Національний університет «Києво-Могилянська академія»

ЛАБОРАТОРНА РОБОТА № 2

Роботу виконали:
Дорошенко Лука Олексійович
студент І курсу НаУКМА
Факультет Інформатики
Спеціальність: інженерія програмного забезпечення

Кулакевич Станіслав Віталійович
студент І курсу НаУКМА
Факультет Інформатики
Спеціальність: інженерія програмного забезпечення

Гнутов Олександр Володимирович
студент І курсу НаУКМА
Факультет Інформатики
Спеціальність: інженерія програмного забезпечення

ПОСТАНОВКА ЗАДАЧІ

Написати гру “Breakout” на Java, використовуючи бібліотеку ACM.

Вимоги до функціоналу:

- Інтерактивний графічний інтерфейс користувача для взаємодії з грою: головне меню, меню вибору рівня, екран гри, екрани перемоги та поразки
- Базові ігрові механіки гри “Breakout”: м’ячик, ракетка для відбивання м’ячика, що керується гравцем, цеглинки, які знищуються при зіткненні з м’ячем і “бусти”, які впливають на ігровий процес (ускладнюють чи створюють позитивний ефект)
- Реалізація фізичної симуляції м’яча, що відбивається від стін, цеглинок і ракетки
- Додаткові компоненти гри: системи рівнів, здоров’я, очок та бустів

РОЗПОДІЛ ЗАДАЧ

Дорошенко Лука:

- Реалізація внутрішньоігрової графіки
- Створення основних ігрових компонентів: м’ячик, цеглинки і ракетка
- Написання документації коду

Кулакевич Станіслав:

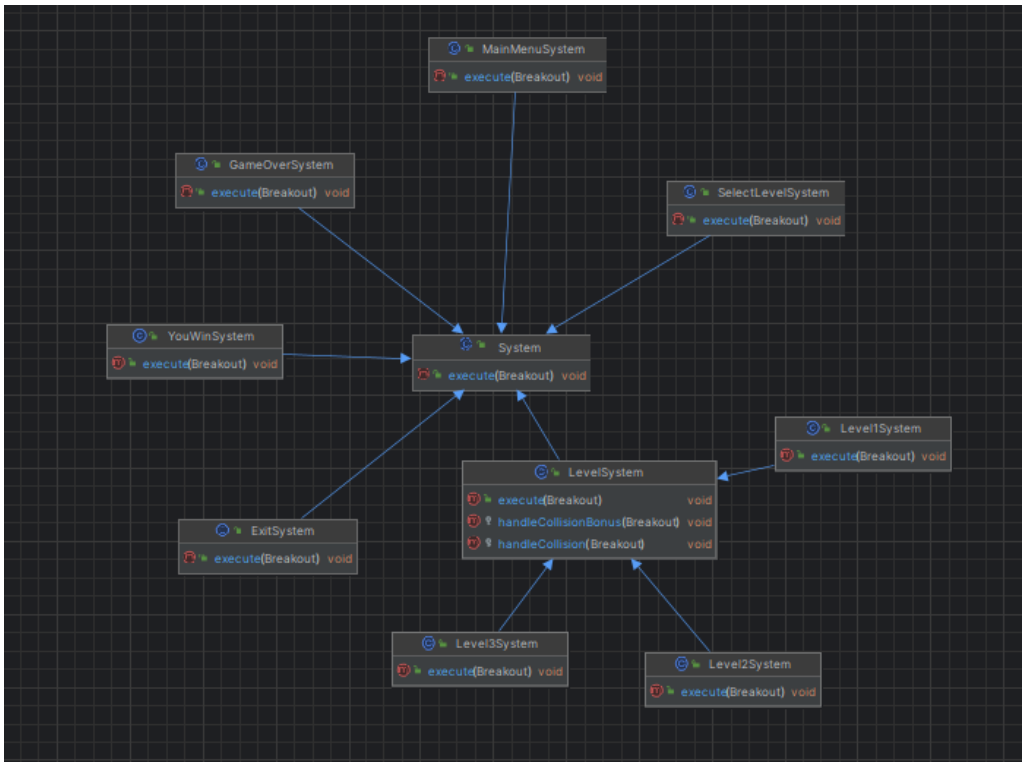
- Реалізація основної ігрової логіки, такої як система очок, здоров’я, система рівнів, та фізичних симуляцій: колізій фізичних об’єктів (м’ячик, цеглинки, ракетка), швидкості і прискорення динамічних твердих тіл (м’ячик)
- Реалізація додаткових “бустів”
- Внутрішньоігровий музичний та звуковий супровід

Гнутов Олександр:

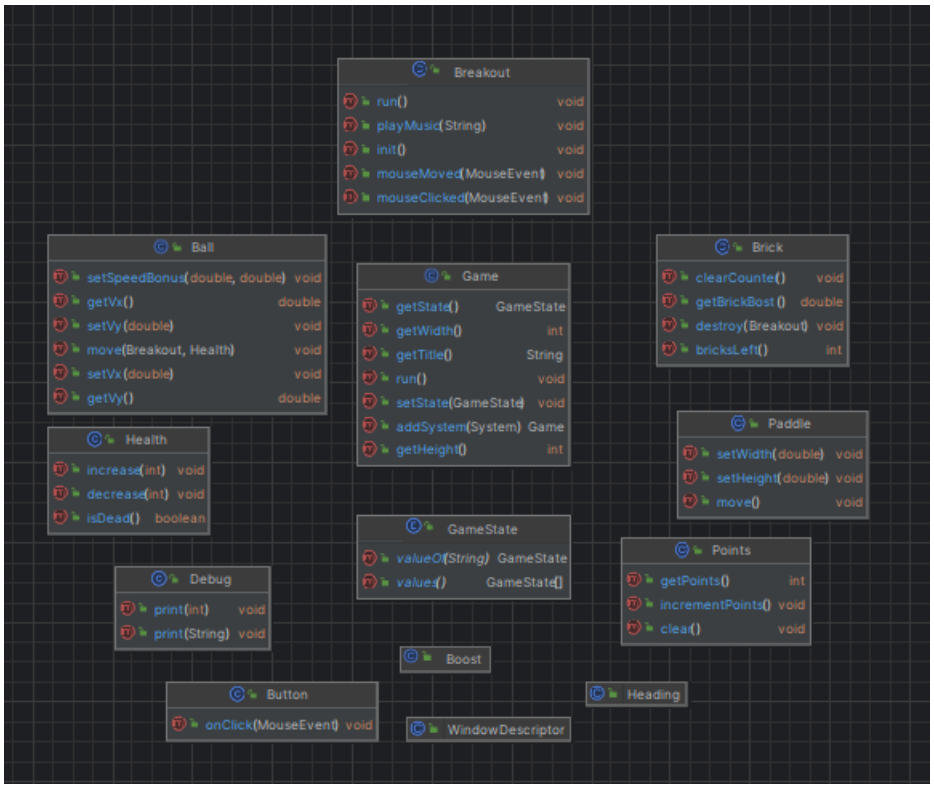
- Створення основної архітектури гри, що заснована на:
  - ігрових станах (стан, у якому знаходиться гра і який визначає її поведінку в цьому стані; наприклад, стани “головне меню”, “гра”, “екран ‘гру завершено’” тощо)
  - системах (процесах, які взаємодіють з окремими компонентами системи в залежності від ігрового стану)
- Реалізація основних компонентів графічного інтерфейсу, таких як заголовок, кнопка зміни стану, кнопка зміни рівня тощо

СТРУКТУРА ПРОГРАМИ

Реалізація парадигми внутрішньоігрових систем



Реалізація логічних класів гри



Гра “**Breakout**” складається з двох основних класів: логічного класу гри *Game*, який відповідає за логіку гри, взаємодію систем та ігрових

станів, ігровий цикл (“*game loop*”) та надає іншим компонентам доступ до параметрів гри (таких як розмір вікна та поточний ігровий стан), а також програмного класу *Breakout*, який відповідає за взаємодію з бібліотекою ACM (контекст вікна, рендеринг, події миші та клавіатури тощо), а також містить у собі константні дані для створення базових компонентів гри.

ОПИС МЕТОДІВ ТА КЛАСІВ

- клас **Breakout** — програмний клас гри, що на низькому рівні взаємодіє з інструментами бібліотеки ACM і пов'язує її з ігровою логікою:
  - методи **mouseClicked** і **mouseMoved** відповідають за взаємодію програмних компонентів (ігрових та інтерфейсних) з подіями миші (клік та рух всередині вікна);
  - метод **playMusic** відповідає за обробку та відтворення музики та звуків у програмі;
  - метод **init** — стандартний метод надкласу **GraphicsProgram**, в якому відбувається ініціалізація "слухачів" подій комп'ютерної периферії у програмі
  - метод **run** — стандартний метод надкласу; **GraphicsProgram**, в якому ініціалізується новий об'єкт логічного класу **Game** разом із основними ігровими системами-класами;
- клас **Game** — логічний клас гри, який керує системами та ігровими станами. Він створюється за допомогою додаткового класу даних **WindowDescriptor**, конструктор якого приймає параметри: назва вікна, ширина вікна, висота вікна і об'єкт **Breakout**.

  Для коректного роботи гри, створюється **тільки один** екземпляр класу **Game** з використанням таких методів:
  - метод **addSystem**, який додає нову ігрову систему в ігровий цикл; системи викликаються в порядку їх додавання
  - метод **run**, який запускає ігровий цикл;

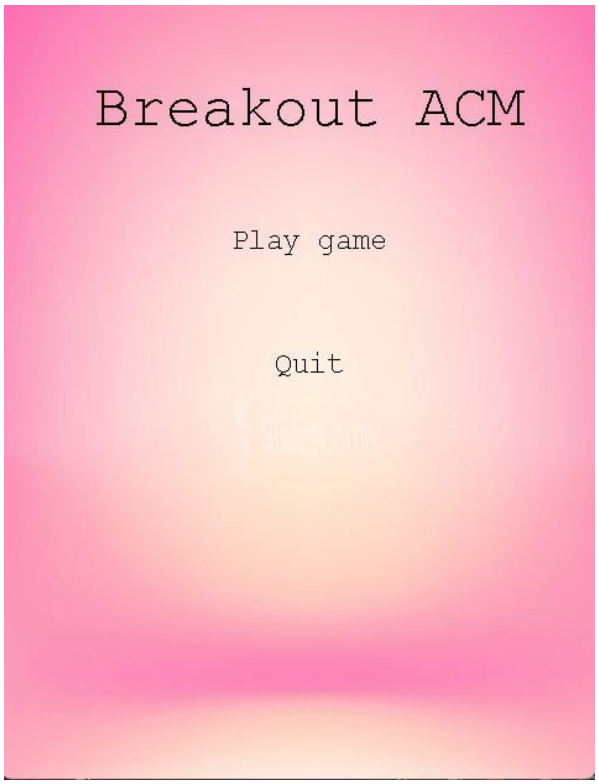
  Для роботи з даними екземпляру гри використовуються такі статичні методи:

    - методи **getState** і **setState** для керування поточним ігровим станом
    - методи **getWidth**, **getHeight** і **getTitle** для отримання інформації про вікно
- перелік **GameState** визначає ігровий стан. Гра може мати такі стани (варіанту переліку):
    - **MainMenu** — головне меню
    - **SelectLevel** — меню вибору рівня
    - **Level1** — рівень 1
    - **Level2** — рівень 2
    - **Level3** — рівень 3
    - **GameOver** — екран “гра закінчена”
    - **YouWin** — екран “ви перемогли”
    - **Exit** — особливий стан “вихід з гри”, який зупиняє ігровий цикл, робить очищення екрану та зупиняє роботу екземпляру програми **Breakout**.
- абстрактний клас **System** допомагає реалізувати парадигму “ігрових систем”, що обробляють окремі компоненти гри в різних ігрових станах, що надаються екземпляром класу **Game**. Він складається з методу **execute**, що приймає в якості параметру екземпляр програми **Breakout**, що дозволяє через системи напряму взаємодіяти з низькорівневими компонентами гри та інструментами бібліотеки ACM. Системи можна поділити на такі групи:
    - Системи GUI:
        - **MainMenuSystem** — система інтерфейсу головного меню
        - **SelectLevelSystem** — система інтерфейсу меню вибору рівня

        - **YouWinSystem** — система інтерфейсу екрану перемоги
        - **GameOverSystem** — система інтерфейсу екрану програшу
    - Системи ігрового процесу:
        - **LevelSystem** — надклас ігрових систем рівнів, де ініціалізуються основні ігрові об'єкти. Він поділяється на підкласи систем, які відповідають ігровим рівням 1-3, де відбувається обробка цих об'єктів: **Level1System, Level2System, Level3System**
        - **ExitSystem** — ігрова система, в якій відбувається процес деініціалізації і завершення роботи програми
- Класи компонентів:
    - Компоненти інтерфейсу:
        - Заголовок (**Heading**)
        - Кнопка (**Button**) для зміни ігрового стану
        - Очки (**Points**) для відображення набраних гравцем очок на рівні
            - метод **clear** очищує кількість набраних очок
            - функція **getPoints** повертає кількість набраних очок
            - метод **incrementPoints** збільшує кількість очок на 10
        - Здоров'я (**Health**) — компонент, що відображає поточне здоров'я гравця
            - метод **decrease** зменшує здоров'я на певну кількість одиниць
            - метод **increase** збільшує здоров'я на певну кількість одиниць
            - функція **isDead**, що повертає, чи гравець мертвий (якщо здоров'я = 0)

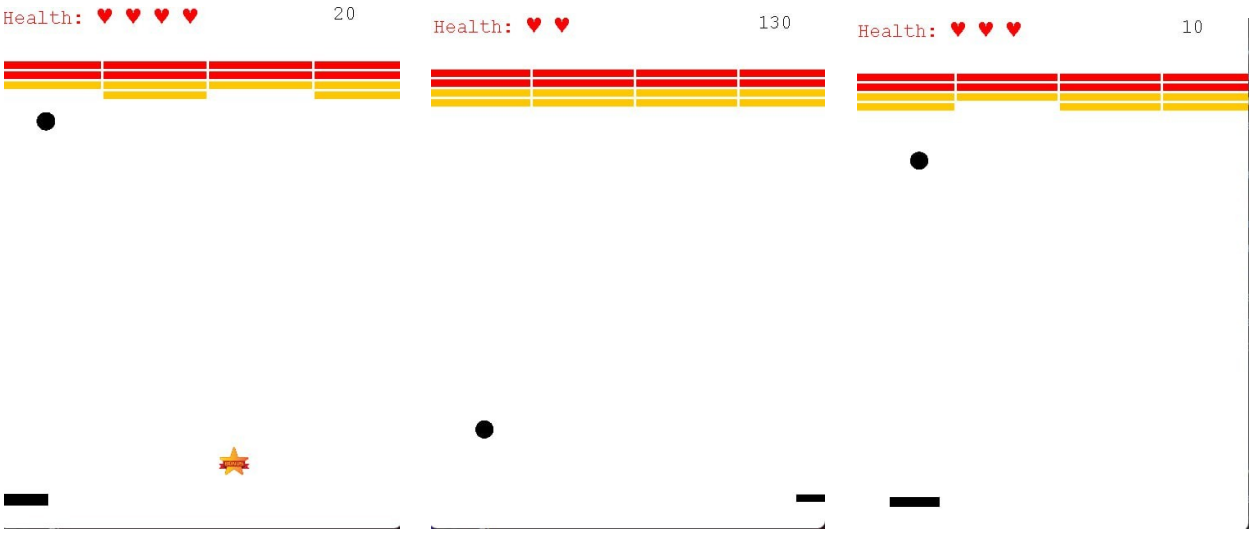
- Ігрові компоненти:
  - Цеглинка (**Brick**) — статичний фізичний об’єкт на ігровій сцені
    - метод **destroy** знищує поточний екземпляр об’єкту
    - метод **clearCounter** очищує лічильник цеглинок
    - функція **bricksLeft** повертає кількість цеглинок, що залишилися на сцені
    - функція **getBrickBost** повертає поточний буст цеглинки
  - Ракетка (**Paddle**) — ігровий персонаж, який керується користувачем за допомогою мишки:
    - метод **move** керує переміщенням ракетки по сцені
    - методи **setWidth** і **setHeight** задають розміри ракетки (використовуються в контексті бустів)
  - М’ячик (**Ball**) — динамічний об’єкт твердого тіла м’яча:
    - методи **getVx, setVx, getVy** і **setVy** використовуються для керування швидкістю м’яча по осях X і Y
    - метод **setSpeedBonus** задає додаткову швидкість
    - метод **move** керує переміщенням м’ячика по сцені
  - Буст (**Boost**) — компонент буста, що падає згори рівнів
- Допоміжні компоненти:
  - Налагоджувальник (**Debug**) використовується для виводу налагоджувальної інформації в консоль розробника

ІНСТРУКЦІЯ КОРИСТУВАЧА



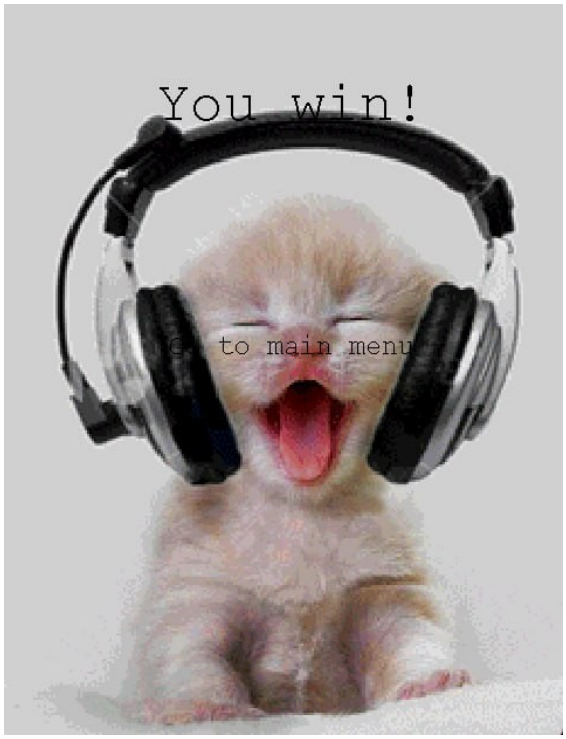
Головне меню гри. Містить 2 інтерактивні кнопки: “Play game”, що перемикає гру у стан вибору рівня і “Quit”, яка перемикає гру у стан завершення програми.

Меню вибору рівня. Містить 3 інтерактивні кнопки вибору одного з трьох рівнів, які різняться за складністю.



Рівні 1-3. Містять ігрові об’єкти (м’ячик, ракетка та цеглинки) та компоненти графічного інтерфейсу (лічильники здоров’я і очок,

показник номеру рівня та кількості цеглинок, що залишились). Складність відрізняється кількістю життів і початковою швидкістю руху м’яча.



Екрани програшу та виграшу. Якщо кількість життів зменшується до 0 (якщо м’ячик забагато разів вдарявся о підлогу), то гравець програє і йому відображається екран програшу. Інакше, якщо ж гравець знищує всі цеглинки, то він виграє, і йому показується екран виграшу.

ПРОБЛЕМИ, ЯКІ ВИНИКАЛИ ТА ШЛЯХИ ЇХ ВИРІШЕННЯ

1. **Проблема:** Застрягання м’яча в ракетці.

   Вирішення: Створення змінної, що визначає, чи може м’ячик зіткнутися з ракеткою
2. **Проблема:** Проходження м’яча крізь центр ракетки.

   Вирішення: Оскільки одночасно відбувається 4 перевірки зіткнення м’яча з ракеткою за усіма його точками, то коли він дотикається ракетки двома точками одночасно (в центрі ракетки), то напрямок руху змінюється двічі; отже, вважати валідною лише першу перевірку дотику.
3. **Проблема:** Невизначена поведінка (UB, undefined behaviour) при запуску різних рівнів гри.

   Вирішення: Для систем різних рівнів (**Level1System**, **Level2System** тощо) визначальним був встановлений стан 1 рівня (**GameState.Level1**), того усі системи рівнів запускались одночасно, що давало невизначену поведінку.

ПРОГРАМНИЙ КОД

Початковий код та документація знаходиться в репозиторії:

https://github.com/konceptosociala/BreakoutLab