TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN
----------o0o----------

BÀI TẬP LỚN MÔN HỌC
CÔNG NGHỆ JAVA

Đề tài : Lập trình Game Tower Defense bằng Java

Giảng viên: Đào Thị Lệ Thủy

Sinh viên: Nguyễn Hà Kiên

Hà Nội, tháng 4 năm 2024

LỜI NÓI ĐẦU

Công nghệ thông tin (CNTT) ngày càng có vai trò quan trọng trong cuộc sống hằng ngày của ta. Việc ứng dụng CNTT vào các lĩnh vực trong đời sống giúp công việc được tiến hành nhanh chóng và hiệu quả hơn.

Lập trình game Tower Defense không chỉ là một thách thức kỹ thuật, mà còn là một cơ hội tuyệt vời để khám phá và áp dụng những kiến thức lập trình Java vào một dự án thực tế. Trong bài báo cáo này, chúng ta sẽ đi sâu vào quá trình phát triển một trò chơi Tower Defense từ đầu đến cuối, bằng cách sử dụng ngôn ngữ lập trình Java.

Tower Defense là một thể loại phổ biến trong làng game thế giới, thu hút người chơi bằng sự kết hợp giữa chiến thuật, xây dựng và hành động. Chúng ta sẽ khám phá cách xây dựng một hệ thống game linh hoạt, bao gồm các yếu tố như đồ họa, cơ chế di chuyển và tương tác với người chơi. Trong bài báo cáo này, chúng ta sẽ thảo luận về các khía cạnh kỹ thuật của lập trình game, bao gồm quản lý đối tượng, xử lý va chạm, và thậm chí là cơ chế AI đơn giản để điều khiển các quái vật. Chúng ta cũng sẽ nắm bắt các nguyên tắc thiết kế và làm việc với các thư viện đồ họa để tạo ra giao diện đồ họa hấp dẫn cho trò chơi.

Cuối cùng, thông qua bài báo cáo này, hy vọng chúng ta sẽ không chỉ có được cái nhìn tổng quan về quá trình phát triển game Tower Defense bằng Java, mà còn nhận thức sâu hơn về sức mạnh và linh hoạt của ngôn ngữ lập trình này trong việc xây dựng các ứng dụng thú vị và có giá trị.

Mục lục

I. Đặt vấn đề

Trong thời đại công nghệ ngày nay, việc phát triển trò chơi điện tử không chỉ là một lĩnh vực giải trí mà còn trở thành một lĩnh vực nghiên cứu và ứng dụng kỹ thuật rộng lớn. Trong số các thể loại game phổ biến, Tower Defense là một thể loại thu hút sự quan tâm đặc biệt từ cộng đồng game thủ.

Tower Defense là một dạng game chiến thuật thời gian thực, trong đó người chơi phải xây dựng các tháp phòng thủ và chiến đấu chống lại các đợt tấn công của kẻ thù. Sự kết hợp giữa yếu tố chiến thuật, xây dựng, và phản ứng nhanh nhạy tạo nên sức hấp dẫn đặc biệt của thể loại này.

Tuy nhiên, việc phát triển một trò chơi Tower Defense không phải là điều dễ dàng. Nó đòi hỏi kiến thức sâu rộng về lập trình, đồ họa và thiết kế game. Đặc biệt, trong môi trường lập trình Java, việc xây dựng một game Tower Defens đòi hỏi sự hiểu biết vững về các khái niệm cơ bản của ngôn ngữ này cùng khả năng sử dụng các thư viện và công cụ phát triển game phức tạp.

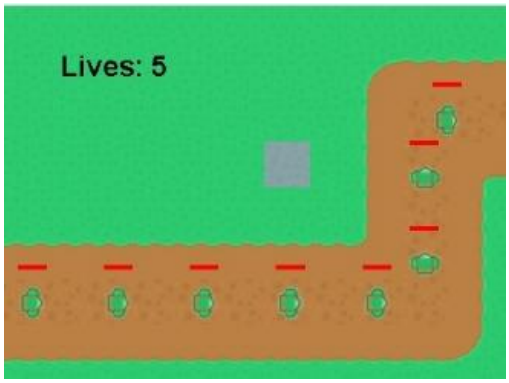
Bài báo cáo này sẽ đặt ra mục tiêu là hướng dẫn và mô tả quá trình phát triển một trò chơi Tower Defense bằng ngôn ngữ lập trình Java. Qua đó, chúng ta sẽ khám phá các thách thức, cơ hội và kỹ thuật quan trọng trong quá trình này, từ việc xây dựng cơ sở dữ liệu cho đến việc tối ưu hóa hiệu suất game. Đồng thời, bài báo cáo cũng nhấn mạnh vào sức mạnh của Java trong việc phát triển các ứng dụng game đa dạng và phức tạp.

II. Luật chơi

1. Mục tiêu của trò chơi

Người chơi phải bảo vệ một hoặc nhiều điểm chặn (thường là một cổng vào hay một tòa nhà quan trọng) khỏi sự tấn công của quái vật. Nếu quái vật thành công trong việc tiến vào điểm chặn, người chơi sẽ thua cuộc.

Cụ thể trong trò chơi này, người chơi sẽ có tổng cộng 5 mạng ứng với mỗi lần quái vật tiến vào điểm chặn, sau 5 lần sẽ thua cuộc. Nếu người chơi có thể chặn đứng hết tất cả quái vật, người chơi sẽ dành được chiến thắng.



2. Tháp phòng thủ (Tower)

Người chơi có thể xây dựng các tháp phòng thủ trên bản đồ để chặn đứng sự tiến công của quái vật. Mỗi loại tháp sẽ có đặc điểm riêng biệt như giá tiền, tầm bắn, tốc độ tấn công và sức mạnh.

Tháp phòng thủ sẽ chỉ được đặt vào những ô cho phép.

3. Quái vật (Enemies)

Các quái vật sẽ xuất hiện từ một điểm xuất phát trên bản đồ và cố gắng tiến về điểm chặn của người chơi. Mỗi loại quái vật sẽ có điểm máu và tốc độ di chuyển khác nhau.

4. Tiền (Gold)

Người chơi sẽ nhận được tiền khi tiêu diệt quái vật. Tiền được sử dụng để xây dựng và nâng cấp các tháp phòng thủ của người chơi. Người chơi cũng có thể nhận lại được tiền (70% giá gốc) khi bán tháp phòng thủ.

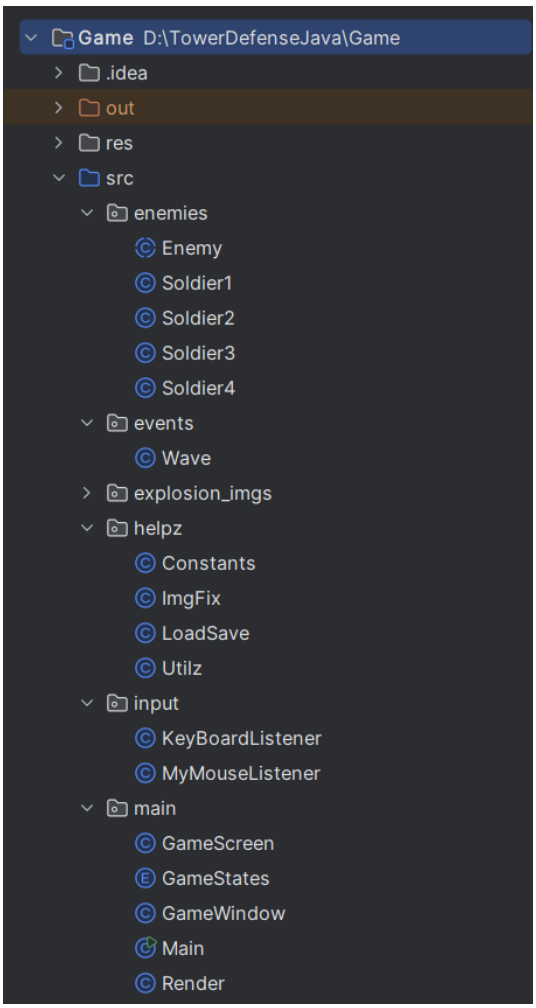
5. Đợt tấn công (Wave)

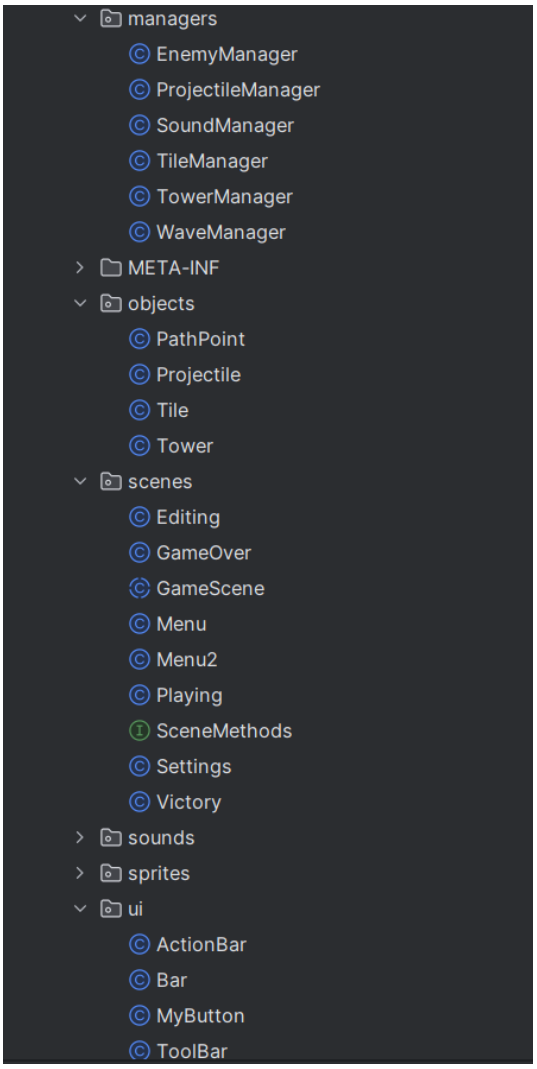
Trò chơi sẽ được chia thành nhiều đợt tấn công, mỗi đợt sẽ có số lượng quái vật và các loại quái vật khác nhau. Các đợt tấn công sau sẽ khó khăn hơn so với lượt trước đó.

6. Chiến thuật

Người chơi sẽ phải sử dụng chiến thuật để sắp xếp tháp phòng thủ, sử dụng tiền sao cho hợp lý giữa việc nâng cấp hay xây dựng thêm tháp để có thể chiến thắng các đợt tấn công.

III. Thiết kế các đối tượng





1. **Package enemies:** Lưu trữ dữ liệu của từng loại quái vật
   - **Class abtract Enemy:** Giúp tạo ra một cấu trúc chung cho các lớp con Soldier (1, 2, 3, 4)
   - **Class Soldier (1, 2, 3, 4):** Lưu thông tin cụ thể của từng loại quái vật
2. **Package events:** Lưu trự thông tin các sự kiện xảy ra trong game
   - Class Wave: Lưu thông tin về đợt tấn công tronng game
3. **Package helpz:** Chứa các lớp hỗ trợ chức việc lập trình game, vận hành và nâng cấp game

- **Class Constants:** Lưu trữ các hằng số
- **Class ImgFix:** Chứa các hàm chỉnh sửa hình ảnh
- **Class LoadSave:** Giúp làm việc với các file lưu màn chơi, cấp độ như xóa, sửa, tạo mới,…
- **Class Utilz (Utility):** Lưu trữ các hàm tiện ích như tính toán, chuyển đổi giữa mảng 1 chiều và 2 chiều,…

4. **Package input:** Lấy tương tác của người dùng
   - **Class KeyBoardListener:** Lấy tương tác của người dùng nhập từ bàn phím
   - **Class MyMouseListener:** Lấy tương tác của người dùng nhập từ chuột
5. **Package main:** Chứa các class vận hành, cốt lõi tạo ra game
   - **Class GameScreen:** Quản lý kích thước màn hình game
   - **Class enum GameStates:** Lưu các State có trong game
   - **Class Game:** Là Class chính của game, mở đầu và điều khiển toàn bộ quá trình trong game:
     - Khởi tạo và cài đặt
     - Điều khiển luồng chạy của game
     - Cập nhật trạng thái game
   - **Class Render:** Vẽ ra màn hình của các trạng thái khác nhau trong game
6. **Package object:** Lưu trữ thông tin các vật thể trong game
   - **Class PathPoint:** Đại diện cho một điểm trong đường đi của quái vật
   - **Class Projectile:** Đại diện cho viên đạn bắn ra từ Tower
   - **Class Tile:** Đại diện cho các loại khối vật thể có trong game
   - **Class Tower:** Đại diện cho Tower
7. **Package manager:** Quản lý các Object và sự kiện có trong game
   - **Class EnemyManager:** Quản lý hành đông, thông số của quái vật
   - **Class ProjectileManager:** Quản lý tất cả các loại đạn được tạo ra từ Tower
   - **Class SoundManager:** Quản lý âm thanh
   - **Class TileManager:** Quản lý các khối vật thể
   - **Class TowerManager:** Quản lý tất cả Tower được tạo ra

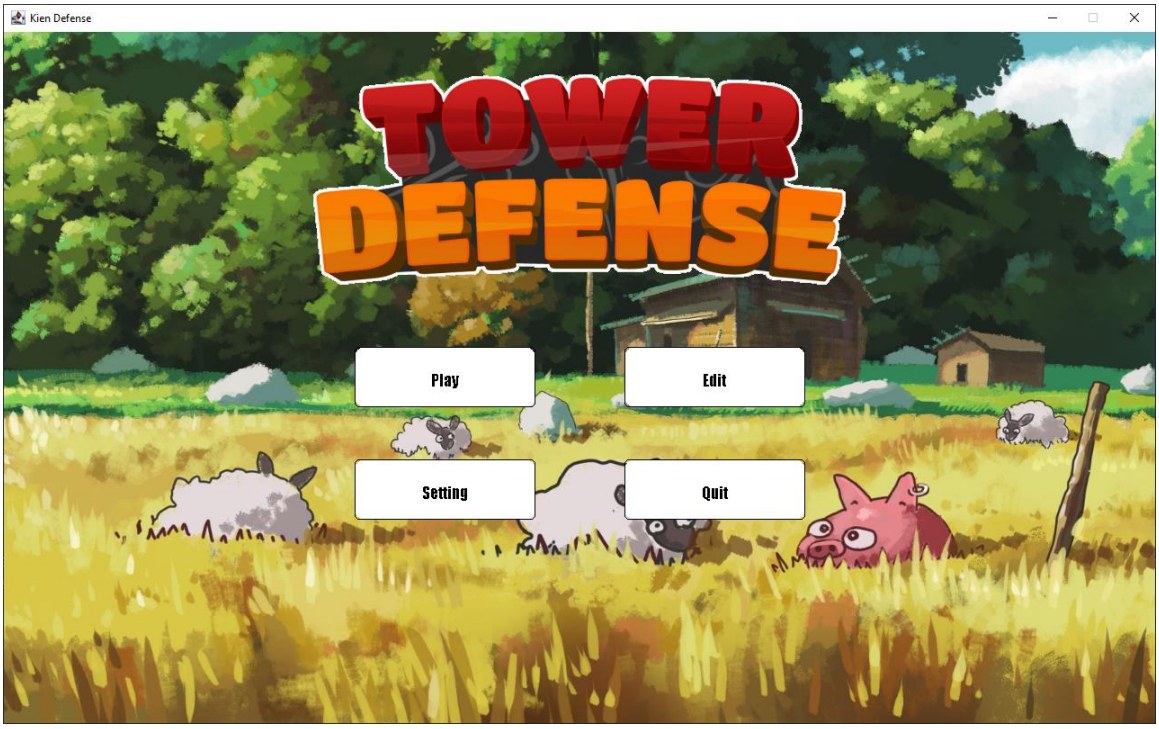
- **Class WaveManager:** Quản lý các đợt tấn công

8. **Package scenes:** Tổ chức, quản lý các scene có trong trò chơi
    - **Class abtract GameScene:** Tạo một cấu trúc chung cho các scene cụ thể
    - **Class Editing:** Màn hình chỉnh sửa, tạo ra màn chơi
    - **Class GameOver:** Xuất hiện khi người chơi thua cuộc
    - **Class Menu:** Màn hình menu, cũng là màn hình chính trong game
    - **Class Menu2:** Màn hình menu được sử dụng khi tạm dừng game trong lúc đang chơi
    - **Class Playing:** Màn hình chơi game
    - **Interface SceneMethods:** Tập hợp các phương thức mà mỗi scene phải thực hiện để xử lý sự kiện và tương tác của người chơi
    - **Class Settings:** Màn hình cài đặt game
    - **Class Victory:** Xuất hiện khi người chơi chiến thắng
9. **Package ui:** Quản lý các thành phần liên quan đến giao diện người chơi
    - **Class Bar:** Là lớp cơ sở cho Bar cụ thể trong các scene khác nhau
    - **Class ActionBar:** Cung cấp các nút thực hiện hành động như xây tháp, nâng cấp,…, cung cấp thông tin về màn chơi như mạng, số lượng quái vật, … trong scene Playing
    - **Class ToolBar:** Cung cấp các nút hành động trong scene Editing
    - **Class MyButton:** Thiết kế, tạo ra các nút có trong trò chơi
10. **Folder imgs:** Lưu trữ các hình ảnh sử dụng trong quá trình lập trình game
11. **Folder sounds:** Lưu trữ các âm thanh sử dụng trong game

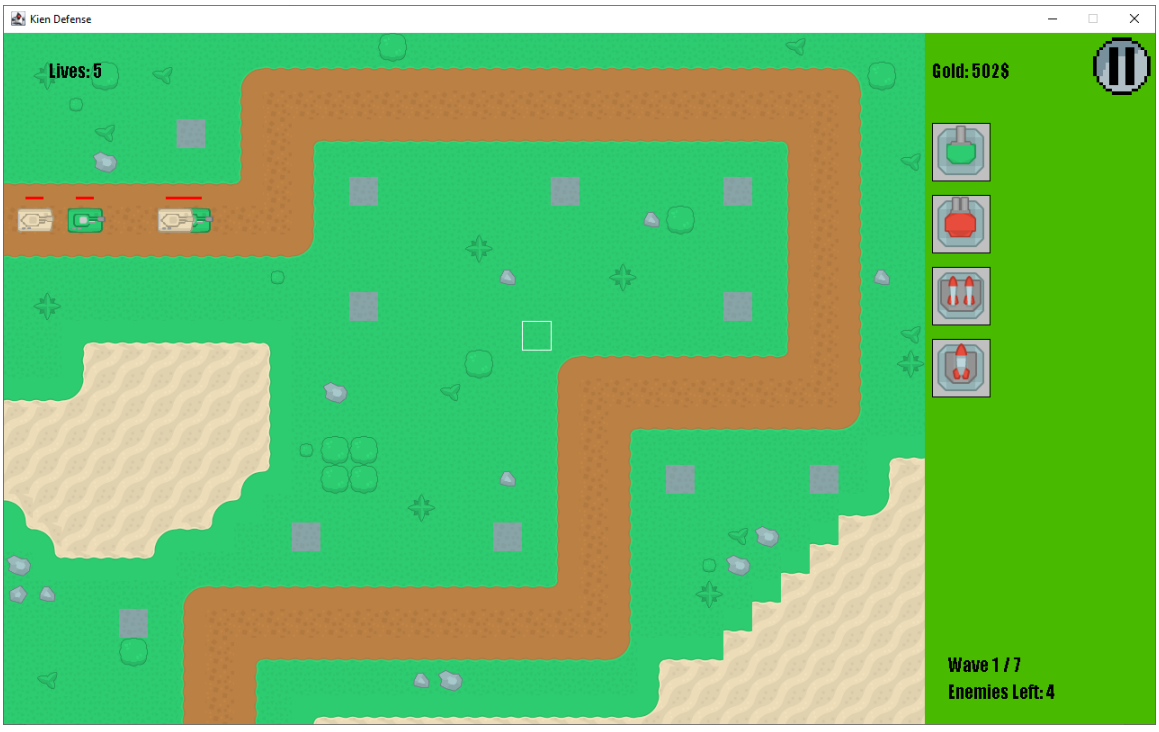
IV. Kết quả đạt được

4.1. Giao diện

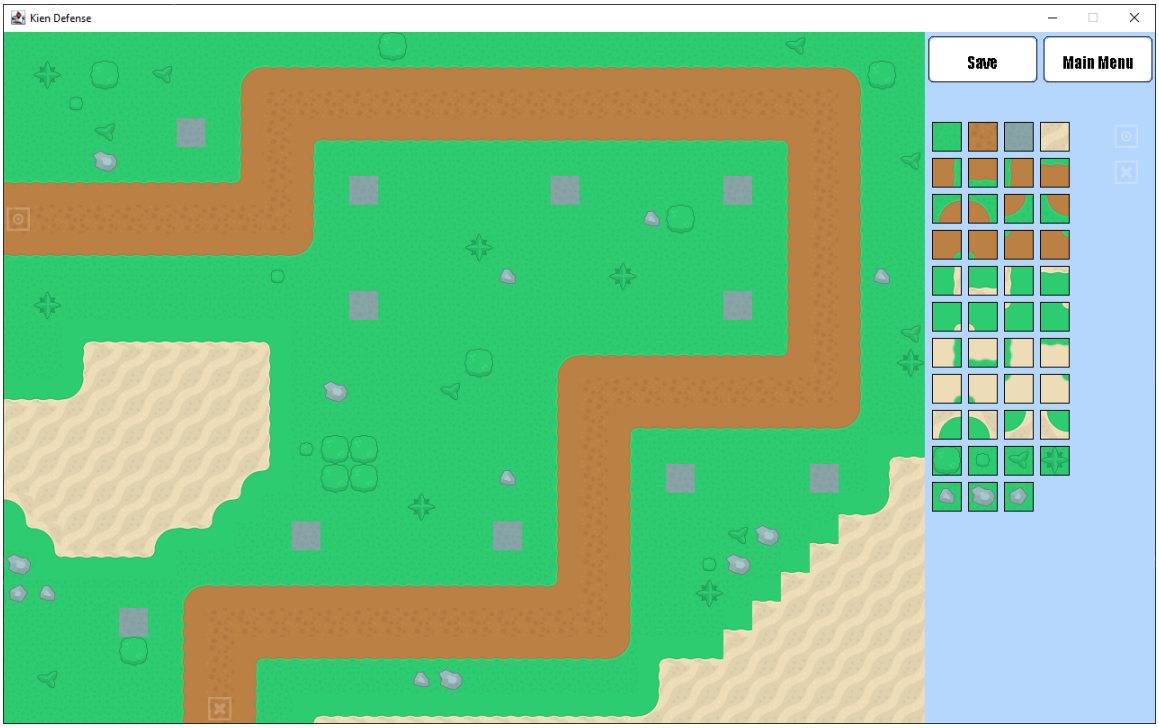
Màn hình chính:



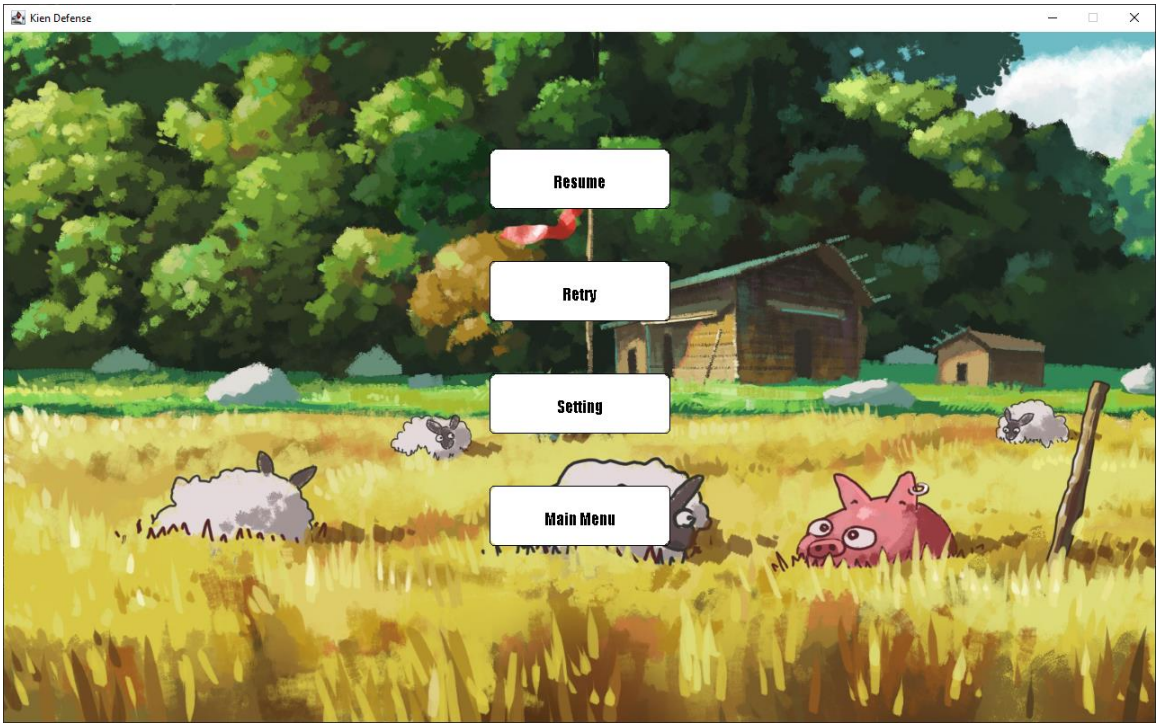
Màn hình chơi:



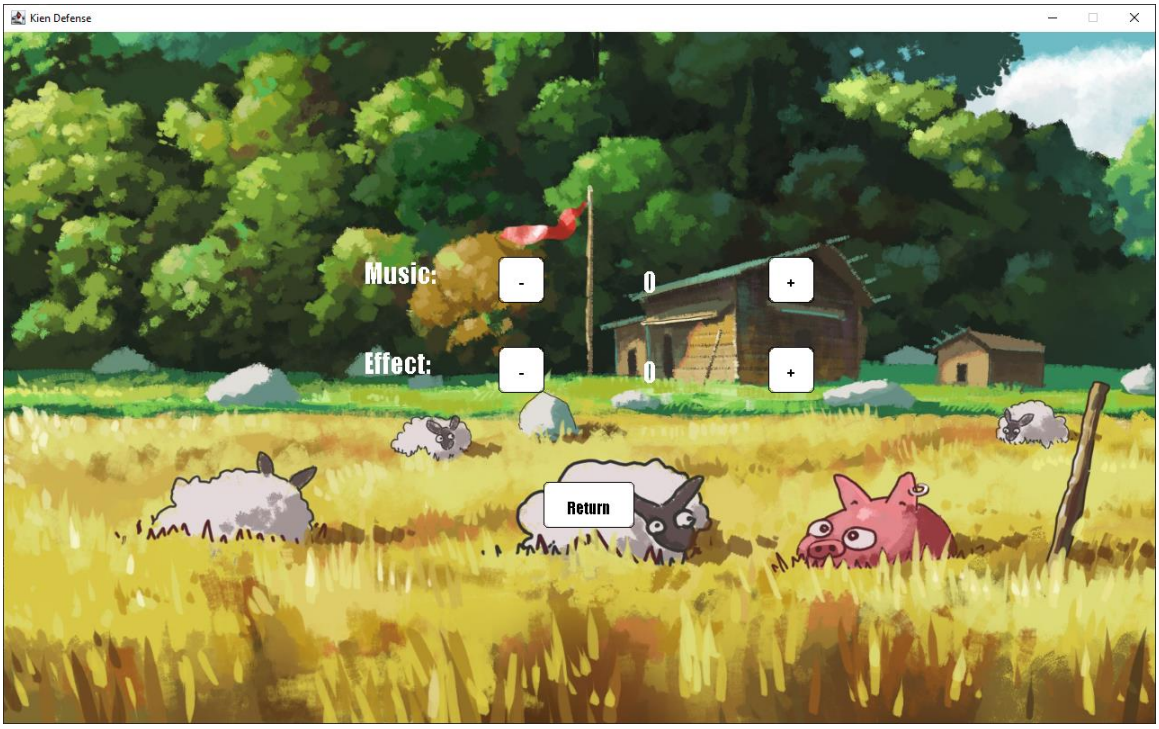
Màn hình chỉnh sửa màn chơi:



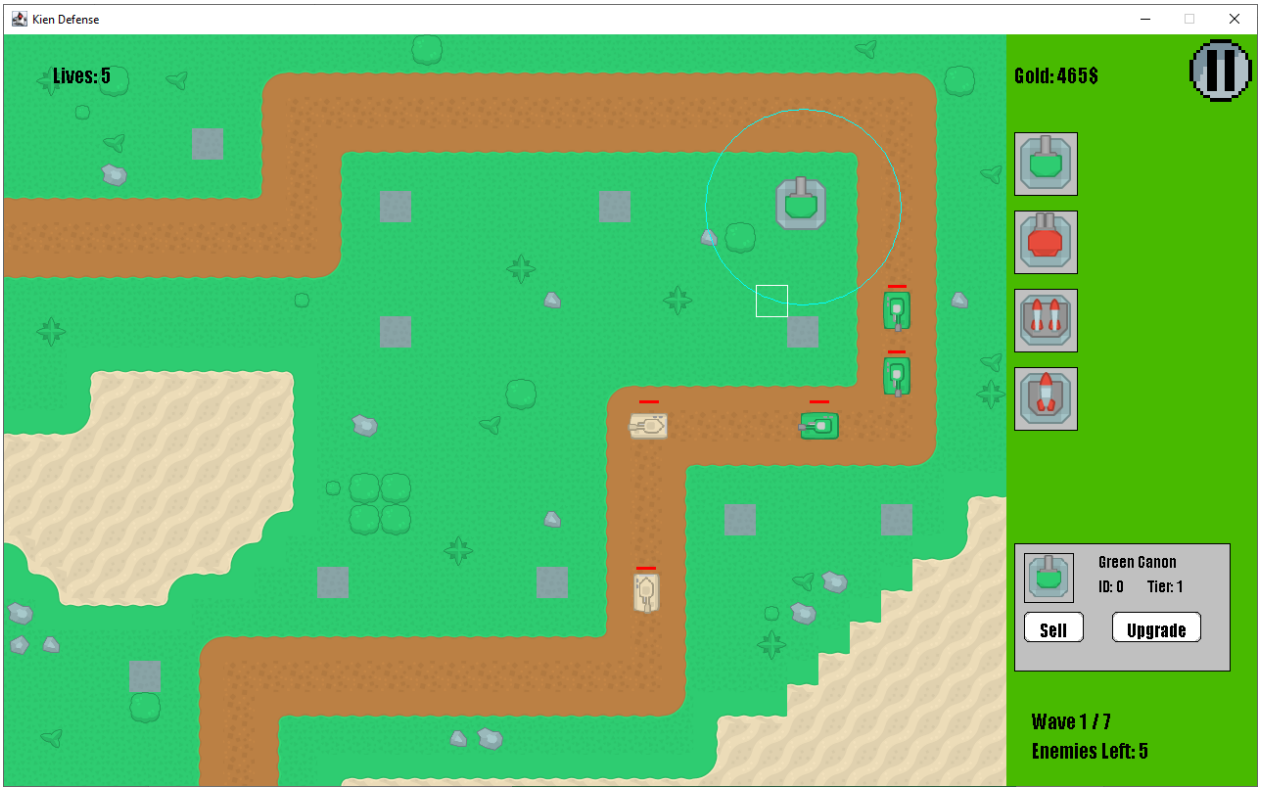
Màn hình tạm dừng game:



Màn hình cài đặt:

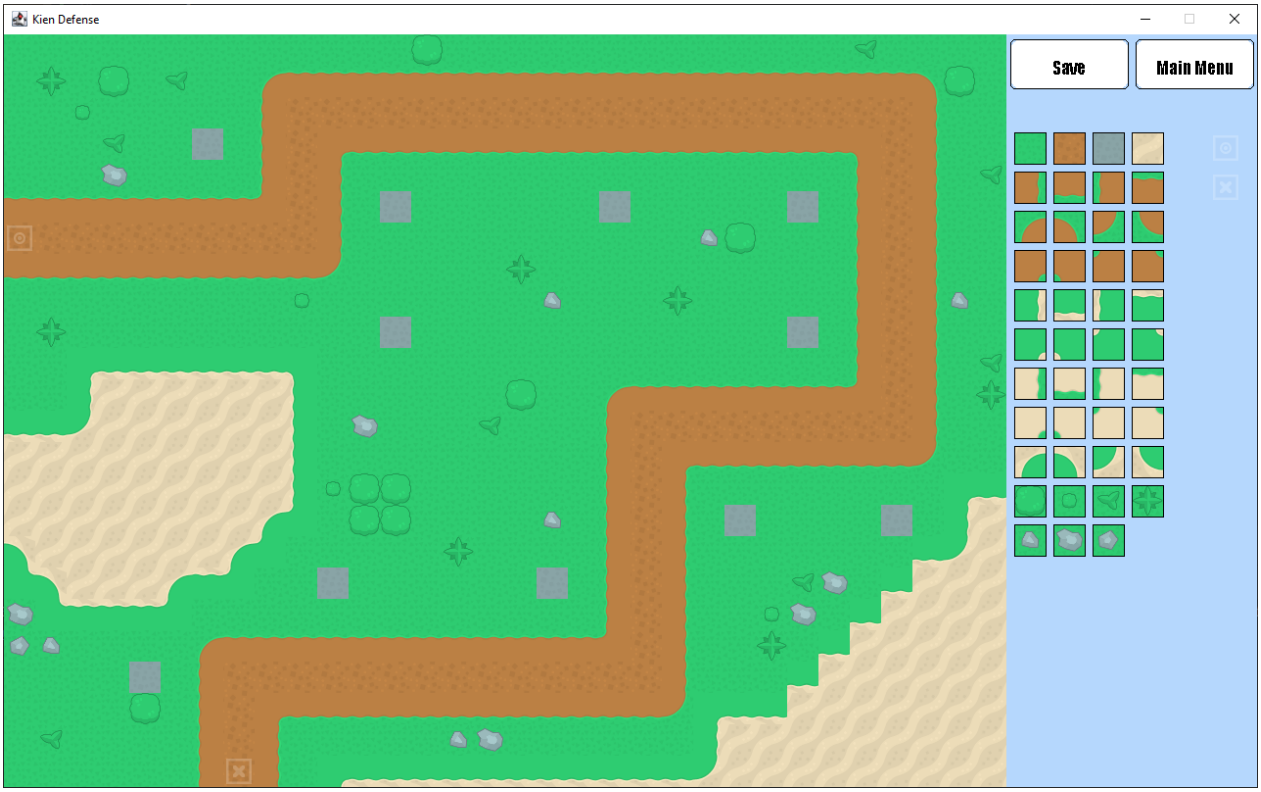


4.2. Hệ thống hiển thị thông số của tháp phòng thủ và thông tin máu của quái vật



4.3. Làm việc với tệp, save và load trong mục Edit

Người chơi có thể tùy ý chính sửa màn chơi của mình và lưu lại. Tất cả sẽ được lưu dưới dạng file text. Khi load màn chơi, dữ liệu trong file text sẽ được chuyển sang dạng mảng 2D để có thể dễ dàng làm việc.



4.4. Thuật toán giúp quái vật có thể tìm thấy đường đi

Trong game tower defense lần này, đường đi sẽ được kiểm tra và tạo sẵn từ trước, quái vật chỉ việc di chuyển theo con được đã được tìm sẵn từ điểm bắt đầu (start) tới điểm cuối (end), không còn phải tìm đường trong quá trình di chuyển.

```
public static int[][] GetRoadDirArr(int[][] lvlTypeArr, PathPoint start,
PathPoint end) {
    int[][] roadDirArr = new int[lvlTypeArr.length][lvlTypeArr[0].length];

    PathPoint currTile = start;
    int lastDir = -1;
    while (!IsCurrSameAsEnd(currTile, end)) {
        PathPoint prevTile = currTile;
        currTile = GetNextRoadTile(prevTile, lastDir, lvlTypeArr);
        lastDir = GetDirFromPrevToCurr(prevTile, currTile);
        roadDirArr[prevTile.getyCord()][prevTile.getxCord()] = lastDir;
    }
    roadDirArr[end.getyCord()][end.getxCord()] = lastDir;
    return roadDirArr;
}
```

Thuật toán sẽ giúp kiểm tra điểm và hướng đi trước đó để tìm điểm tiếp theo quái vật sẽ di chuyển tới. Nếu tìm thấy đường đi thì sẽ lấy làm điểm tiếp theo.

```
private static PathPoint GetNextRoadTile(PathPoint prevTile, int lastDir,
int[][] lvlTypeArr) {

    int testDir = lastDir;
    PathPoint testTile = GetTileInDir(prevTile, testDir, lastDir);

    while (!IsTileRoad(testTile, lvlTypeArr)) {
        testDir++;
        testDir %= 4;
        testTile = GetTileInDir(prevTile, testDir, lastDir);
    }

    return testTile;
}
```

```
private static boolean IsTileRoad(PathPoint testTile, int[][] lvlTypeArr) {
    if (testTile != null)
        if (testTile.getyCord() >= 0)
            if (testTile.getyCord() < lvlTypeArr.length)
                if (testTile.getxCord() >= 0)
                    if (testTile.getxCord() < lvlTypeArr[0].length)
                        if
(lvlTypeArr[testTile.getyCord()][testTile.getxCord()] == DIRT_TILE)
                            return true;

    return false;
}
```

Sau đó là kiểm tra lại hướng của đường đi từ điểm trước tới điểm vừa tìm được xem hướng đi có thay đổi không.

```
private static int GetDirFromPrevToCurr(PathPoint prevTile, PathPoint
currTile) {
    if (prevTile.getxCord() == currTile.getxCord()) {
        // Up or down
        if (prevTile.getyCord() > currTile.getyCord())
            return UP;
        else
            return DOWN;
    } else {
        // Right or left
        if (prevTile.getxCord() > currTile.getxCord())
            return LEFT;
        else
            return RIGHT;
    }
}
```

Thuật toán sẽ lặp lại cho tới khi nào kiểm tra và tìm thấy điểm cuối cùng (end).

```
private static boolean IsCurrSameAsEnd(PathPoint currTile, PathPoint end) {
    if (currTile.getxCord() == end.getxCord())
        if (currTile.getyCord() == end.getyCord())
            return true;
    return false;
}
```

V. Cải thiện và nâng cấp

5.1. Vấn đề hình ảnh

Giao diện được thiết kế đơn giản, hình ảnh chưa tốt, chưa có nhiều hiệu ứng bắt mắt.

5.2. Vấn đề cơ sở dữ liệu

Chưa thiết kế được cơ sở dữ liệu, chỉ có thể lưu và chỉnh sửa trên tệp.

5.3. Một số điều cần phát triển thêm

- Tạo ra nhiều màn chơi hơn, có màn hình chọn màn chơi.
- Hình ảnh đẹp hơn, bắt mắt hơn.
- Thêm nhiều loại quái vật và tháp phòng thủ.

VI. Tài liệu tham khảo

- Slide bài giảng của giảng viên **Đào Thị Lệ Thủy** – Đại học Giao thông vận tải
- https://www.youtube.com/@KaarinGaming
- https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
- https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
- https://researchgate.net/figure/Hypo-distance-between-two-distribution-functions-F-solid-line-and-G-dashed-line_fig1_317611947