типу Snake, якщо стоїть у неї на дорозі, метод checkSnake() для вбивання змії та власний таймер. В конструкторі Stone запускається таймер та додається слухач подій, який переміщує камінь як тільки для нього з’являється вільне місце. Прописано багато перевірок на колізії з іншими каменями при одночасному їх падінні.
* ***Rock***: клас імплементує інтерфейс Trap та наслідується від абстрактного класу Stone, дозволяє створювати брилу. Має методи beShovenLeft() та beShovenRight() для штовхання брили хлопчиком.
* ***Harmless***: інтерфейс для подальшої реалізації нешкідливих для героя об’єктів на карті.
* ***Diamond***: клас імплементує інтерфейс Harmless та наслідується від абстрактного класу Stone. Дозволяє створювати фіолетові діаманти, які може збирати герой.
* ***Tumbleweed***: клас імплементує інтерфейс Harmless і дозволяє створювати рослину перекотиполе, яка зникає, коли на неї наступити.
* ***Chest***: клас імплементує інтерфейс Harmless і дозволяє створювати скрині. Метод openChest() викликається при натисканні на цифру 1 і запускає анімацію відкривання скрині і діставання з неї предметів.
* ***ThingInChest***: абстрактний клас для подальшої реалізації об’єктів зі скрині. Має метод paintObject() і абстрактний метод disappear()для визначення дій при зникненні об’єкту з екрану.
* ***PurpleDiamond***: клас наслідується від абстрактного класу ThingInChest і дозволяє створювати фіолетовий діамант, який можна знайти в скрині. Метод disappear() збільшує кількість фіолетових діамантів у героя.
* ***RedDiamond***: клас наслідується від абстрактного класу ThingInChest і дозволяє створювати червоний діамант, який можна знайти в скрині. Метод disappear() збільшує кількість червоних діамантів у героя.
* ***SilverKey***: клас наслідується від абстрактного класу ThingInChest і дозволяє створювати срібний ключ, який можна знайти в скрині. Метод disappear() збільшує кількість срібних ключів у героя.
* ***GoldKey***: клас наслідується від абстрактного класу ThingInChest і дозволяє створювати золотий ключ, який можна знайти в скрині. Метод disappear() збільшує кількість золотих ключів у героя.
* ***Energy***: клас наслідується від абстрактного класу ThingInChest і дозволяє створювати енергію, яку можна знайти в скрині. Метод disappear() збільшує кількість енергії у героя.
* ***AnimatableImage*** – клас, який дозволяє полегшити анімацію кнопок на панелях гри, адже містить корисні методи (animate), що дозволяють зручно змінювати зображення через певний період часу.
* ***GameFrame*** – клас, що наслідується від JFrame, є основним вікном, у якому відбувається гра. У цього фрейму є приватні поля типу MainMenu, MapPanel, PuzzleMap, InfoDialog, StatusbarPanel. У класі реалізовано методи, які дозволяють показувати всі панелі за необхідності (showMap(), showPuzzleMap(), showInfoDialog()), показувати рівень та наступний рівень (showLevel(), showNextLevel()), оновлювати статус бар гри та панель з артефактом (updateStatusbarPanel(), updatePuzzleMap()).
* ***InfoDialog*** – клас, що наслідується від JDialog, імплементує MouseListener, та відповідає за виведення на екран інформації про гру. Містить приватне поле InfoPanel та JScrollPane, у яких і розміщується інструкція з використання програми.
* ***LevelUI*** – клас, що містить приватні поля типу PlayPanel, StatusbarPanel, GameFrame та відповідає за створення рівня гри. Містить метод перезапуску рівня.
* ***MainMenu*** – клас, що наслідується від JPanel, імплементує MouseListener, слугує для відображення кнопок головного меню на екрані та переведення користувача до інших панелей гри.
* ***MapPanel*** – клас, що наслідується від JPanel, імплементує MouseListener, відображає карту з рівнями гри, дозволяє користувачеві переходити на різні рівні гри, до головного меню та панелі з артефактом.
* ***PauseMenuDialog*** – клас, що наслідується від JDialog, імплементує MouseListener, відповідає за взаємодію з користувачем під час паузи гри. Дозволяє переходити на карту рівнів, перезапускати рівень, вмикати-вимикати музику, зупиняти гру, продовжувати гру.
* ***PuzzlePanel*** – клас, що наслідується від JPanel, імплементує MouseListener, відповідає за відображення панелі з артефактом (з його частинами). Дозволяє користувачеві переходити до головного меню або до карти. На цій панелі виводиться привітання з перемогою, коли артефакт повністю зібрано.
* ***StatusBarPanel*** – клас, що наслідується від JPanel, імплементує MouseListener, відповідає за відображення поточної кількості зібраних ключів, діамантів, рівень енергії, поточний рівень. Дозволяє користувачеві використовувати контрольну точку, зупиняти гру.
* ***Util*** – клас, що містить статичні корисні методи для роботи програми: getFont(String path), wait(int miliseconds, ActionListener actionListener), setWalls(Cell[][] level), які використовуються для отримання шрифту з файлу, затримки виконання операції та встановлення бортиків для стін відповідно.
* ***Values*** – клас, що містить незмінні значення, необхідні для коректної роботи гри: координати об’єктів, їхні розміри.
* ***Checkpoint*** – клас, що імплементує інтерфейс Block, слугує для відновлення карти після використання контрольної точки в грі. Містить метод, що повертає відновлений масив об’єктів getRestoredMatrix().
* ***Resettable*** – інтерфейс, що імплементується об’єктами, які потрібно повернути в початковий стан після рестарту або використання контрольної точки.
* ***BlockedDoor*** – клас, що імплементує інтерфейс Block та відповідає за створення дверей, які неможливо відкрити.
* ***DiamondDoor*** – клас, що імплементує інтерфейс Block та відповідає за створення діамантових дверей, містить методи оновлення поточної кількості діамантів setNumberOfDiamonds(int number).
* ***DoorWithKeyHole*** – клас відповідає за створення дверей з замком. Містить два внутрішніх класи GoldDoor та SilverDoor, що імплементують інтерфейс Block, Resettable, містять методи openTheDoor().
* ***DoubleDoor*** – клас, що відповідає за створення подвійних дверей, що відчиняються після знищення всіх ворогів. Містить два внутрішніх класи LeftDoor, RightDoor, що імплементують інтерфейс Block, Resettable, містять метод взаємодії із гравцем interact(PlayPanel playPanel, Boy boy).
* ***PressMechanism*** – клас, що відповідає за створення дверей з кнопкою, містить два внутрішні класи PressPanel та Door, що імплементують інтерфейс Block, містять метод взаємодії із гравцем interact(Cell[][] matrix, int positionOfBoyInArrayX, int positionOfBoyInArrayY).

**Розподіл ролей**  
*Джосан Володимир:*

* Створення ієрархії об’єктів на карті.
* Створення пустої карти (без пасток та інших об’єктів).
* Створення таких пасток, як “Скорпіон”, “Змія”, ”Вогняна пастка”, їхній рух.
* Поведінка об’єкту “Ламка стіна” .
* 4 рівень гри.

*Сітьков Ілля:*

* Створення інтерфейсу гри: головного меню, карти рівнів, меню паузи, панелі з поточною інформацією рівня.
* Пересування персонажа по карті.
* Створення таких об’єктів для карти, як «Двері з ключем», «Двері, що відчиняються натисканням кнопки», «Подвійні двері», «Діамантові двері», «Заблоковані двері».
* Створення панелі з артефактом.
* Створення вікна з інформацією про гру, створення зображень з інформацією про гру.
* Візуальний таймер для пасток та повідомлення під час гри.
* Реалізація контрольних точок на карті.
* Додавання бортиків до стін лабіринту.
* Пауза, перезапуск гри.
* Вигадування й написання 2-го рівня гри та вигадування й написання половини 5-го рівня.

*Матвієнко Ірина:*

* Створення персонажа, його рух.
* Створення об’єктів «Камінь» та «Діамант», їхній рух та колізії.
* Розміщення рослин «Перекотиполе» на карті.
* Реалізація об’єкту «Скриня» та 5-ти речей всередині: «Фіолетовий діамант», «Червоний діамант», «Срібний ключ», «Золотий ключ», «Енергія».
* Boolean методи перевірок типу itIsObject() для всіх об’єктів на карті.
* Збереження прогресу в файл.
* Вигадування й написання 3-го рівня гри та вигадування й написання половини 5-го рівня.

Спільні задачі:

* Пошук та виправлення помилок.
* Пошук картинок для об’єктів на карті.
* Обробка картинок у програмі Photoshop.
* Створення 1 рівня гри.
* Написання звіту.

**Опис гри**

**Diamond Rush**. Наша версія гри відрізняється від оригінальної новою локацією із назвою “Єгипетська піраміда”, а також новими об’єктами та пастками, з якими персонаж повинен або може взаємодіяти. Більше того, на відміну від оригінального Diamond Rush`у, у нашій грі можна зібрати артефакт, за умови ідеального проходження усіх рівнів (детальніше про артефакт та рівні читайте далі).

П’ять рівнів складності гри – це п’ять карт, які користувач може пройти. Кожна наступна карта складніша за попередню.

Щоб пройти рівень, користувач повинен знайти вихід із лабіринту. Проте це тільки мінімальна умова проходження рівня. Щоб отримати частинку артефакту, персонаж повинен зібрати усі діаманти, які розкидані по карті.

У чому ж складність цієї гри, спитаєте Ви? На кожному рівні є пастки, які заважатимуть пересуватися лабіринтом. Якщо гравець був неакуратний та потрапив в одну з них, його герой втрачає енергію, необхідну для продовження гри. Також під час проходження карти потрібно бути дуже обачним, щоб випадково себе не заблокувати. Адже тоді єдиним виходом буде скористатися checkpoint’ом, який коштує багато енергії. Отже, як можна було здогадатись, енергія досить цінний ресурс, саме тому потрібно завжди ретельно обмірковувати наступні кроки. Окрім цього, треба бути уважним, адже в лабіринті є замасковані переходи, що ведуть до таємних кімнат. Саме там найчастіше заховані найбільш цінні діаманти – червоні. Тепер усе здається не таким простим, чи не так?

Постановка задачі

Написати гру «Diamond Rush».

1. Завантажте опис проекту.

2. Визначення основних структур та процедур. Побудувати діаграму класів, що будуть використовуватися в проекті.

3. В процесі роботи використовувати Систему контролю версій — Git.

4. Створити малюнок, що побудований з використанням графічних бібліотек. Написати графічну програму, в якій присутня анімація. Написати інтерактивну графічну гру. Мінімум 5 різних дій від користувача. Мінімум 5 рівнів гри, кожен з яких складніший за попередній.

5. Має бути реалізований GUI (графічний інтерфейс користувача, дизайн).

6. Завантажити  інструкцію користувача

7. Завантажте фінальний проект або посилання на нього.

Проблеми та шляхи їх вирішення

1. Проблеми з ГітХабом, через що втрачалися чи неправильно об’єднувалися частини коду.

*Вирішення:* зберігати копію проекту на комп’ютері перед тим, як об’єднувати код, або по декілька разів писати те саме.

1. Створення багатьох зайвих таймерів, тому програма працювала некоректно.

*Вирішення:* звести кількість створюваних таймерів до мінімуму.

1. Падіння купи каміння: камені налазили один на одного.

*Вирішення:* багато перевірок простору у кожного каменя.

1. Відновлення чекпоїнтом рухомих об’єктів.

*Вирішення:* встановлювати всім змінним початкові значення, зупиняти таймери.

1. Ексепшени в кінці карти.

*Вирішення:* блокувати можливість руху хлопчика, коли він доходить до останніх дверей на рівні.

1. Зменшення та збільшення довжини маршруту змії залежно від перешкод на її шляху.

*Вирішення:* перевірки клітинки попереду на наявність перешкоди.