

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



NGUYỄN VĂN KHÁNH ÂN - 51900475
HUỲNH GIA HUY - 51900798

**ĐỀ TÀI CUỐI KỲ MÔN PHÁT TRIỂN
TRÒ CHƠI**

**ĐỀ TÀI: XÂY DỰNG GAME KINGDOM
DEFENSE**

Người hướng dẫn
ThS. Vũ Đình Hồng

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Sau khi đã hoàn thành bài báo cáo này thì em muốn gửi lời cảm ơn chân thành đến với những người đã đồng hành và hỗ trợ em để em có thể hoàn thành bài báo cáo này một cách trọn vẹn, dù cho sự trợ giúp mà em đã nhận được là nhiều hay ít thì nó cũng đã góp phần lớn để bài báo cáo này có thể hay hơn và hoàn thiện hơn. Và đặc biệt em muốn gửi lời cảm ơn sâu sắc nhất tới thầy Vũ Đình Hồng, người đã hỗ trợ em rất nhiều cho bài báo cáo này, thầy đã không ngại thời gian mà hướng dẫn cho em rất tận tình, cho em rất nhiều kiến thức và định hướng quý báu trong quá trình nghiên cứu đề tài này. Nhờ có sự chỉ dẫn của thầy mà em đã học hỏi được các kiến thức cũng như các kỹ năng nghiên cứu và phân tích, em tin rằng những kỹ năng và kiến thức này sẽ vô cùng hữu ích cho tương lai phía trước của em. Một lần nữa em vô cùng cảm kích vì sự hỗ trợ của mọi người đã giành cho em, em chân thành cảm ơn mọi người.

TP. Hồ Chí Minh, ngày 17 tháng 12 năm 2023.

Tác giả

Nguyễn Văn Khanh An

Huỳnh Gia Huy

(Ký tên và ghi rõ họ tên)

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của ThS.Vũ Đình Hồng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 17 tháng 12 năm 2023.

Tác giả

Nguyễn Văn Khánh Ân

Huỳnh Gia Huy

(Ký tên và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH VẼ.....	4
CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	7
1.1 Lý do chọn đề tài	7
1.2 Bối cảnh game	7
1.2.1 <i>Cốt truyện.</i>	7
1.3 Các yêu cầu thực hiện đề tài	7
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	8
2.1 Transform.translate là gì?	8
2.2 Collider là gì ?	8
2.3 Animator là gì ?	8
2.4 Audio source là gì?	8
CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ.....	10
3.1 Sơ đồ UseCase:.....	10
3.2 Sơ đồ tuần tự (Sequence diagram) của từng chức năng:	11
3.2.1 <i>Start game:</i>	11
3.2.2 <i>Choose level</i>	12
3.2.3 <i>Pause game:</i>	12
3.2.4 <i>Continue:</i>	13
3.2.5 <i>Retry</i>	13
3.2.6 <i>Main menu:</i>	14
3.2.7 <i>Win level:</i>	14
3.2.8 <i>Next level:</i>	15

3.2.9 Xây dựng trụ phòng thủ:	15
3.2.10 Nâng cấp trụ phòng thủ:	16
3.2.11 Bán trụ:	16
3.2.12 Cửa hàng:.....	17
CHƯƠNG 4. TRIỂN KHAI VÀ THỰC NGHIỆM HỆ THỐNG	18
❖ Triển khai hệ thống bao gồm các chức năng sau.....	18
4.1 Thiết kế chức năng start game:	18
4.2 Thiết kế chức năng choose level:	19
4.3 Thiết kế Game-play:	20
4.3.1 Quái vật:	20
4.3.2 Các trụ phòng thủ:.....	27
4.3.3 Các loại đạn:.....	29
4.3.4 Thiết kế xây dựng các trụ phòng thủ, nâng cấp và bán trụ:	33
4.3.5 Pause Game:	39
4.3.6 Win Game:	40
4.4 Thiết kế cửa hàng:	42
4.5 Thiết kế cho phép người chơi chỉnh sửa cấu hình trong game:	44
4.6 Âm thanh tổng cho game	46
4.7 Màn hình chính của game	48
4.8 Màn hình chọn màn chơi	48
4.9 Màn hình một vòng chơi của game	49
4.10 PauseUI đơn giản của game	49
4.11 Màn hình game khi chơi thua.....	49

4.12 Khi chiến thắng một màn chơi	50
4.13 Menu cài đặt của game	50
4.14 Menu shop trong game	51
CHƯƠNG 5. KẾT LUẬN	52
5.1 Đã đạt được	52
5.2 Chưa đạt được	52
5.3 Hướng phát triển.....	53
TÀI LIỆU THAM KHẢO.....	54

DANH MỤC HÌNH VẼ

Hình 3.1 Sơ đồ usecase của game	10
Hình 3.2 Sơ đồ tuần tự chức năng Start game	11
Hình 3.3 Sơ đồ tuần tự chức năng choose level.....	12
Hình 3.4 Sơ đồ tuần tự chức năng pause game.....	12
Hình 3.5 Sơ đồ tuần tự chức năng continue game	13
Hình 3.6 Sơ đồ tuần tự chức năng retry	13
Hình 3.7 Sơ đồ tuần tự chức năng main menu	14
Hình 3.8 Chức năng win level	14
Hình 3.9 Sơ đồ tuần tự chức năng next level.....	15
Hình 3.10 Sơ đồ tuần tự chức năng xây dựng trụ phòng thủ.....	15
Hình 3.11 Sơ đồ tuần tự chức năng nâng cấp trụ.....	16
Hình 3.12 Sơ đồ tuần tự chức năng bán trụ	17
Hình 3.13 Sơ đồ tuần tự chức năng cửa hàng.....	17
Hình 4.1 Code chức năng start game	19
Hình 4.2 Code chức năng chọn level	20
Hình 4.3 Code đặt mục tiêu điểm đến cho quái.....	21
Hình 4.4 Code quái tự động di chuyển đến mục tiêu.....	21
Hình 4.5 Code chuyển mục tiêu của quái.....	22
Hình 4.6 Code quái tấn công thành	22
Hình 4.7 Code trừ máu của người chơi theo từng loại quái đánh vào thành	23
Hình 4.8 Code tạo khác đợt quái.....	24
Hình 4.9 Code xuất hiện UI khi chiến thắng màn chơi.....	24

Hình 4.10 Class đối tượng các đợt quái	25
Hình 4.11 Code hàm tạo ra các đợt quái	26
Hình 4.12 Hàm tạo ra từng con quái vào game	27
Hình 4.13 Code cập nhật vị trí mục tiêu.....	27
Hình 4.14 Code nòng súng theo vị trí của mục tiêu.....	28
Hình 4.15 Hàm Shoot() cho phép trụ bắn đạn	29
Hình 4.16 Đạn di chuyển đến mục tiêu và gây sát thương.....	29
Hình 4.17 Hàm đạn va chạm với mục tiêu	30
Hình 4.18 Hàm TakeDamage của enemy	31
Hình 4.19 Hàm làm chậm tốc độ di chuyển của quái	31
Hình 4.20 Hàm gây sát thương độc cho quái	32
Hình 4.21 Code đối tượng quản lý từng loại trụ	33
Hình 4.22 Class TurretHolder.....	34
Hình 4.23 Class NodeManager	35
Hình 4.24 Code nâng cấp trụ	39
Hình 4.25 Code tạm ngừng game	39
Hình 4.26 Code tiếp tục game.....	40
Hình 4.27 Code win game	41
Hình 4.28 Code cửa hàng mua các loại trụ.....	42
Hình 4.29 Code mua loại trụ mới trong cửa hàng.....	43
Hình 4.30 Hiện trụ đã mua trong game	44
Hình 4.31 Chính sửa resolution trong game.....	45
Hình 4.32 Các hàm chỉnh độ phân giải và phóng full màn hình	46

Hình 4.33 Code để lưu và chỉnh âm thanh trong game.....47

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Lý do chọn đề tài

Thể loại game thủ thành (Tower Defense) là một thể loại game chiến thuật được khá nhiều game thủ đón nhận do sự đơn giản trong lối chơi, nhưng cũng cần có có một số tư duy, chiến thuật nhất định để qua màn chơi. Game thể loại thủ thành đã có được sự thành công từ đầu những năm 2000, một số game chiến thuật mới hiện nay cũng dựa trên thể loại này để phát triển. Đây là lý do nhóm chọn đề tài này để làm.

1.2 Bối cảnh game

1.2.1 Cốt truyện

Ở một vương quốc yên bình, nơi người dân có kỹ năng trồng trọt rất cao, họ đã phát triển được nhiều loại cây phục vụ cho các nhu cầu trong đời sống hằng ngày. Đó cũng là điều mà mọi sinh vật khác mong muốn có, bọn chúng đã nhăm đến vương quốc này từ lâu, và khi không còn đủ khả năng để duy trì các nhu yếu phẩm hằng ngày, bọn quái vật bắt đầu cuộc xâm lăng của chúng đối với vương quốc nhằm xâm chiếm lấy các loại cây trồng và lương thực dồi dào. Nhưng quốc vương đã chuẩn bị cho mọi việc, từ các loại cây trồng do người dân lai tạo, ông đã sử dụng chúng để chế tạo ra các vũ khí phòng thủ ưu việt, nhưng trái bắp được lấy hạt và làm đông cứng lại, cùng với thuốc súng có thể cho ra sự công phá tuyệt vời. Những loại kịch độc từ những cây nấm được điều chế để bắn vào kẻ thù. Chúng ta, những người thủ lĩnh, sẽ sử dụng tài nguyên của vương quốc một cách hợp lý để đẩy lùi kẻ thù, tái lập sự yên bình một lần nữa cho đất nước.

1.3 Các yêu cầu thực hiện đề tài

Xây dựng được các map, các tháp trụ tấn công và hoạt cảnh di chuyển của quái. Tạo âm thanh khi trụ bắn, đặt tháp trụ cho đến khi đợt quái kết thúc thì sẽ chiến thắng.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Transform.translate là gì?

Là hàm được sử dụng để thực hiện việc di chuyển các gameObject một cách tự động trong quá trình chơi game. Ta có thể sử dụng nó trong hàm Update() để có thể cập nhật vị trí của gameObject theo thời gian.

2.2 Collider là gì ?

Collider trong unity thường được sử dụng để có thể nhận ra sự va chạm vật lý giữa các gameObject trong game. Các collider thường có hình dạng như box collider, sphere collider, capsule collider. Để có thể bắt được các sự kiện khi có sự va chạm xảy ra thì chúng ta sẽ sử dụng trigger để có thể tạo ra sự kiện khi phát hiện va chạm. Unity có cung cấp sẵn các API để phát hiện ra các sự va chạm:

- Trigger collider: là thành phần của collider được đánh dấu như một trigger bằng cách bật thuộc tính Is Trigger.
- OnTriggerEnter: là hàm trong Unity được gọi đến khi một gameObject bắt đầu va chạm với một đối tượng được bật Is Trigger từ đó ta có thể viết code vào hàm này để tạo ra sự kiện khi 2 đối tượng va chạm với nhau

2.3 Animator là gì ?

Animator là component trong unity được gắn vào nhân vật để điều khiển animation cho nhân vật. Bằng việc sử dụng animator controller ta có thể code để có thể điều khiển chuyển trạng thái các animation của nhân vật.

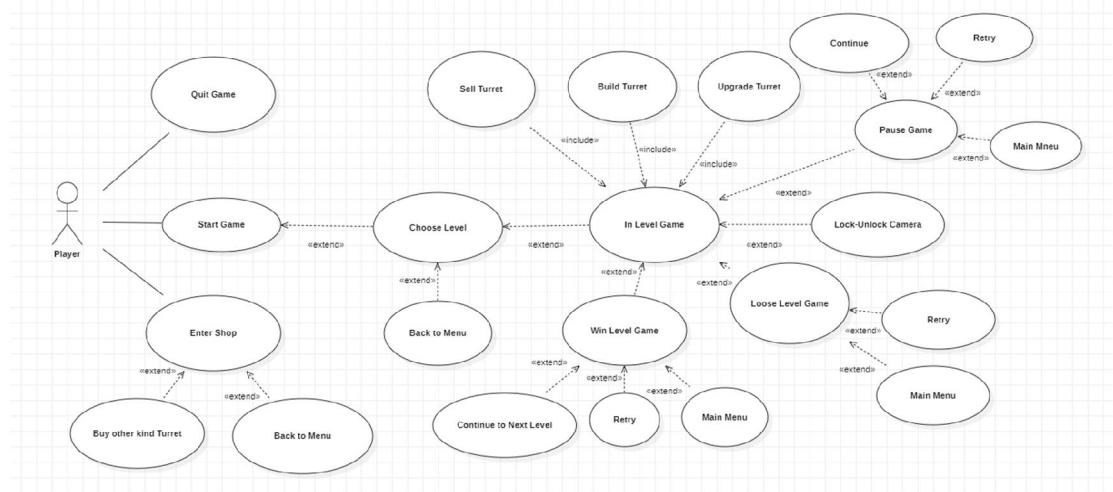
2.4 Audio source là gì?

Là 1 component có sẵn trong unity được dùng để chứa các nguồn âm thanh phát ra trong game-play. Bằng cách tạo ra một gameObject trong game, sau đó gắn component Audio source vào tiếp đến là thêm vào 1 đoạn âm thanh audio clip, audio

clip có thể hỗ trợ nhiều loại định dạng file như MP3, Wav,... là ta có thể nghe được âm thanh trong game

CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ

3.1 Sơ đồ UseCase:



Hình 3.1 Sơ đồ usecase của game

Game thủ thành có các chức năng điển hình cho người chơi như xây dựng các tháp phòng thủ để có thể chống lại các đợt quái vật tấn công vào thành trì. Các trụ có thể được nâng cấp bằng tiền rơi ra khi giết được quái vật, người chơi đồng thời cũng có thể vào cửa hàng để mua các loại trụ mới bằng kim cương. Kim cương có thể đạt được khi người chơi vượt qua được màn chơi đó.

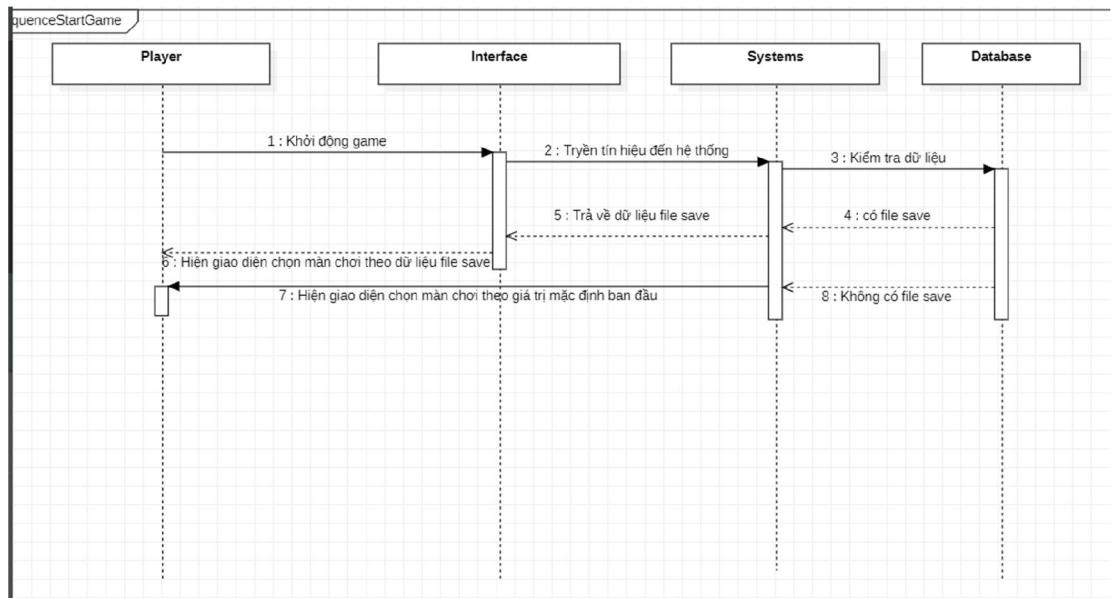
Các usecase trong trò chơi bao gồm:

- Start game: cho phép người chơi khởi động game
- Quit game: cho phép người chơi thoát khỏi trò chơi
- Enter Shop: Người chơi có thể vào cửa hàng để mở khóa súng
- Choose level: cho phép người chơi chọn màn chơi
- Main menu: cho phép người chơi trở ra màn hình chính
- Build turret: cho phép người chơi xây dựng tháp phòng thủ trong game
- Sell turret: cho phép người chơi bán đi các trụ đã xây
- Upgrade turret: cho phép người chơi có thể nâng cấp trụ (tối đa 3 cấp)
- Pause game: cho phép người chơi tạm dừng màn chơi
- Continue game: cho phép người chơi tiếp tục màn chơi

- Retry: cho phép người chơi chơi lại màn chơi
- Next level: cho phép người chơi qua màn tiếp theo khi đã hoàn thành màn trước đó

3.2 Sơ đồ tuần tự (Sequence diagram) của từng chức năng:

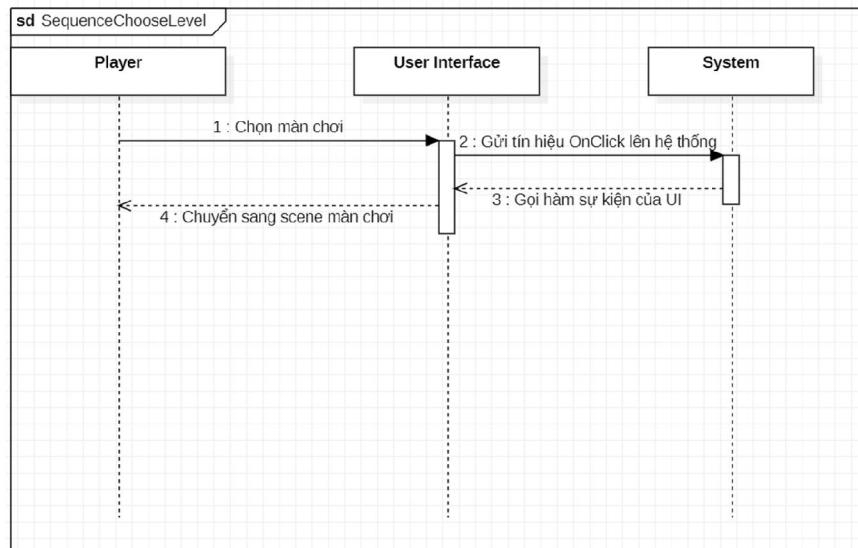
3.2.1 Start game:



Hình 3.2 Sơ đồ tuần tự chức năng Start game

Khi người chơi nhấn vào nút Start game trên menu chính để bắt đầu trò chơi thì hệ thống sẽ kiểm tra dữ liệu file save lưu trữ tiến trình của chơi của người chơi trên cơ sở dữ liệu, sau đó sẽ chuyển sang màn hình chọn màn chơi. Màn hình chọn màn chơi sẽ hiện ra các màn chơi đã mở khóa và chưa mở khóa dựa trên dữ liệu tiến trình của người chơi.

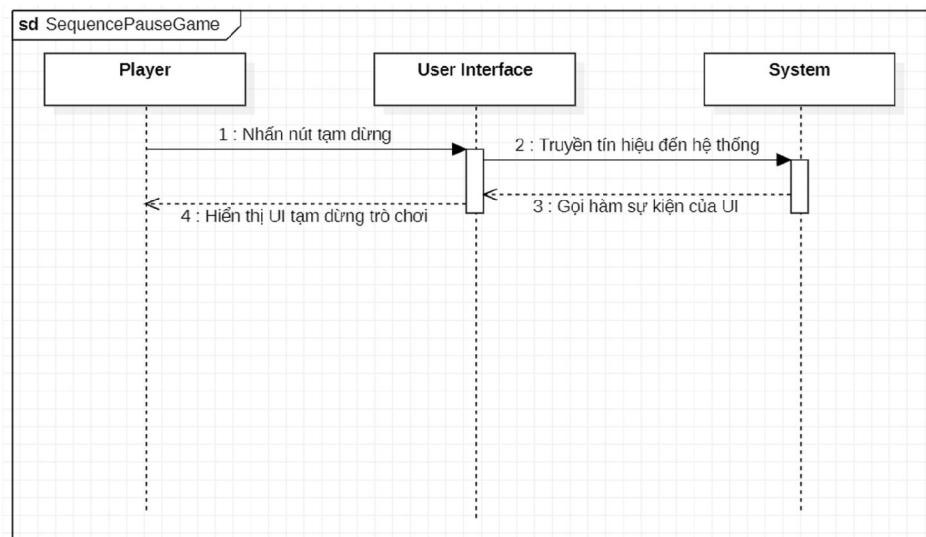
3.2.2 Choose level



Hình 3.3 Sơ đồ tuần tự chức năng choose level

Khi người chơi nhấn chọn màn chơi trên màn hình chọn màn chơi, thì hệ thống sẽ chuyển màn hình (scene) sang màn hình màn chơi và bắt đầu tiến hành chơi.

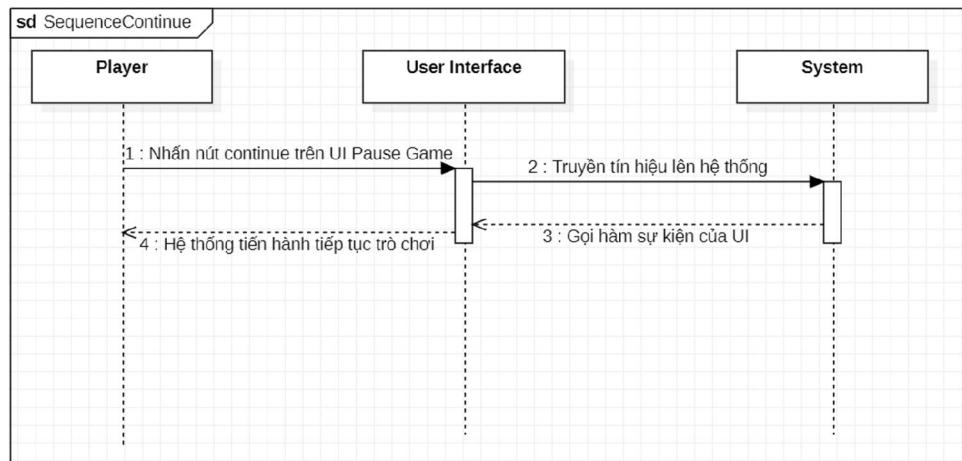
3.2.3 Pause game:



Hình 3.4 Sơ đồ tuần tự chức năng pause game

Khi người chơi nhấn phím P trên bàn phím thì hệ thống sẽ cho trò chơi tạm thời ngưng lại và sau đó hiển thị UI dừng trò chơi.

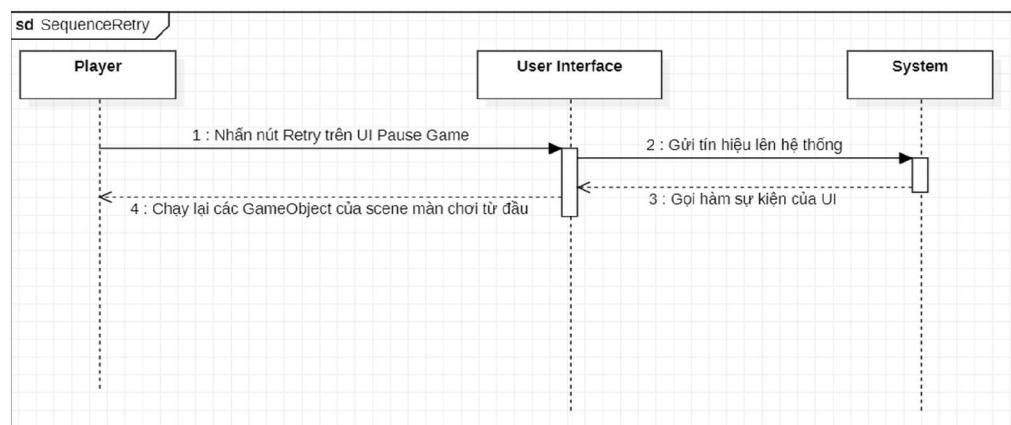
3.2.4 Continue:



Hình 3.5 Sơ đồ tuần tự chức năng continue game

Khi người chơi nhấn nút continue trên UI pause game thì hệ thống sẽ tiến hành tắt đi không hiển thị UI pause game và cho trò chơi tiếp tục chạy từ lúc đã dừng.

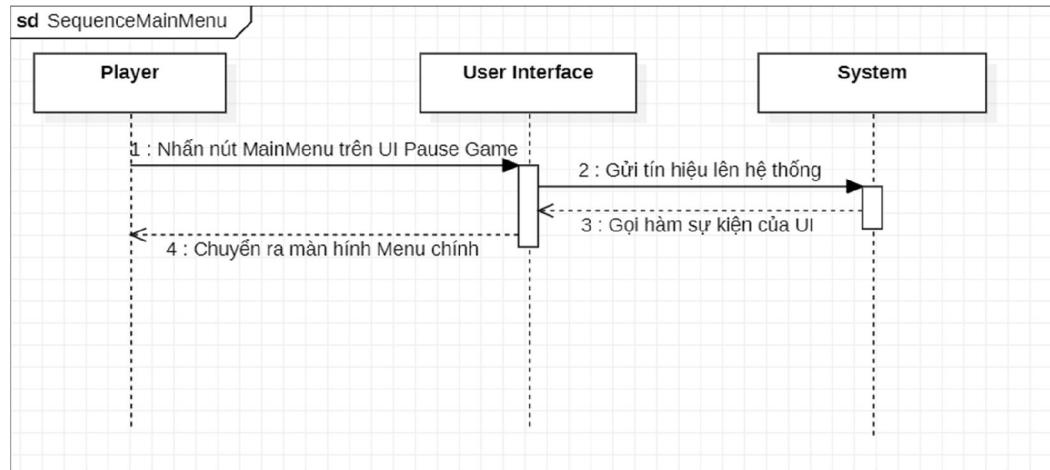
3.2.5 Retry



Hình 3.6 Sơ đồ tuần tự chức năng retry

Khi người chơi nhấn vào nút retry trên UI Pause game thì hệ thống sẽ tiến hành tải lại scene mà người chơi đang chơi.

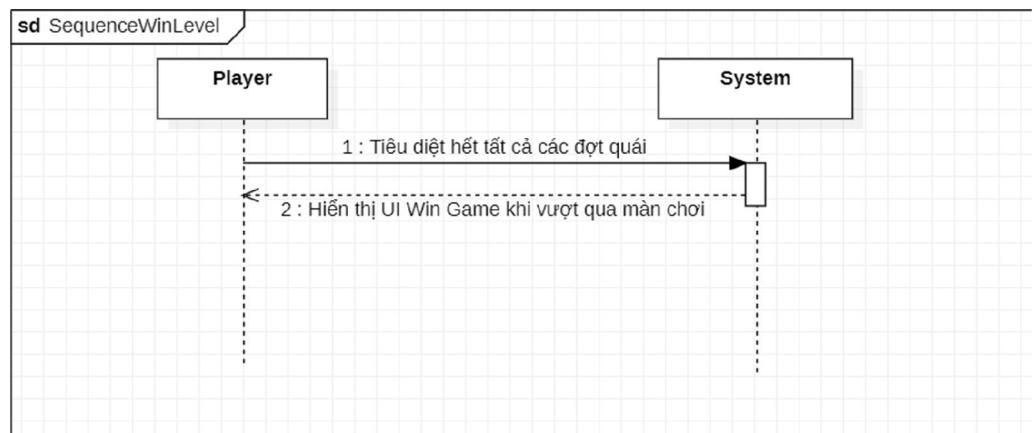
3.2.6 Main menu:



Hình 3.7 Sơ đồ tuần tự chức năng main menu

Khi người chơi nhấn vào nút main menu trên UI pause game thì hệ thống sẽ chuyển người chơi về lại màn hình chính.

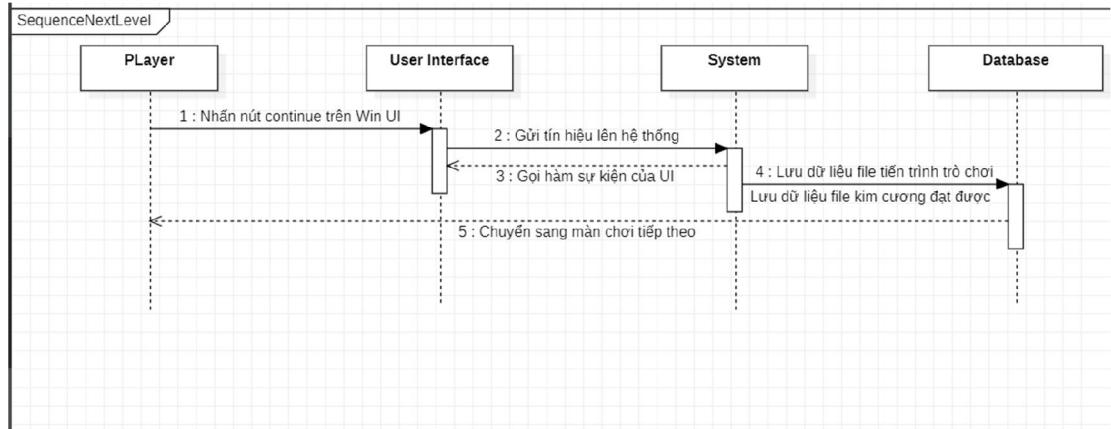
3.2.7 Win level:



Hình 3.8 Chức năng win level

Khi người chơi tiêu diệt được hết một số lượt quái nhất định của màn chơi đó thì hệ thống sẽ hiện ra win UI cho người chơi

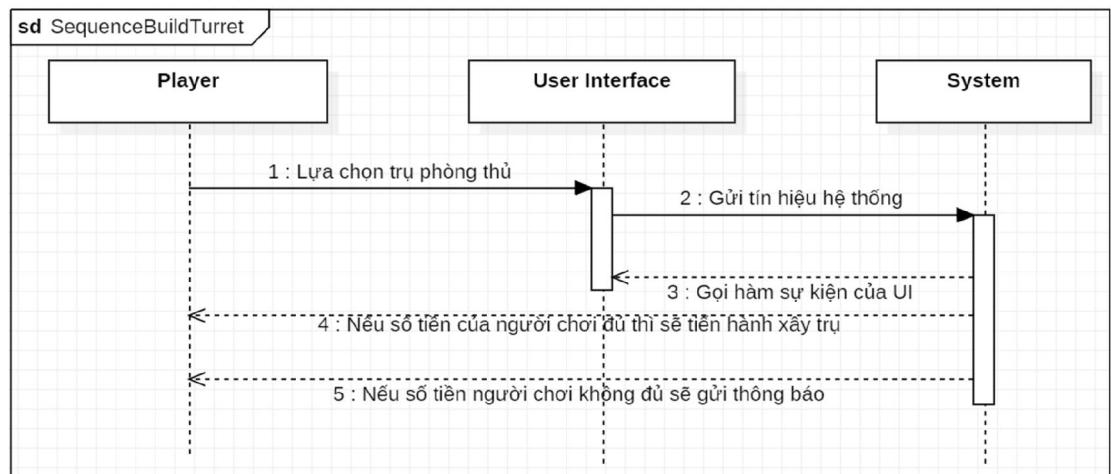
3.2.8 Next level:



Hình 3.9 Sơ đồ tuần tự chức năng next level

Khi người chơi nhấn vào nút continue trên win UI thì hệ thống sẽ tiến hành lưu trữ dữ liệu tiến trình trò chơi và số lượng kim cương đạt được của người chơi vào file và sau đó chuyển màn hình qua cảnh màn chơi tiếp theo.

3.2.9 Xây dựng trụ phòng thủ:

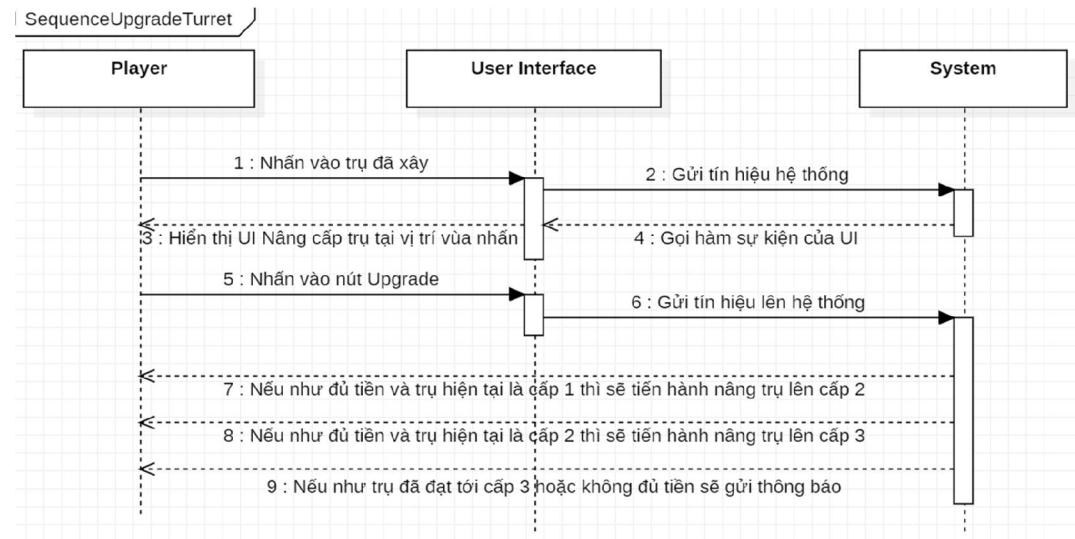


Hình 3.10 Sơ đồ tuần tự chức năng xây dựng trụ phòng thủ

Khi người chơi lựa chọn trụ phòng thủ và đặt vào ô để tiến hành xây trụ thì hệ thống sẽ kiểm tra trụ mà người chơi lựa chọn sau đó sẽ tiến hành kiểm tra số tiền còn lại của người chơi. Nếu như số tiền của người chơi nhỏ hơn giá tiền của trụ thì sẽ gửi

thông báo không đủ tiền cho người chơi, còn nếu như người chơi đủ tiền thì sẽ tiến hành xây dựng trụ cho người chơi và sẽ trừ đi số tiền của người chơi.

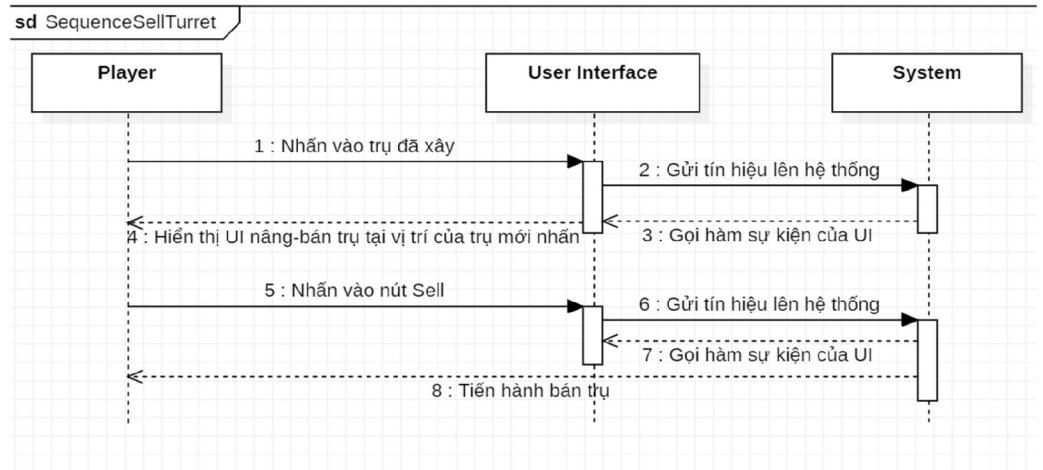
3.2.10 Nâng cấp trụ phòng thủ:



Hình 3.11 Sơ đồ tuần tự chức năng nâng cấp trụ

Khi người chơi nhấn vào trụ đã xây dựng thì hệ thống sẽ hiện ra UI nâng cấp/bán trụ cho người chơi. Khi người chơi nhấn vào nút nâng cấp thì hệ thống sẽ tiến hành kiểm tra tiền của người chơi và cấp độ trụ hiện tại của trụ mà người chơi đã bấm chọn. Nếu người chơi đủ tiền và cấp độ của trụ hiện đang ở cấp 1 thì sẽ trừ tiền và nâng trụ lên cấp 2 và tương tự khi trụ hiện ở cấp độ 2 thì sẽ nâng lên cấp độ 3. Đến khi trụ của người chơi đạt cấp tối đa (cấp 3) thì sẽ gửi thông báo trụ đã đạt cấp tối đa cho người chơi.

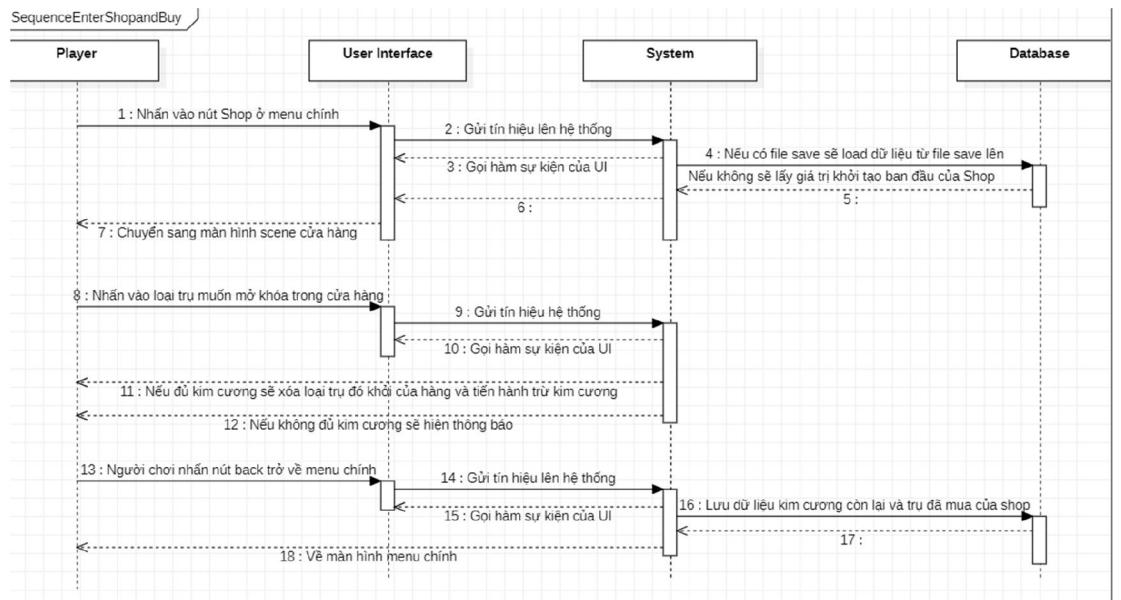
3.2.11 Bán trụ:



Hình 3.12 Sơ đồ tuần tự chức năng bán trụ

Khi người chơi nhấn vào trụ đã xây thì hệ thống sẽ hiện ra UI nâng cấp/bán trụ, khi người chơi nhấn vào nút bán trụ thì hệ thống sẽ tiến hành xóa trụ đó và hoàn tiền cho người chơi.

3.2.12 Cửa hàng:



Hình 3.13 Sơ đồ tuần tự chức năng cửa hàng

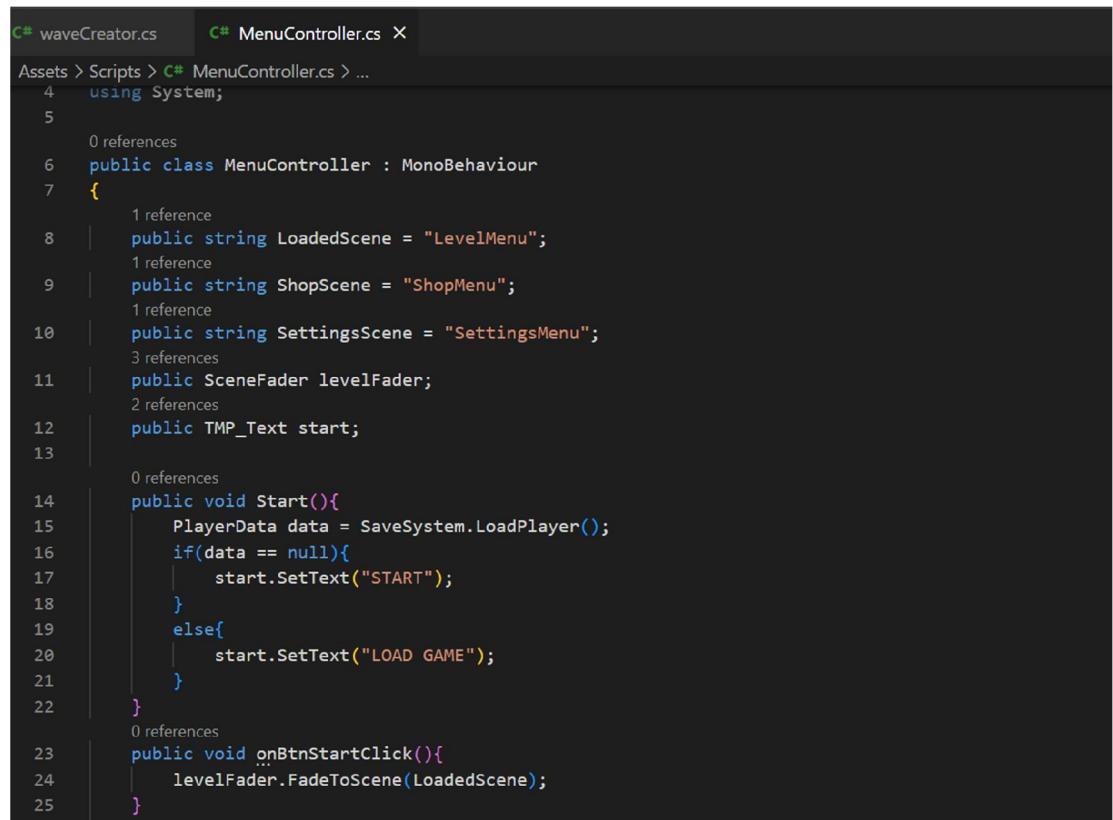
Khi người chơi nhấn vào cửa hàng thì hệ thống sẽ kiểm tra dữ liệu file save của shop sau đó sẽ chuyển sang màn hình cửa hàng. Màn hình cửa hàng sẽ load dữ liệu từ file save lên để hiển thị các tru mà người chơi chưa mua và số kim cương hiện có của người chơi (kim cương là đơn vị tiền tệ dùng để mở khóa các loại tru khác nhau cho người chơi, kim cương có thể đạt được khi người chơi vượt qua được 1 màn chơi).

Khi người chơi nhấn chọn vào loại tru muốn mở khóa thì hệ thống sẽ kiểm tra số kim cương còn lại của người chơi có đủ hay không, nếu đủ thì sẽ tiến hành mở khóa tru để người chơi có thể sử dụng tru trong các màn chơi. Sau khi mở khóa thành công hệ thống sẽ tiến hành trừ đi số kim cương hiện có của người chơi. Sau khi hoàn tất mua bán và nhấn vào nút back để trở về màn hình chính thì hệ thống sẽ tiến hành lưu lại các tru đã mua cũng như chưa mua của người chơi và số kim cương còn lại của người chơi vào file save.

CHƯƠNG 4. TRIỂN KHAI VÀ THỰC NGHIỆM HỆ THỐNG

✧ **Triển khai hệ thống bao gồm các chức năng sau**

4.1 Thiết kế chức năng start game:



```

C# waveCreator.cs      C# MenuController.cs X
Assets > Scripts > C# MenuController.cs > ...
4  using System;
5
6  public class MenuController : MonoBehaviour
7  {
8      public string LoadedScene = "LevelMenu";
9      public string ShopScene = "ShopMenu";
10     public string SettingsScene = "SettingsMenu";
11     public SceneFader levelFader;
12     public TMP_Text start;
13
14     public void Start()
15     {
16         PlayerData data = SaveSystem.LoadPlayer();
17         if(data == null){
18             start.SetText("START");
19         }
20         else{
21             start.SetText("LOAD GAME");
22         }
23     }
24     public void onBtnStartClick()
25     {
26         levelFader.FadeToScene(LoadedScene);
27     }
}

```

Hình 4.1 Code chức năng start game

Khi người chơi nhấn vào nút Start game của màn hình menu chính để bắt đầu chơi game thì hệ thống sẽ gọi đến hàm onBtnStartClick() của class MenuController để chuyển sang scene LevelMenu để tiến hành lựa chọn màn chơi. Hệ thống sẽ kiểm tra nếu thấy có file save để lưu trữ dữ liệu tiến trình của người chơi tức là người chơi này đã chơi rồi và muốn tiếp tục thì nút START sẽ được chuyển thành LOAD GAME

4.2 Thiết kế chức năng choose level:

```
C# LevelController.cs X
Assets > Scripts > C# LevelController.cs > ...
1 reference
7     public string menuScene = "MainMenu";
2 references
8     public SceneFader sceneFader;
3 references
9     public Button[] Levels;
3 references
10    int highestLevel;
0 references
11    public void Start(){
12        PlayerData data = SaveSystem.LoadPlayer();
13        if(data == null){
14            highestLevel = 1;
15        }else {
16            highestLevel = data.HighestLevel;
17        }
18        for(int i = 0; i < Levels.Length; i++){
19            if(i + 1 > highestLevel){
20                Levels[i].interactable = false;
21            }
22            Debug.Log(Levels[i]);
23        }
24
25
26
27    }
0 references
28    public void onLevelClicked(string levelChose){
29        sceneFader.FadeToScene(levelChose);
30    }
}
```

Hình 4.2 Code chức năng chọn level

Khi chuyển sang màn hình lựa chọn màn chơi thì hệ thống sẽ kiểm tra xem có file dữ liệu tiến trình của người chơi không. Nếu có, thì hệ thống sẽ mở khóa những màn chơi mà người chơi đã vượt qua và khóa lại các màn chơi mà người chơi chưa thể đạt đến dựa trên file dữ liệu tiến trình của người chơi. Nếu không có file dữ liệu thì hệ thống sẽ mặc định mở khóa màn chơi đầu tiên cho người chơi.

Sau khi người chơi nhấn chọn màn chơi trên màn hình thì hệ thống sẽ gọi hàm `onLevelClicked()` và chuyển đến scene với màn chơi tương ứng.

4.3 Thiết kế Game-play:

4.3.1 Quái vật:

4.3.1.1 Thiết kế di chuyển của quái:

```

C# Enemy.cs X
Assets > Scripts > C# Enemy.cs > ...
22     public GameObject dieEffect;
23     1 reference
24     public GameObject dieAudio;
25     2 references
26     private bool isDead = false;
27     2 references
28     public Image HPBar;
29
30     0 references
31     private void Start() {
32         target = Waypoints.waypoints[0];
33         oldHealth = health;
34         oldSpeed = speed;
35         animator = GetComponent<Animator>();
36     }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

```

Hình 4.3 Code đặt mục tiêu điểm đến cho quái

Để quái có thể tự di chuyển được thì nhóm chúng em sẽ thiết kế các điểm đến cho quái và bỏ các điểm đến này vào 1 mảng. Ta sẽ đặt target cho quái là điểm đến đầu tiên trong mảng.

```

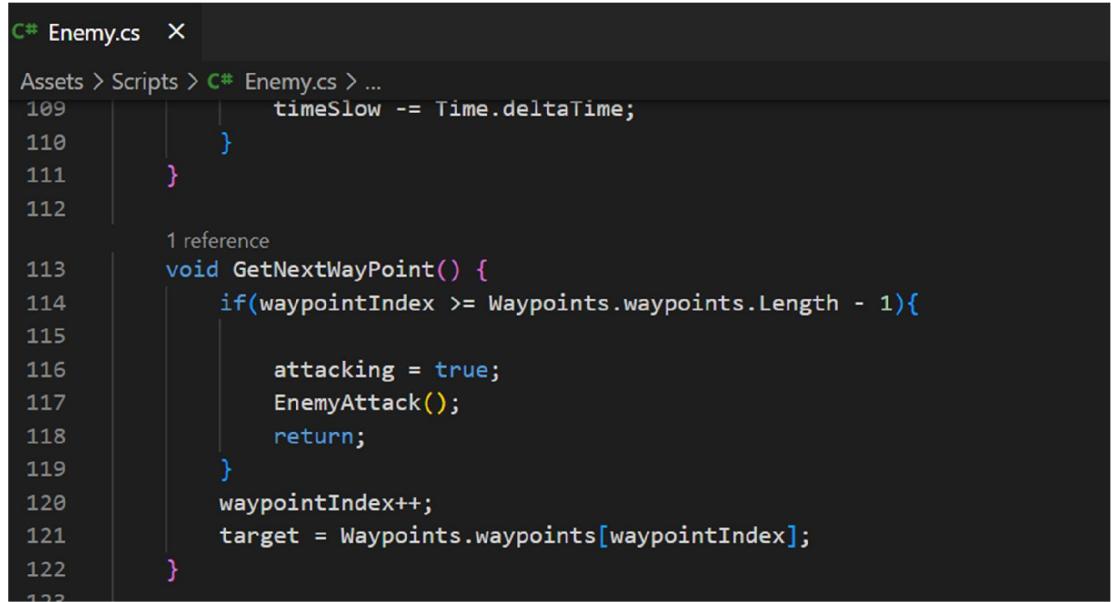
C# Enemy.cs X
Assets > Scripts > C# Enemy.cs > ...
84         speed = speedSlow;
85     }
86     1 reference
87     public void TakeBackSpeed(){
88         speed = oldSpeed;
89     }
89     0 references
90     private void Update(){
91         if(health>0 && attacking == false){
92             Vector3 direction = target.position - transform.position;
93             transform.Translate(direction.normalized * speed * Time.deltaTime, Space.World);
94             transform.LookAt(target);
95         }
96
97         //Kiểm tra nhân vật đã đến waypoint chưa
98         if(Vector3.Distance(transform.position, target.position) <= 0.4f)
99         {
100             GetNextWayPoint();
101         }
102

```

Hình 4.4 Code quái tự động di chuyển đến mục tiêu

Sau đó, em sẽ cho quái tiến hành tự di chuyển đến mục tiêu bằng việc sử dụng hàm transform.Translate() và cho quái nhìn hướng vào mục tiêu.

Sau khi quái đã đến được mục tiêu đầu tiên em sẽ tiến hành gọi hàm GetNextWayPoint()



```
C# Enemy.cs
Assets > Scripts > C# Enemy.cs > ...
109     timeSlow -= Time.deltaTime;
110 }
111 }
112
113 void GetNextWayPoint() {
114     if(waypointIndex >= Waypoints.waypoints.Length - 1){
115
116         attacking = true;
117         EnemyAttack();
118         return;
119     }
120     waypointIndex++;
121     target = Waypoints.waypoints[waypointIndex];
122 }
```

Hình 4.5 Code chuyển mục tiêu của quái

Hàm GetNextWayPoint() sẽ thay đổi giá trị mục tiêu target thành điểm đến tiếp theo trong mảng chứa waypoints[] chứa các điểm đến. Vì vị trí target được thay đổi nên quái sẽ tự động di chuyển đến điểm đến tiếp theo. Cứ như vậy tiếp tục cho đến khi quái đến được điểm đến cuối cùng đồng nghĩa với việc đến được thành trì của ta thì ta sẽ cho quái thực hiện việc tấn công bằng việc gọi hàm EnemyAttack()

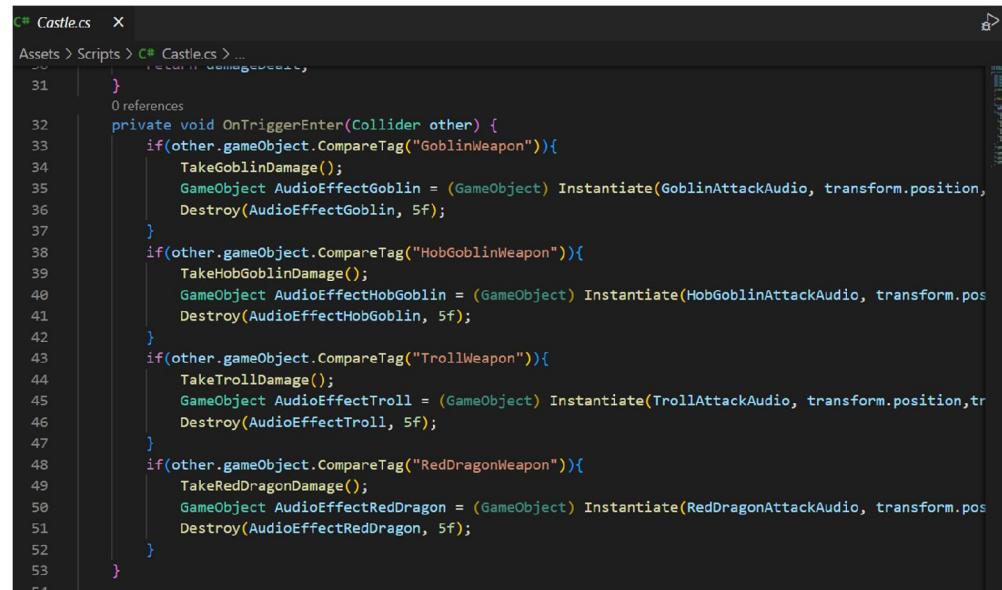


```
j
1 reference
void EnemyAttack() {
    animator.SetBool("IsAttacking", attacking);
}
```

Hình 4.6 Code quái tấn công thành

Hàm này sẽ set animator cho quái để thực hiện việc tấn công thành trì, đánh dấu Is Trigger lên collider vũ khí của các quái vật và khi tác động tới thành trì được

gắn collider sẽ thực hiện việc trừ máu lên người chơi. Với những vũ khí khác nhau của quái ta sẽ gắn tag cho từng loại vũ khí theo từng loại quái để thực hiện việc trừ máu của người chơi theo những loại quái khác nhau khi tấn công thành trì.



```

C# Castle.cs
Assets > Scripts > C# Castle.cs > ...
31 }
32     0 references
33     private void OnTriggerEnter(Collider other) {
34         if(other.gameObject.CompareTag("GoblinWeapon")){
35             TakeGoblinDamage();
36             GameObject AudioEffectGoblin = (GameObject) Instantiate(GoblinAttackAudio, transform.position,
37             Destroy(AudioEffectGoblin, 5f);
38         }
39         if(other.gameObject.CompareTag("HobGoblinWeapon")){
40             TakeHobGoblinDamage();
41             GameObject AudioEffectHobGoblin = (GameObject) Instantiate(HobGoblinAttackAudio, transform.pos
42             Destroy(AudioEffectHobGoblin, 5f);
43         }
44         if(other.gameObject.CompareTag("TrollWeapon")){
45             TakeTrollDamage();
46             GameObject AudioEffectTroll = (GameObject) Instantiate(TrollAttackAudio, transform.position,tr
47             Destroy(AudioEffectTroll, 5f);
48         }
49         if(other.gameObject.CompareTag("RedDragonWeapon")){
50             TakeRedDragonDamage();
51             GameObject AudioEffectRedDragon = (GameObject) Instantiate(RedDragonAttackAudio, transform.pos
52             Destroy(AudioEffectRedDragon, 5f);
53     }

```

Hình 4.7 Code trừ máu của người chơi theo từng loại quái đánh vào thành

4.3.1.2 Thiết kế các đợt quái:

```

C# waveCreator.cs × C# WaveManagement.cs
Assets > Scripts > C# waveCreator.cs > waveCreator > Update
29
30    }
31    0 references
32    private void Update() {
33        Debug.Log("Current wave: " + waveIndex);
34
35
36        if(waveIndex == waves.Length && enemiesStillLive <= 0){
37            gameController.LevelWin();
38            ShopMenu.diamonds += 50;
39            this.enabled = false;
40        }
41        else {
42            if(enemiesStillLive==0){
43                countdown-= Time.deltaTime;
44                countdown = Mathf.Clamp(countdown, 0f, Mathf.Infinity);
45                waveCountDownText.text = string.Format("{0:0.0}", countdown);
46                waveCountDownText.SetText("Next Wave: " + countdown);
47            }
48            if(countdown <= 0 && enemiesStillLive <= 0){
49                StartCoroutine(CreateWave());
50                countdown = timeBetweenWaves;
51            }
52        }
53
54    }
55}

```

Hình 4.8 Code tạo khác đợt quái

```

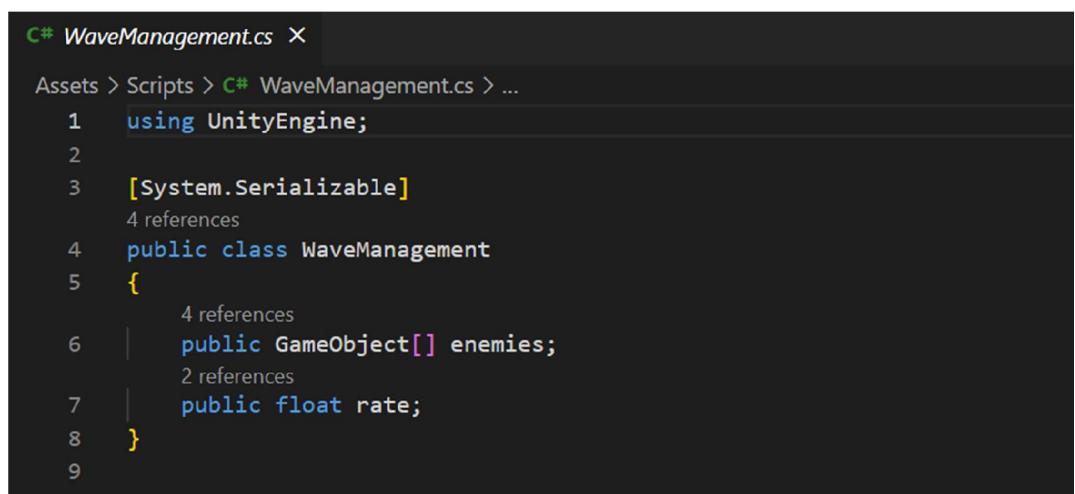
C# waveCreator.cs C# GameController.cs ×
Assets > Scripts > C# GameController.cs > ...
33        Debug.Log("Game Over");
34    }
35
36    1 reference
37    public void LevelWin(){
38        gameFinished = true;
39        levelComplete.SetActive(true);
40        Debug.Log("LEVEL WON");
41    }
42
43
44}
45

```

Hình 4.9 Code xuất hiện UI khi chiến thắng màn chơi

Khi vào trò chơi người chơi sẽ được chờ một khoảng thời gian trước khi các đợt quái bắt đầu xuất hiện, người chơi sẽ được xem là hoàn thành màn chơi sau khi đi đến đợt quái cuối cùng của màn chơi và sau khi tiêu diệt hết tất cả những con quái vật xuất hiện đến khi chỉ số enemyStillLive ≤ 0 thì người chơi xem như đã hoàn thành trò chơi. Sau khi vượt qua đợt quái cuối cùng thì người chơi sẽ được cộng 50 kim cương dùng để mở khóa súng trong cửa hàng và hàm gameController.LevelWin() sẽ làm xuất hiện UI win lên màn hình của người chơi.

Người chơi phải diệt hết tất cả những quái vật còn sống của đợt đó thì đợt quái vật tiếp theo mới được tạo ra. Các đợt quái vật được gọi ra bởi hàm CreateWave()



```
C# WaveManagement.cs ×
Assets > Scripts > C# WaveManagement.cs > ...
1  using UnityEngine;
2
3  [System.Serializable]
4  public class WaveManagement
5  {
6      public GameObject[] enemies;
7      public float rate;
8  }
9
```

Hình 4.10 Class đối tượng các đợt quái

Class waveManagement là đối tượng đợt quái chứa các thuộc tính gồm mảng enemies[] chứa các prefab các đợt quái mà em sẽ muốn xuất hiện vào mỗi đợt và biến rate để cho việc tạo quái diễn ra một cách từ từ khi mà prefab quái này đang được tạo ra thì em sẽ cho code chờ một khoảng thời gian rồi sau đó mới tiếp tục tạo ra prefab quái khác vào game điều này sẽ giúp cho các con quái không bị dính vào nhau khi ma thời gian chạy của code nhanh hơn so với thời gian tạo 1 quái.

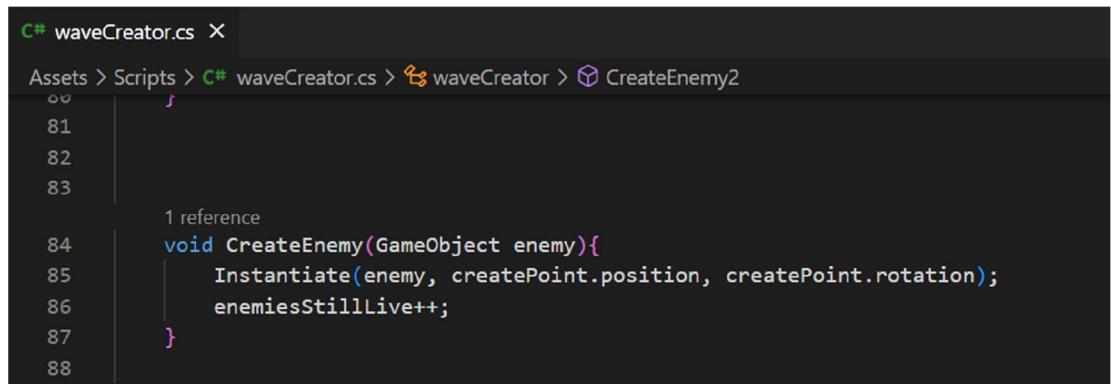
```

C# waveCreator.cs X C# GameController.cs
Assets > Scripts > C# waveCreator.cs > waveCreator
55     }
56
57     1 reference
58     IEnumerator CreateWave(){
59
60         WaveManagement wave = waves[waveIndex];
61         for (int i = 0; i < wave.enemies.Length; i++)
62         {
63             CreateEnemy(wave.enemies[i]);
64             yield return new WaitForSeconds(wave.rate);
65         }
66
67         WaveManagement wave2 = waves2[waveIndex];
68         for (int i = 0; i < wave2.enemies.Length; i++)
69         {
70             CreateEnemy2(wave2.enemies[i]);
71             yield return new WaitForSeconds(wave2.rate);
72         }
73         waveIndex++;
74
75         int waveIndex1 = waveIndex;
76         PlayerController.currentLevel = "";
77         PlayerController.currentLevel += "Level " + level + "-" + waveIndex1;
78         Debug.Log(PlayerController.currentLevel);
79
80     }
81

```

Hình 4.11 Code hàm tạo ra các đợt quái

Tiếp theo, ta sẽ tạo ra một mảng chứa các đối tượng đợt quái và cho vòng lặp để duyệt mảng đó. Sau đó, khi vào từng đối tượng đợt quái ta sẽ lấy ra mảng enemies[] chứa prefab của từng con quái sau đó gọi hàm CreateEnemy() và truyền vào prefab của từng con quái. Sau mỗi lần gọi hàm CreateEnemy() để tạo quái thì ta sẽ gọi hàm WaitForSeconds() với giá trị rate của đối tượng đợt quái được truyền vào để cho code dừng lại một khoảng thời gian để cho những prefab quái không bị dính vào nhau

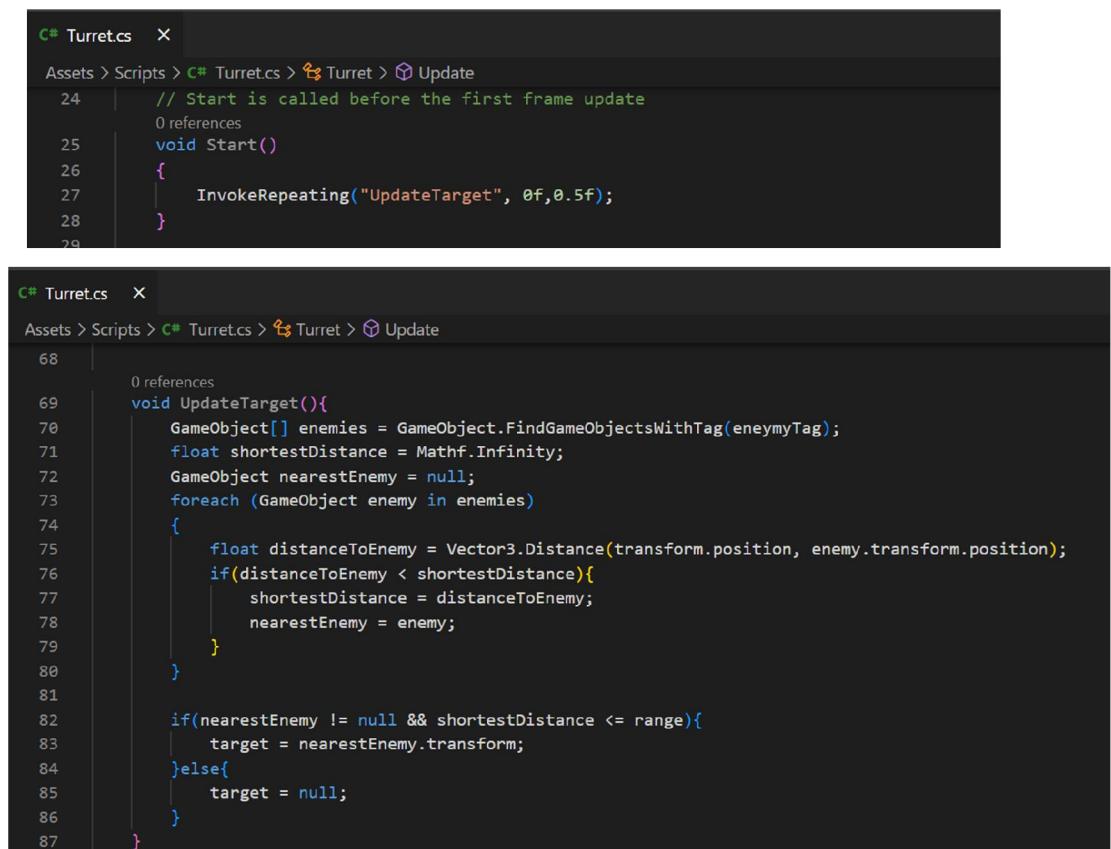


```
C# waveCreator.cs X
Assets > Scripts > C# waveCreator.cs > waveCreator > CreateEnemy2
80
81
82
83
84     void CreateEnemy(GameObject enemy){
85         Instantiate(enemy, createPoint.position, createPoint.rotation);
86         enemiesStillLive++;
87     }
88
```

Hình 4.12 Hàm tạo ra từng con quái vào game

Hàm CreateEnemy() sẽ tiến hành tạo quái vào game và sau mỗi khi tạo được 1 con quái thì giá trị enemiesStillLive sẽ được tăng 1.

4.3.2 Các trụ phòng thủ:



```
C# Turret.cs X
Assets > Scripts > C# Turret.cs > Turret > Update
24     // Start is called before the first frame update
25     0 references
26     void Start()
27     {
28         InvokeRepeating("UpdateTarget", 0f, 0.5f);
29     }
```

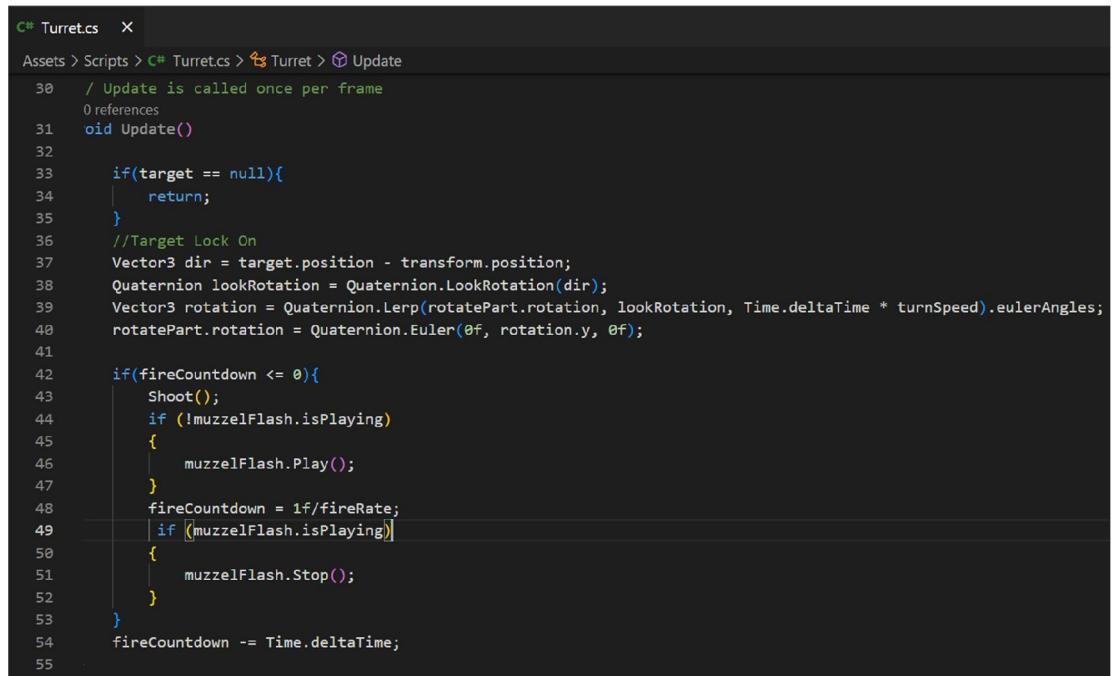


```
C# Turret.cs X
Assets > Scripts > C# Turret.cs > Turret > Update
68
69     0 references
70     void UpdateTarget(){
71         GameObject[] enemies = GameObject.FindGameObjectsWithTag(enemyTag);
72         float shortestDistance = Mathf.Infinity;
73         GameObject nearestEnemy = null;
74         foreach (GameObject enemy in enemies)
75         {
76             float distanceToEnemy = Vector3.Distance(transform.position, enemy.transform.position);
77             if(distanceToEnemy < shortestDistance){
78                 shortestDistance = distanceToEnemy;
79                 nearestEnemy = enemy;
80             }
81         }
82         if(nearestEnemy != null && shortestDistance <= range){
83             target = nearestEnemy.transform;
84         }else{
85             target = null;
86         }
87     }
```

Hình 4.13 Code cập nhật vị trí mục tiêu

Các trụ phòng thủ em sẽ thiết kế các trụ sẽ tự động quay nòng súng theo vị trí của mục tiêu. Bằng việc liên tục gọi hàm UpdateTarget() thì ta sẽ thay đổi mục tiêu cần được nhắm trúng.

Trong hàm UpdateTarget() ta sẽ tạo ra các mảng chứa các gameObject có gắn tag “enemy” trong game. Sau đó, ta sẽ cho tiến hành tính toán khoảng cách giữa các enemy đó tới trụ phòng thủ, và tìm ra khoảng cách nào ngắn nhất và nếu khoảng cách ấy nhỏ hơn tầm bắn của trụ thì sẽ xác định enemy đó là mục tiêu cần tiêu diệt. Việc liên tục gọi hàm UpdateTarget() sẽ giúp cho các trụ phòng thủ kịp thời thay đổi mục tiêu khi mà có những quái vật có tốc độ nhanh hơn mục tiêu trước đó thì các trụ phòng thủ sẽ thay đổi mục tiêu mà hướng nòng súng về những mục tiêu đó.



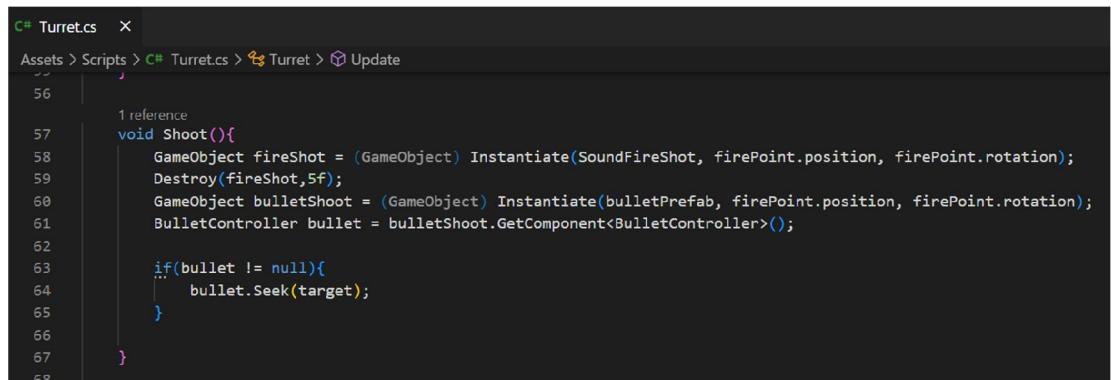
```

C# Turret.cs X
Assets > Scripts > C# Turret.cs > Turret > Update
30  // Update is called once per frame
31  void Update()
32
33  if(target == null){
34      return;
35  }
36  //Target Lock On
37  Vector3 dir = target.position - transform.position;
38  Quaternion lookRotation = Quaternion.LookRotation(dir);
39  Vector3 rotation = Quaternion.Lerp(rotatePart.rotation, lookRotation, Time.deltaTime * turnSpeed).eulerAngles;
40  rotatePart.rotation = Quaternion.Euler(0f, rotation.y, 0f);
41
42  if(fireCountdown <= 0){
43      Shoot();
44      if (!muzzleFlash.isPlaying)
45      {
46          muzzleFlash.Play();
47      }
48      fireCountdown = 1f/fireRate;
49      if (!muzzleFlash.isPlaying)
50      {
51          muzzleFlash.Stop();
52      }
53  }
54  fireCountdown -= Time.deltaTime;
55

```

Hình 4.14 Code nòng súng theo vị trí của mục tiêu

Khi giá trị target khác null thì ta sẽ cho trụ phòng thủ xoay súng theo vị trí của mục tiêu và khi giá trị fireCountdown giảm về 0 ta sẽ cho súng bắn ra đạn bằng việc gọi hàm Shoot() và sau khi bắn xong 1 viên đạn thì ta sẽ cho fireCountdown trở về giá trị > 0 và tiến hành giảm giá trị này theo thời gian. Và trong thời gian chờ thì các trụ phòng thủ sẽ chỉ có thể xoay nòng súng theo vị trí mục tiêu mà không thể bắn.



```

C# Turret.cs X
Assets > Scripts > C# Turret.cs > Turret > Update
56
57     void Shoot(){
58         GameObject fireShot = (GameObject) Instantiate(SoundFireShot, firePoint.position, firePoint.rotation);
59         Destroy(fireShot, 5f);
60         GameObject bulletShoot = (GameObject) Instantiate(bulletPrefab, firePoint.position, firePoint.rotation);
61         BulletController bullet = bulletShoot.GetComponent<BulletController>();
62
63         if(bullet != null){
64             bullet.Seek(target);
65         }
66     }
67 }
68

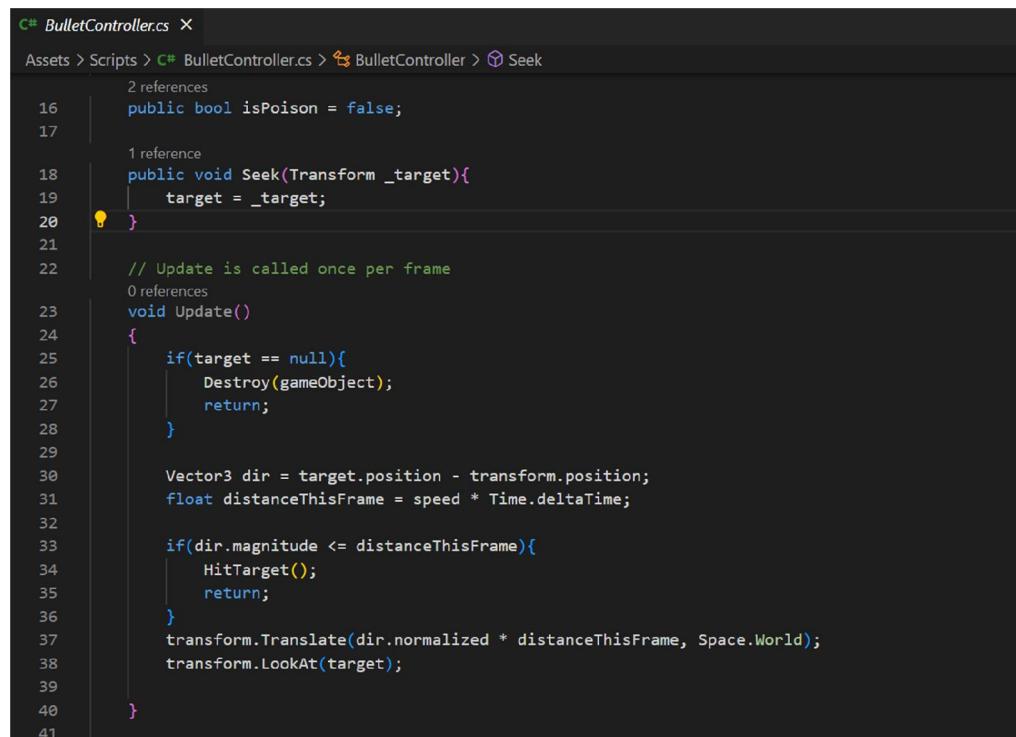
```

Hình 4.15 Hàm Shoot() cho phép trụ bắn đạn

Hàm Shoot() sẽ tạo ra gameObject bullet vào trong game và đưa giá trị mục tiêu (target) vào script của bullet.

4.3.3 Các loại đạn:

Trong game này, ta sẽ thiết kế các loại đạn khác nhau tương ứng cho từng loại trụ khác nhau. Các loại đạn bao gồm đạn bắn 1 mục tiêu, đạn gây sát thương AOE, đạn làm chậm và đạn gây sát thương độc theo thời gian.



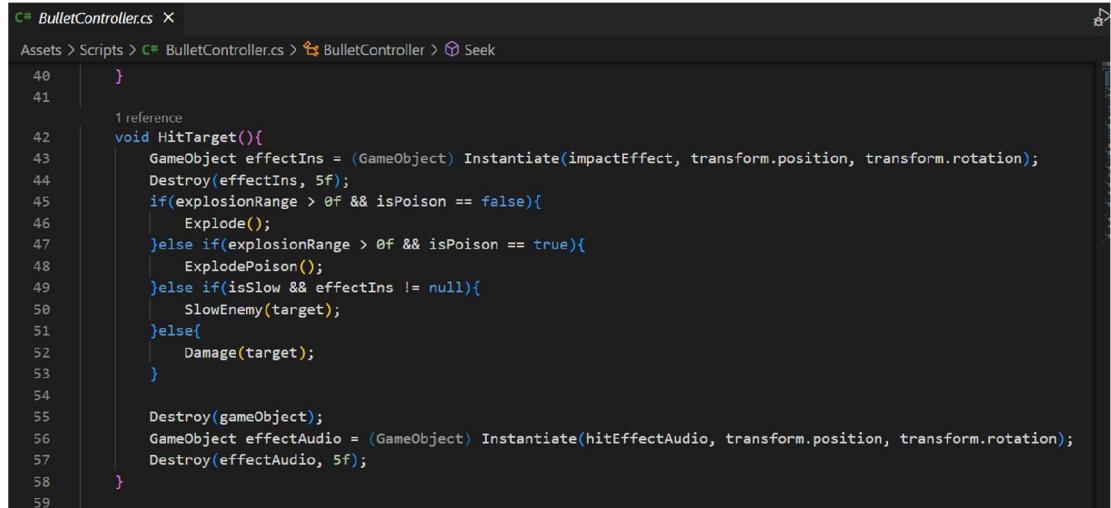
```

C# BulletController.cs X
Assets > Scripts > C# BulletController.cs > BulletController > Seek
2 references
16     public bool isPoison = false;
17
18     1 reference
19     public void Seek(Transform _target){
20         target = _target;
21     }
22
23     // Update is called once per frame
24     0 references
25     void Update()
26     {
27         if(target == null){
28             Destroy(gameObject);
29             return;
30         }
31
32         Vector3 dir = target.position - transform.position;
33         float distanceThisFrame = speed * Time.deltaTime;
34
35         if(dir.magnitude <= distanceThisFrame){
36             HitTarget();
37             return;
38         }
39         transform.Translate(dir.normalized * distanceThisFrame, Space.World);
40         transform.LookAt(target);
41     }

```

Hình 4.16 Đạn di chuyển đến mục tiêu và gây sát thương

Khi BulletController nhận được giá trị target từ Turret truyền qua thì sẽ tiến hành tính toán khoảng cách từ đạn đến kẻ thù mục tiêu sau đó sử dụng hàm Transform.translate() để cho viên đạn tự động di chuyển đến vị trí của mục tiêu. Khi viên đạn đến gần mục tiêu thì sẽ gọi hàm EncountEnemy()



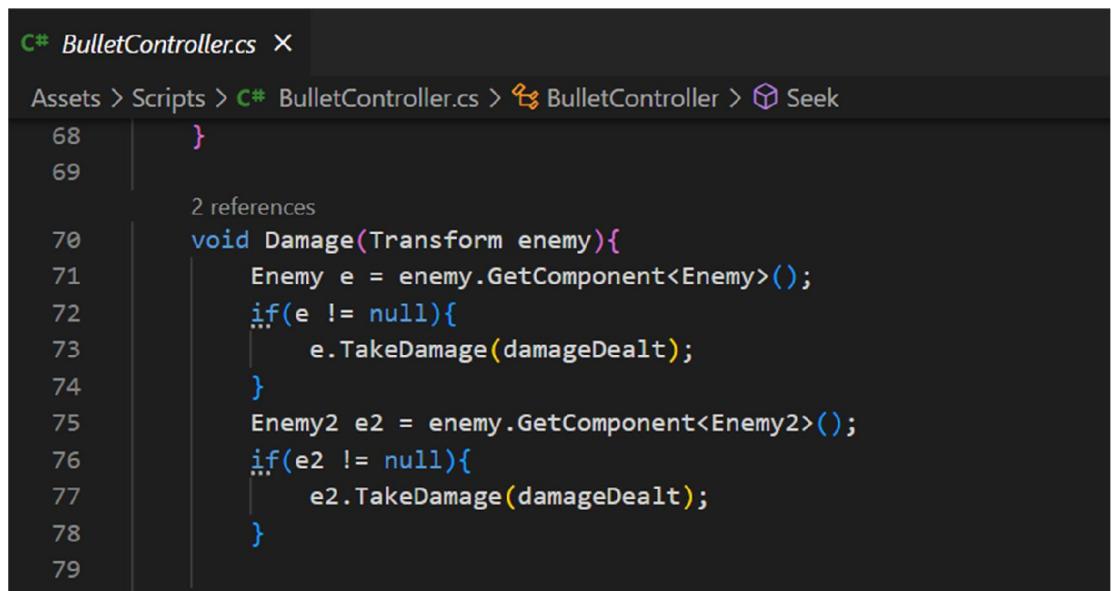
```

C# BulletController.cs ×
Assets > Scripts > C# BulletController.cs > BulletController > Seek
40    }
41
42    void HitTarget(){
43        GameObject effectIns = (GameObject) Instantiate(impactEffect, transform.position, transform.rotation);
44        Destroy(effectIns, 5f);
45        if(explosionRange > 0f && isPoison == false){
46            Explode();
47        }else if(explosionRange > 0f && isPoison == true){
48            ExplodePoison();
49        }else if(isSlow && effectIns != null){
50            SlowEnemy(target);
51        }else{
52            Damage(target);
53        }
54
55        Destroy(gameObject);
56        GameObject effectAudio = (GameObject) Instantiate(hitEffectAudio, transform.position, transform.rotation);
57        Destroy(effectAudio, 5f);
58    }
59

```

Hình 4.17 Hàm đạn va chạm với mục tiêu

Tùy theo tính năng của từng loại đạn mà khi va chạm với mục tiêu sẽ gọi các hàm với các tính năng tương ứng



```

C# BulletController.cs ×
Assets > Scripts > C# BulletController.cs > BulletController > Seek
68    }
69
70    void Damage(Transform enemy){
71        Enemy e = enemy.GetComponent<Enemy>();
72        if(e != null){
73            e.TakeDamage(damageDealt);
74        }
75        Enemy2 e2 = enemy.GetComponent<Enemy2>();
76        if(e2 != null){
77            e2.TakeDamage(damageDealt);
78        }
79

```

Như hàm Damage() sẽ gây sát thương một mục tiêu cho quái và gọi đến hàm nhận sát thương của enemy và truyền vào giá trị sát thương mà đạn đó gây ra được.

```

C# Enemy.cs ×

Assets > Scripts > C# Enemy.cs > ⚡ Enemy > ⚡ Die

32
33
34     public void TakeDamage(float amount){
35         health -= amount;
36         HPBar.fillAmount = health/oldHealth;
37
38         if(health <= 0){
39             Die();
40         }
41     }

```

Hình 4.18 Hàm TakeDamage của enemy

Khi hàm TakeDamage() được gọi thì sẽ tiến hành trừ máu hiện tại với lượng sát thương nhận vào sau đó, hiển thị thanh máu theo phần trăm tương ứng. Nếu như máu của quái về 0 thì sẽ gọi hàm Die() để quái chết.

```

C# BulletController.cs ×

Assets > Scripts > C# BulletController.cs > ⚡ BulletController > ⚡ Seek

81     void SlowEnemy(Transform enemy){
82         Enemy e = enemy.GetComponent<Enemy>();
83         if(e != null){
84             e.TakeSlow(5f);
85             Enemy.timeSlow = timeSlowEffect;
86         }
87         Enemy2 e2 = enemy.GetComponent<Enemy2>();
88         if(e2 != null){
89             e2.TakeSlow(5f);
90             Enemy2.timeSlow = timeSlowEffect;
91         }
92     }

```



```

C# Enemy.cs ×

Assets > Scripts > C# Enemy.cs > ⚡ Enemy > ⚡ Die

82     }
83     public void TakeSlow(float speedSlow){
84         speed = speedSlow;
85     }

```

Hình 4.19 Hàm làm chậm tốc độ di chuyển của quái

Với đạn làm chậm mục tiêu thì hàm này sẽ truyền vào giá trị tốc độ bị làm chậm của mục tiêu và thời gian hiệu ứng làm chậm này tồn tại.

The screenshot shows two code snippets from Unity's script editor:

BulletController.cs

```

C# BulletController.cs ×
Assets > Scripts > C# BulletController.cs > BulletController > Seek
93
    1 reference
94     void ExplodePoison(){
95         Collider[] explodedPoisonObjects = Physics.OverlapSphere(transform.position, explosionRange);
96         foreach (Collider item in explodedPoisonObjects)
97         {
98             if(item.tag == "Enemy"){
99                 PoisonEnemy(item.transform);
100            }
101        }
102    }
103
104    1 reference
105    void PoisonEnemy(Transform enemy){
106        Enemy e = enemy.GetComponent<Enemy>();
107        if(e != null){
108            e.TakePoisonEffect(damageDealt,timePoisonEffect);
109        }
110        Enemy2 e2 = enemy.GetComponent<Enemy2>();
111        if(e2 != null){
112            e2.TakePoisonEffect(damageDealt,timePoisonEffect);
113        }
    }

```

Enemy.cs

```

C# Enemy.cs ×
Assets > Scripts > C# Enemy.cs > Enemy > Die
42
    1 reference
43     public void TakePoisonEffect(float amount, int time){
44         timePoison += time;
45         damagePoison = amount;
46         StartCoroutine(TakePoisonDamage());
    }
47
    1 reference
48     IEnumerator TakePoisonDamage(){
49         while (timePoison>0)
50         {
51             health -= health - oldHealth*damagePoison;
52             HPBar.fillAmount = health/oldHealth;
53             if(health <= 0){
54                 Die();
55                 yield break;
    }
56
57             yield return new WaitForSeconds(1f);
58             timePoison-=1;
    }

```

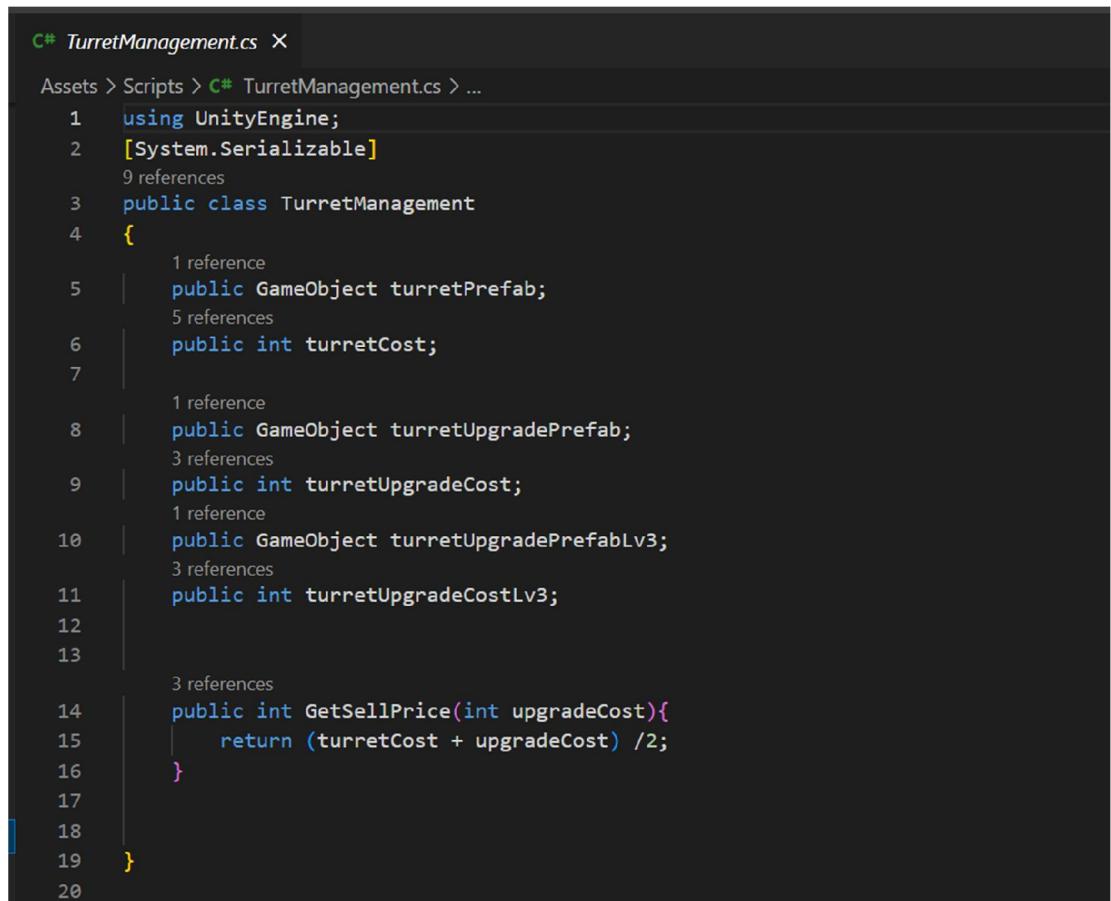
Hình 4.20 Hàm gây sát thương độc cho quái

Với loại đạn gây sát thương độc cho quái thì sẽ gọi hàm TakePoisonEffect() của quái và truyền vào giá trị phần trăm máu sẽ mất sau mỗi giây và thời gian của

hiệu ứng, quái sau khi trúng đạn sẽ bị rút máu sau mỗi giây và sẽ tiếp tục bị như vậy cho đến khi hết thời gian hiệu ứng. Nếu trong khoảng thời gian hiệu ứng vẫn còn mà máu của quái đã trở về 0 thì sẽ gọi hàm Die() của quái để quái chết sau đó thoát khỏi vòng lặp While() để ngưng việc trừ máu.

4.3.4 Thiết kế xây dựng các trụ phòng thủ, nâng cấp và bán trụ:

4.3.4.1 Xây dựng các trụ phòng thủ:



```
C# TurretManagement.cs ×

Assets > Scripts > C# TurretManagement.cs > ...
1  using UnityEngine;
2  [System.Serializable]
3  public class TurretManagement
4  {
5      public GameObject turretPrefab;
6      public int turretCost;
7
8      public GameObject turretUpgradePrefab;
9      public int turretUpgradeCost;
10     public GameObject turretUpgradePrefabLv3;
11     public int turretUpgradeCostLv3;
12
13
14     public int GetSellPrice(int upgradeCost){
15         return (turretCost + upgradeCost) / 2;
16     }
17
18 }
19
20 }
```

Hình 4.21 Code đối tượng quản lý từng loại trụ

Class TurretManagement sẽ là đối tượng để lưu trữ prefab các trụ với các cấp độ khác nhau cũng như giá tiền của từng trụ.

```

C# TurretHolder.cs ×
Assets > Scripts > C# TurretHolder.cs > ...
1  using UnityEngine;
2
3  public class TurretHolder : MonoBehaviour
4  {
5      public TurretManagement cannonFort;
6      public TurretManagement cornFort;
7      public TurretManagement slowFort;
8      public TurretManagement poisonFort;
9      Building building;
10     void Start()
11     {
12         building = Building.instance;
13     }
14     public void CannonTurretOnSelect()
15     {
16         Debug.Log("click on cannon");
17         building.BuildTurretSelected(cannonFort);
18     }
19     public void CornTurretOnSelect()
20     {
21         Debug.Log("click on corn");
22         building.BuildTurretSelected(cornFort);
23     }
}

```

Hình 4.22 Class TurretHolder

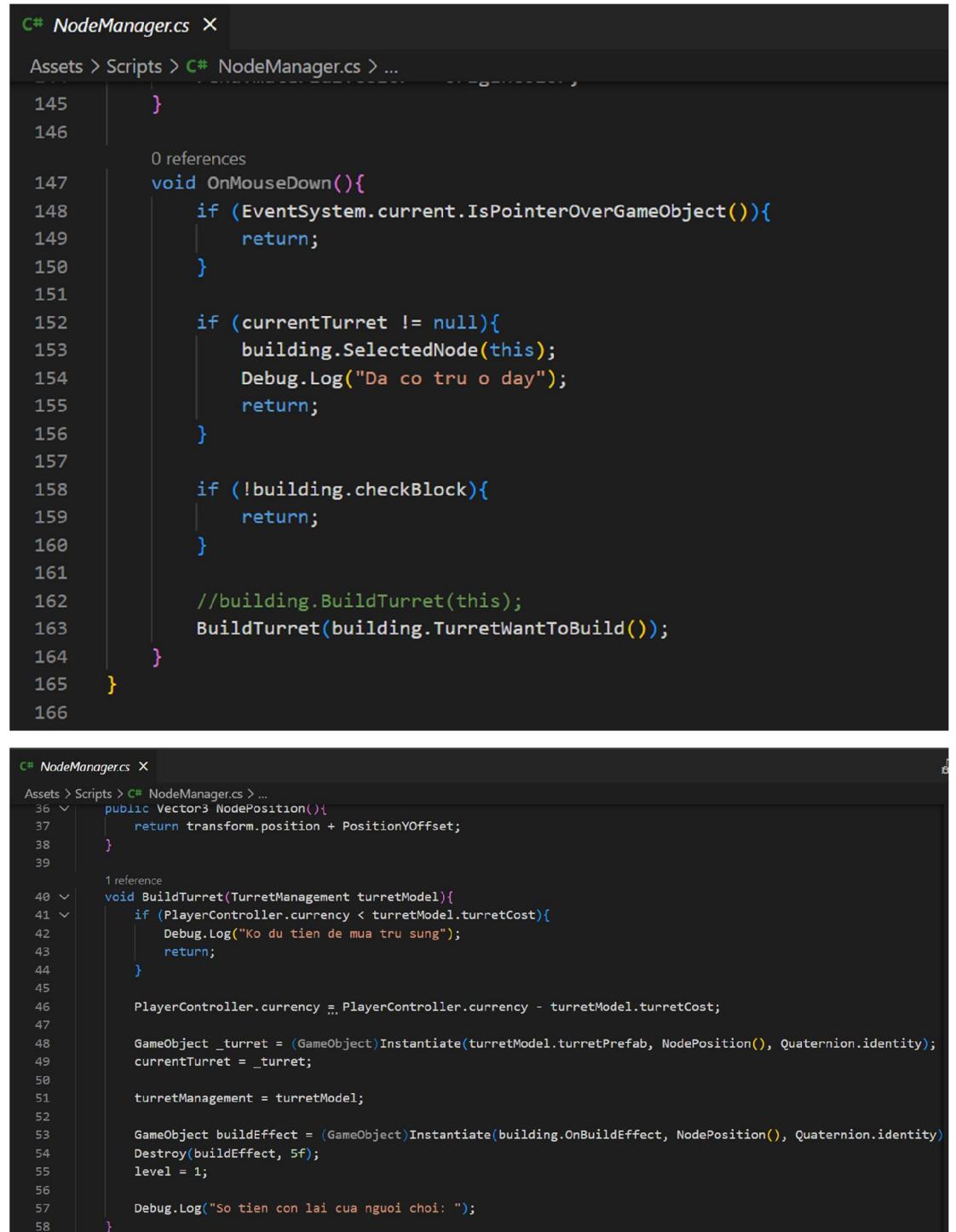
Class TurretHolder sẽ chứa các đối tượng TurretManagement với từng đối tượng TurretManagement là từng loại trụ khác nhau và khi người chơi nhấp vào loại trụ nào trên màn hình thì loại trụ đó sẽ được truyền qua hàm BuildTurretSelected() của đoạn script Building

```

C# Building.cs ×
Assets > Scripts > C# Building.cs > ...
38     upgradeMenuController.hideUI();
39 }
40
41     public void BuildTurretSelected(TurretManagement turretInfo)
42     {
43         turretOnBuild = turretInfo;
44         undoSelectedNode();
45     }
46
47     public TurretManagement TurretWantToBuild()
48     {
49         return turretOnBuild;
50     }
}

```

Sau đó, đoạn script Building sẽ có được loại trụ tương ứng, hàm TurretWantToBuild() sẽ trả về giá trị loại trụ đó.



```

C# NodeManager.cs X
Assets > Scripts > C# NodeManager.cs > ...
145     }
146
147     void OnMouseDown(){
148         if (EventSystem.current.IsPointerOverGameObject()){
149             return;
150         }
151
152         if (currentTurret != null){
153             building.SelectedNode(this);
154             Debug.Log("Da co tru o day");
155             return;
156         }
157
158         if (!building.checkBlock){
159             return;
160         }
161
162         //building.BuildTurret(this);
163         BuildTurret(building.TurretWantToBuild());
164     }
165 }
166

C# NodeManager.cs X
Assets > Scripts > C# NodeManager.cs > ...
36     public Vector3 NodePosition(){
37         return transform.position + PositionYOffset;
38     }
39
40     void BuildTurret(TurretManagement turretModel){
41         if (PlayerController.currency < turretModel.turretCost){
42             Debug.Log("Ko du tien de mua tru sung");
43             return;
44         }
45
46         PlayerController.currency -= PlayerController.currency - turretModel.turretCost;
47
48         GameObject _turret = (GameObject)Instantiate(turretModel.turretPrefab, NodePosition(), Quaternion.identity);
49         currentTurret = _turret;
50
51         turretManagement = turretModel;
52
53         GameObject buildEffect = (GameObject)Instantiate(building.OnBuildEffect, NodePosition(), Quaternion.identity);
54         Destroy(buildEffect, 5f);
55         level = 1;
56
57         Debug.Log("So tien con lai cua nguoi choi: ");
58     }

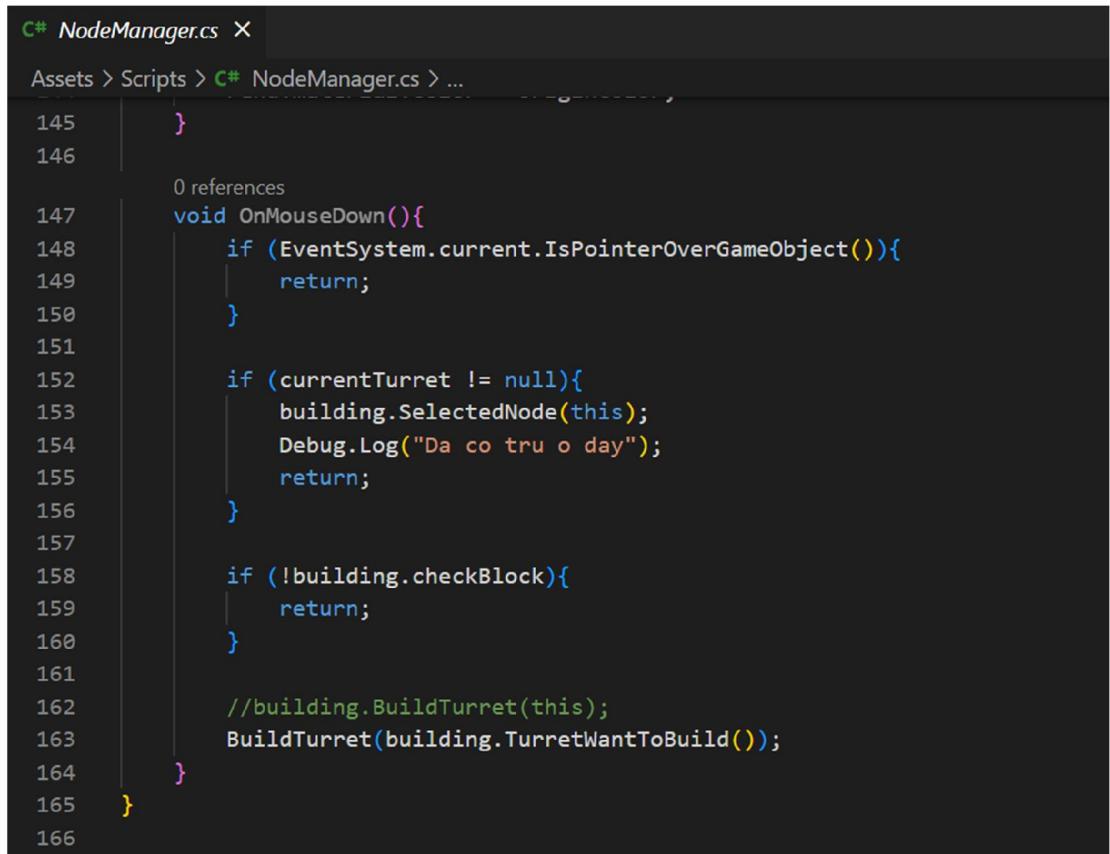
```

Hình 4.23 Class NodeManager

Ta sẽ tạo đoạn script NodeManager để quản lý việc xây dựng, nâng cấp cũng như bán trụ. Sau khi người chơi click chọn loại trụ muốn xây và nhấn đặt loại trụ đó và node thì sẽ tiến hành kiểm tra xem vị trí node đó đã có trụ hay chưa nếu chưa thì sẽ tiến hành gọi hàm BuildTurret() với tham số truyền vào là loại trụ lấy được từ script Building.

Hàm BuidTurret() sẽ tiến hành kiểm tra số tiền hiện có của người chơi, nếu số tiền đó không đủ thì sẽ gửi thông báo không đủ tiền cho người chơi. Còn nếu đủ thì sẽ tiến hành trừ tiền và bắt đầu xây dựng trụ phòng thủ có cấp độ 1.

4.3.4.2 Nâng cấp trụ:



```

C# NodeManager.cs X

Assets > Scripts > C# NodeManager.cs > ...
145     }
146
147     void OnMouseDown(){
148         if (EventSystem.current.IsPointerOverGameObject()){
149             return;
150         }
151
152         if (currentTurret != null){
153             building.SelectedNode(this);
154             Debug.Log("Da co tru o day");
155             return;
156         }
157
158         if (!building.checkBlock){
159             return;
160         }
161
162         //building.BuildTurret(this);
163         BuildTurret(building.TurretWantToBuild());
164     }
165 }
166

```

Sau khi người chơi nhấn vào trụ đã xây trên màn hình thì đoạn script sẽ kiểm tra xem đã có trụ tại node này hay chưa nếu đã có trụ thì sẽ gọi hàm SelectedNode() của script Building với tham số truyền vào là đoạn script NodeManager tại vị trí đó.

C# Building.cs X

Assets > Scripts > C# Building.cs > ...

```

21     2 references
22         public UpgradeMenuController upgradeMenuController;
23
24     1 reference
25         public void SelectedNode(NodeManager node){
26
27             if (onClickNode == node){
28                 undoSelectedNode();
29                 return;
30             }
31
32             onClickNode = node;
33             turretOnBuild = null;
34         }
35
36     2 references
37         public void undoSelectedNode(){
38             onClickNode = null;
39             upgradeMenuController.hideUI();
        }
```

Hàm SelectedNode() của script Building sau khi nhận được đoạn script NodeManager sẽ gọi hàm setUI() của đoạn script UpgradeMenuController với tham số truyền vào là đoạn script NodeManager.

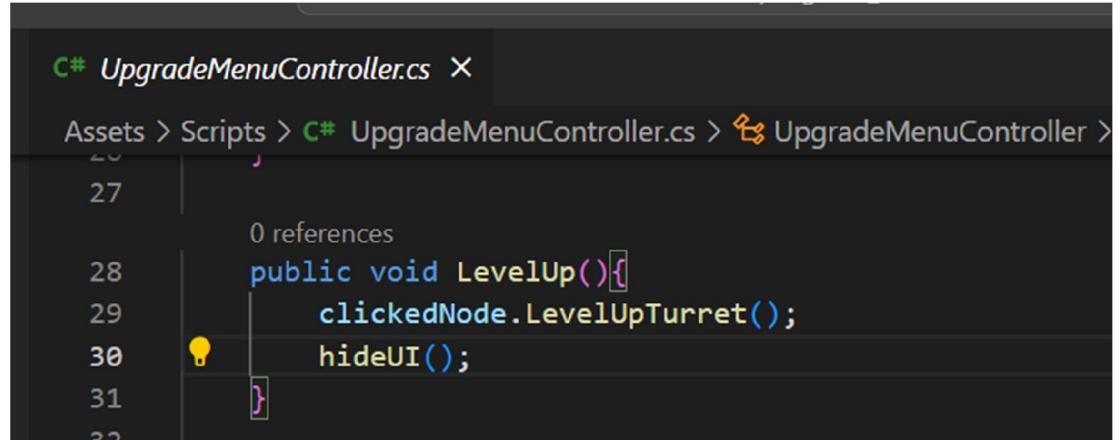
C# UpgradeMenuController.cs X

Assets > Scripts > C# UpgradeMenuController.cs > ...

```

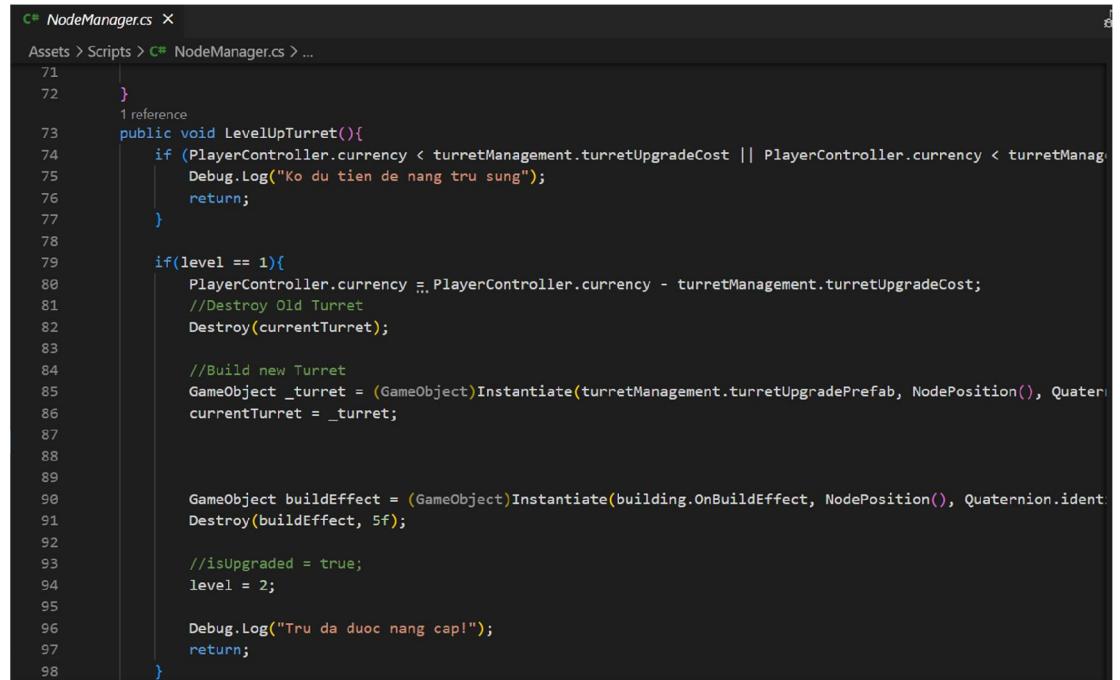
7
8     4 references
9         private NodeManager clickedNode;
10
11    2 references
12        public GameObject menuUI;
13
14    1 reference
15        public void setUI(NodeManager clickedNode){
16
17            this.clickedNode = clickedNode;
18
19            transform.position = this.clickedNode.NodePosition();
20
21        }
```

Hàm setUI() của đoạn script UpgradeMenuController sau khi nhận đoạn script NodeManager được truyền qua sẽ sử gọi lại hàm NodePosition() của đoạn script đó. Đây là hàm được sử dụng để biết được vị trí của node vừa click vào, nhờ đó hàm setUI() sẽ có thể hiện UI upgrade tại đúng vị trí của đoạn node vừa click.



```
C# UpgradeMenuController.cs
Assets > Scripts > C# UpgradeMenuController.cs > UpgradeMenuController >
27
28     public void LevelUp(){
29         clickedNode.LevelUpTurret();
30         hideUI();
31     }
32
```

Khi người chơi nhấn vào nút upgrade thì sẽ gọi đến hàm LevelUp(), hàm này sẽ gọi đến hàm LevelUpTurret() của script NodeManager để tiến hành nâng cấp trự



```
C# NodeManager.cs
Assets > Scripts > C# NodeManager.cs > ...
71
72 }
73 1 reference
74 public void LevelUpTurret(){
75     if (PlayerController.currency < turretManagement.turretUpgradeCost || PlayerController.currency < turretManag
76         Debug.Log("Ko du tien de nang tru sung");
77     return;
78 }
79
80 if(level == 1){
81     PlayerController.currency -= PlayerController.currency - turretManagement.turretUpgradeCost;
82     //Destroy Old Turret
83     Destroy(currentTurret);
84
85     //Build new Turret
86     GameObject _turret = (GameObject)Instantiate(turretManagement.turretUpgradePrefab, NodePosition(), Quaternion.identity);
87     currentTurret = _turret;
88
89
90     GameObject buildEffect = (GameObject)Instantiate(building.OnBuildEffect, NodePosition(), Quaternion.identity);
91     Destroy(buildEffect, 5f);
92
93     //isUpgraded = true;
94     level = 2;
95
96     Debug.Log("Tru da duoc nang cap!");
97
98 }
```

```

C# NodeManager.cs ×
Assets > Scripts > C# NodeManager.cs > ...
97     return;
98 }
99 if(level == 2){
100     PlayerController.currency -= PlayerController.currency - turretManagement.turretUpgradeCostLv3;
101     //Destroy Old Turret
102     Destroy(currentTurret);
103
104     //Build new Turret
105     GameObject _turret = (GameObject)Instantiate(turretManagement.turretUpgradePrefabLv3, NodePosition(), Quaternion.identity);
106     currentTurret = _turret;
107
108     GameObject buildEffect = (GameObject)Instantiate(building.OnBuildEffect, NodePosition(), Quaternion.identity);
109     Destroy(buildEffect, 5f);
110
111     isUpgraded = true;
112
113     Debug.Log("Trụ đã được nâng cấp!");
114     level = 3;
115     return;
116 }
117 if(isUpgraded){
118     Debug.Log("Trụ đã đạt cấp tối đa");
119     return;
120 }
121

```

Hình 4.24 Code nâng cấp trụ

Sau khi hàm LevelUpTurret() được gọi thì sẽ tiến hành trừ tiền của người chơi, xóa đi trụ có cấp 1 và xây dựng prefab trụ cấp 2 tại vị trí node đó tương tự khi nâng lên trụ đến cấp 3. Sau khi trụ đạt đến cấp 3 thì là đã đạt cấp độ cao nhất.

4.3.5 Pause Game:

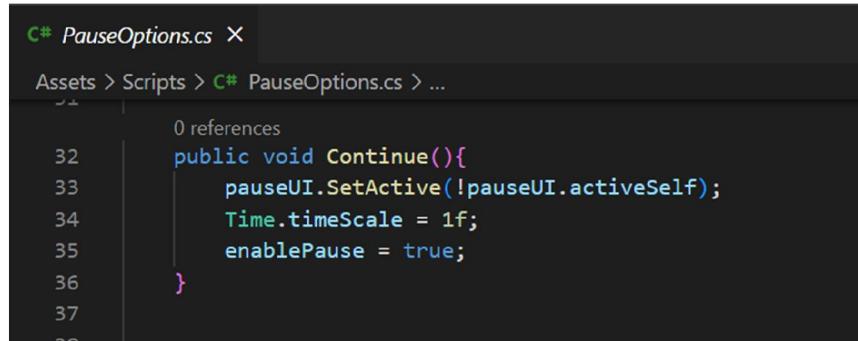
```

C# PauseOptions.cs ×
Assets > Scripts > C# PauseOptions.cs > ...
11     public string menuScene = "MainMenu";
12
13     public bool enablePause = true;
14
15     private void Update(){
16         if(Input.GetKeyDown(KeyCode.P)){
17             //ChangeState();
18             OnPause();
19         }
20
21
22     //Đổi trạng thái giữa pause và continue()
23     public void OnPause(){
24         if(enablePause){
25             pauseUI.SetActive(!pauseUI.activeSelf);
26             Time.timeScale = 0f;
27             enablePause = false;
28         }
29
30     }
31

```

Hình 4.25 Code tạm ngừng game

Khi người chơi nhấn phím P trên bàn phím thì hàm OnPause() sẽ được gọi, khi hàm được gọi thì UI Pause game sẽ được hiện ra và Time.timeScale = 0 sẽ làm cho toàn bộ thời gian và gameObject trong scene đó dừng lại.

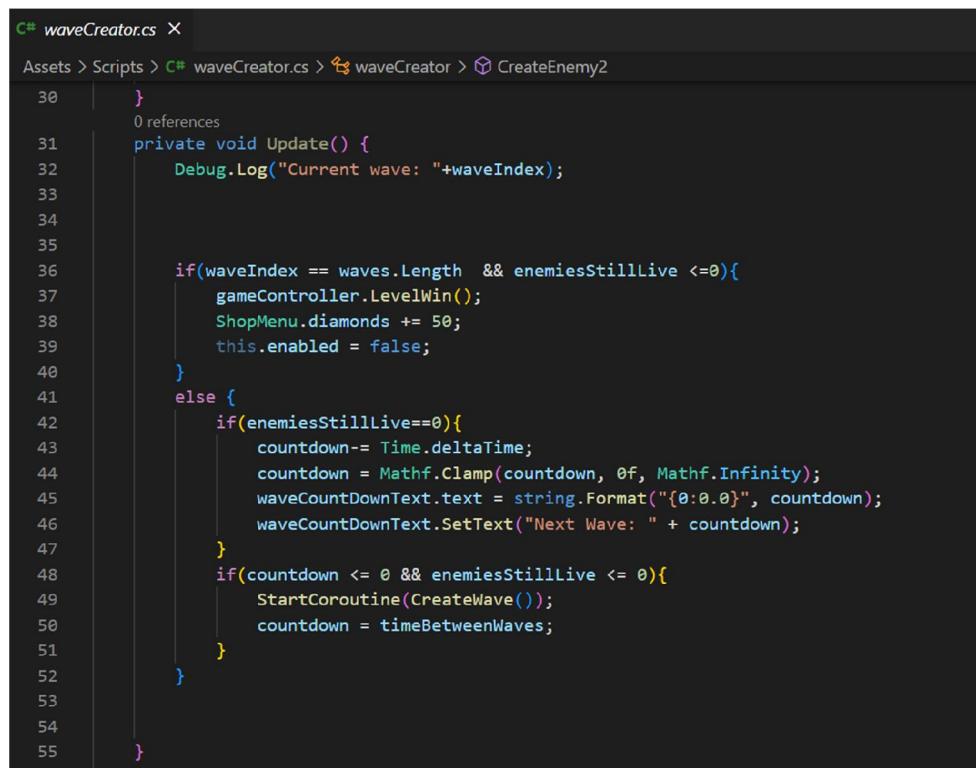


```
C# PauseOptions.cs ×
Assets > Scripts > C# PauseOptions.cs > ...
0 references
32     public void Continue(){
33         pauseUI.SetActive(!pauseUI.activeSelf);
34         Time.timeScale = 1f;
35         enablePause = true;
36     }
37
38
```

Hình 4.26 Code tiếp tục game

Khi người chơi nhấn vào nút continue thì hàm Continue() sẽ được gọi, UI Pause game sẽ được tắt đi và giá trị Time.timeScale sẽ được set lại = 1 thì các gameObject trong scene đó sẽ được tiếp tục chạy kể từ lúc dừng game.

4.3.6 Win Game:



```
C# waveCreator.cs ×
Assets > Scripts > C# waveCreator.cs > ↗ waveCreator > ↗ CreateEnemy2
0 references
30     }
31     private void Update() {
32         Debug.Log("Current wave: "+waveIndex);
33
34
35
36         if(waveIndex == waves.Length && enemiesStillLive <=0){
37             gameController.LevelWin();
38             ShopMenu.diamonds += 50;
39             this.enabled = false;
40         }
41         else {
42             if(enemiesStillLive==0){
43                 countdown-= Time.deltaTime;
44                 countdown = Mathf.Clamp(countdown, 0f, Mathf.Infinity);
45                 waveCountDownText.text = string.Format("{0:0.0}", countdown);
46                 waveCountDownText.SetText("Next Wave: " + countdown);
47             }
48             if(countdown <= 0 && enemiesStillLive <= 0){
49                 StartCoroutine(CreateWave());
50                 countdown = timeBetweenWaves;
51             }
52         }
53
54     }
55 }
```

```

C# GameController.cs ×

Assets > Scripts > C# GameController.cs > ...
33     Debug.Log("Game Over");
34 }
35
36     1 reference
37     public void LevelWin(){
38         gameFinished = true;
39         levelComplete.SetActive(true);
40         Debug.Log("LEVEL WON");
41     }
42

```

Hình 4.27 Code win game

Sau khi người chơi đã đi đến đợt quái cuối cùng và tiêu diệt được hết tất cả quái còn sống thì ta sẽ tiến hành cộng kim cương cho người chơi và gọi hàm LevelWin() của Script gameController. Hàm này sẽ gọi UI Win game hiện lên

```

C# LevelComplete.cs ×

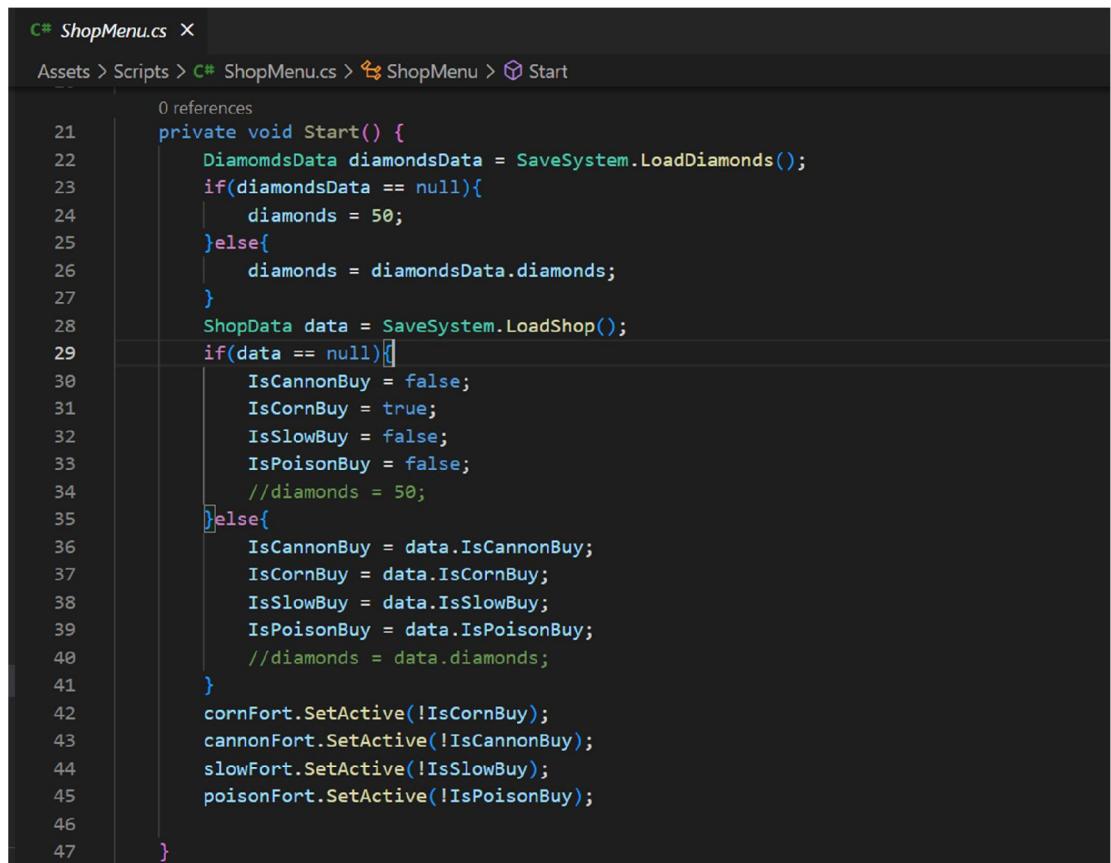
Assets > Scripts > C# LevelComplete.cs > ...
8     public int nextHighestLevel;
9     3 references
9     public SceneFader sceneFader;
10    0 references
10    public void onContinueButton(){
11        PlayerPrefs.SetInt("highestLevel",nextHighestLevel);
12        sceneFader.FadeToScene(nextLv);
13        SaveSystem.SavePlayer(this);
14        SaveSystem.SaveDiamonds();
15    }
16
16    0 references
17    public void RetryButtonClick(){
18        sceneFader.FadeToScene(SceneManager.GetActiveScene().name);
19    }
20
20    0 references
21    public void onMenuButton(){
22        SaveSystem.SaveDiamonds();
23        sceneFader.FadeToScene(menuScene);
24    }
25 }
25

```

Sau khi UI Win Game hiện ra thì khi người chơi bấm continue thì hàm onContinueButton() sẽ được gọi tới và tiến hành lưu trữ tiến độ của người chơi và số lượng kim cương hiện có vào file save sau đó sẽ chuyển scene cho người chơi đến màn chơi tiếp theo.

Tương tự khi bấm vào nút Main Menu thì hàm onMenuButton() sẽ được gọi và sẽ tiến hành lưu trữ kim cương và tiến độ màn chơi của người chơi vào file sau đó sẽ chuyển người chơi đến scene Main Menu.

4.4 Thiết kế cửa hàng:



```

C# ShopMenu.cs X
Assets > Scripts > C# ShopMenu.cs > ShopMenu > Start
0 references
21     private void Start() {
22         DiamondsData diamondsData = SaveSystem.LoadDiamonds();
23         if(diamondsData == null){
24             diamonds = 50;
25         }else{
26             diamonds = diamondsData.diamonds;
27         }
28         ShopData data = SaveSystem.LoadShop();
29         if(data == null){
30             IsCannonBuy = false;
31             IsCornBuy = true;
32             IsSlowBuy = false;
33             IsPoisonBuy = false;
34             //diamonds = 50;
35         }else{
36             IsCannonBuy = data.IsCannonBuy;
37             IsCornBuy = data.IsCornBuy;
38             IsSlowBuy = data.IsSlowBuy;
39             IsPoisonBuy = data.IsPoisonBuy;
40             //diamonds = data.diamonds;
41         }
42         cornFort.SetActive(!IsCornBuy);
43         cannonFort.SetActive(!IsCannonBuy);
44         slowFort.SetActive(!IsSlowBuy);
45         poisonFort.SetActive(!IsPoisonBuy);
46     }
47 }

```

Hình 4.28 Code cửa hàng mua các loại trụ

Khi người chơi nhấn vào nút Shop trên menu chính thì hệ thống sẽ chuyển người chơi sang scene cửa hàng. Khi chuyển sang scene cửa hàng thì hệ thống sẽ tiến hành kiểm tra xem có tồn tại dữ liệu file save lưu thông tin các loại trụ mà người chơi đã mua không và có file save lưu dữ liệu kim cương của người chơi không. Nếu có thì ta sẽ set Active UI của các loại súng trong cửa hàng giống với giá trị có trong file save và giá trị kim cương hiện lên sẽ cho giống với giá trị kim cương có trong file save. Còn nếu không có file save thì các giá trị trên sẽ được khởi tạo với giá trị mặc định ban đầu

```

C# ShopMenu.cs ×

Assets > Scripts > C# ShopMenu.cs > ShopMenu > Start
  ↗ REFERENCES
53     public void CornTurretBuy(){
54         if(diamonds < 50){
55             Debug.Log("Khong du kim cuong de mo khoa");
56         }else{
57             IsCornBuy = true;
58             diamonds -= 50;
59             Debug.Log("Da mo khoa thanh cong");
60             cornFort.SetActive(!IsCornBuy);
61         }
62     }
63 }

0 references
64     public void CannonTurretBuy(){
65         if(diamonds < 50){
66             Debug.Log("Khong du kim cuong de mo khoa");
67         }else{
68             IsCannonBuy = true;
69             diamonds -= 50;
70             Debug.Log("Da mo khoa thanh cong");
71             cannonFort.SetActive(!IsCannonBuy);
72         }
73 }

0 references
74     public void SlowTurretBuy(){
75         if(diamonds < 50){
76             Debug.Log("Khong du kim cuong de mo khoa");
77         }else{
78             IsSlowBuy = true;
79             diamonds -= 50;
80             Debug.Log("Da mo khoa thanh cong");
}

```

Hình 4.29 Code mua loại trù mới trong cửa hàng

Khi người chơi nhấn chọn mua loại trù nào trong cửa hàng thì các hàm CannonTurretBuy(), SlowTurretBuy() và PoisonTurretBuy() sẽ được gọi tương ứng. Sau khi mua thành công thì UI của các trù đã mua sẽ được setActive để biến mất sau đó tiến hành trừ kim cương của người chơi.

```

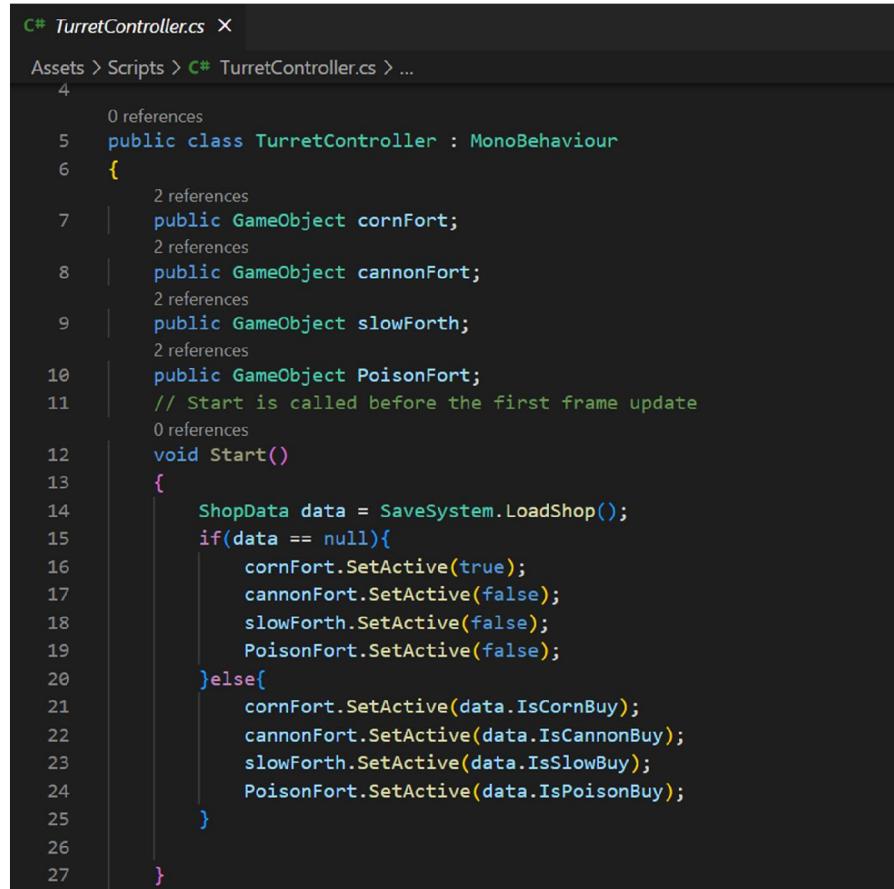
C# ShopMenu.cs ×

Assets > Scripts > C# ShopMenu.cs > ShopMenu > Start
  ↗ REFERENCES
45         slowFort.SetActive(!IsSlowBuy),
46         poisonFort.SetActive(!IsPoisonBuy);
47     }
48 }

0 references
49     public void onBtnBackClick(){
50         levelFader.FadeToScene("MainMenu");
51         SaveSystem.SaveShop();
52         SaveSystem.SaveDiamonds();
}

```

Sau khi bấm nút Back để trở về màn hình chính thì hàm onBtnBackClick() sẽ được gọi và sẽ tiến hành lưu lại các trù mà người chơi đã mua và số lượng kim cương còn lại của người chơi vào file save.



```

C# TurretController.cs ×

Assets > Scripts > C# TurretController.cs > ...
4
    0 references
5  public class TurretController : MonoBehaviour
6  {
7      2 references
8      public GameObject cornFort;
9      2 references
10     public GameObject cannonFort;
11     2 references
12     public GameObject slowForth;
13     2 references
14     public GameObject PoisonFort;
15     // Start is called before the first frame update
16     0 references
17     void Start()
18     {
19         ShopData data = SaveSystem.LoadShop();
20         if(data == null){
21             cornFort.SetActive(true);
22             cannonFort.SetActive(false);
23             slowForth.SetActive(false);
24             PoisonFort.SetActive(false);
25         }
26     }
27 }

```

Hình 4.30 Hiện trù đã mua trong game

Sau khi hoàn thành việc mua trù thì script TurretController sẽ setActive các UI các trù đã mua để hiện trong màn chơi dựa trên dữ liệu database các trù đã mua trong cửa hàng, tương ứng với những trù nào chưa được mua thì sẽ không được hiện lên UI trong gamePlay

4.5 Thiết kế cho phép người chơi chỉnh sửa cấu hình trong game:

```

15     0 references
16     void Start(){
17         resolutions = Screen.resolutions;
18         resolutionsDropdown.ClearOptions();
19         List<string> options = new List<string>();
20         int currentResolutionIndex = 0;
21         for (int i = 0; i < resolutions.Length; i++)
22         {
23             string option = resolutions[i].width+ " x " +resolutions[i].height;
24             options.Add(option);
25
26             if(resolutions[i].width == Screen.currentResolution.width &&
27                 resolutions[i].height == Screen.currentResolution.height)
28             {
29                 currentResolutionIndex = i;
30             }
31         }
32         resolutionsDropdown.AddOptions(options);
33         resolutionsDropdown.value = currentResolutionIndex;
34         resolutionsDropdown.RefreshShownValue();
35         setResolution(currentResolutionIndex);
36         setQuality(0);
37         Debug.Log("resolution: "+currentResolutionIndex);
38         //Sound
39         Load();
40     }
41     0 references
42     void Update(){}

```

Hình 4.31 Chính sửa resolution trong game

Khi người chơi khởi động game, game sẽ mặc định cho người chơi độ phân giải cao nhất là 1920x1080. Một vòng lặp sẽ được chạy để lấy các độ phân giải mà màn hình hoặc máy tính đó có thể chạy được, sau đó sẽ add vào một danh sách chuỗi options. Các options này sau đó sẽ được thêm vào Dropdown Button có trong setting game, hiển thị độ phân giải đang được chọn và các độ phân giải có thể chọn khác. Một điều kiện if của vòng lặp nhằm mục đích khi mà lần thứ i của vòng lặp, nếu độ phân giải của màn hình theo chiều ngang và dọc bằng với độ phân giải mà lần thứ i của vòng lặp thu được, biến currentResolutionIndex sẽ được gán bằng i. Sau đó ta sẽ gọi hàm setResolution() để chỉnh độ phân giải màn hình bằng đúng với độ phân giải cao nhất của màn hình.

```

1 reference
public void setQuality(int qualityIndex){
    QualitySettings.SetQualityLevel(qualityIndex);

}

0 references
public void setFullScreen(bool isFullScreen){
    Screen.fullScreen = isFullScreen;
}

1 reference
public void setResolution(int resolutionIndex){
    Resolution resolution = resolutions[resolutionIndex];
    Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);
}

```

Hình 4.32 Các hàm chỉnh độ phân giải và phóng full màn hình

setQuality sẽ được gắn vào sự kiện onClick của dropdown button trong phần cài đặt, hàm sẽ nhận một giá trị int tương ứng với số thứ tự của từng phần chất lượng hình ảnh được cấu hình sẵn trong Unity build player, mặc định của Unity build player sẽ có năm mức chất lượng hình ảnh lần lượt là Very low, Low, Medium, High, Very High và Ultra. Nhưng trong trường hợp của game này, nhóm đã lược bỏ các cấu hình Very low và Ultra, chỉ để lại ba mức thấp, trung bình và cao. setFullScreen() sẽ là hàm được gọi trong một check button của menu, giá trị đưa vào sẽ là biến bool true hoặc false, khi button được nhấn và cho kết quả true, màn hình sẽ được phóng full màn hình, khi false thì game sẽ thoát khỏi chế độ full màn hình và chuyển về chế độ cửa sổ. setResolution() sẽ được gắn vào dropdown menu của chỉnh độ phân giải như đã nói ở trên, hàm này sẽ sử dụng các thư viện có sẵn của Unity như Resolution để lấy độ phân giải, Screen dùng để chỉnh sửa màn hình để phục vụ cho hàm.

4.6 Âm thanh tổng cho game

```

0 references
public void setVolume(float volume){
    audioMixer.SetFloat("volume", volume);
    Save();
}

1 reference
public float GetMasterLevel(){
    float value;
    ...
    bool result = audioMixer.GetFloat("volume", out value);
    if(result){
        return value;
    }else{
        return 0f;
    }
}
2 references
private void Save(){
    PlayerPrefs.SetFloat("volume", GetMasterLevel());
}

1 reference
private void Load(){
    float value = PlayerPrefs.GetFloat("volume");
    audioMixer.SetFloat("volume", PlayerPrefs.GetFloat("volume"));
    slider.SetValueWithoutNotify(value);
}

```

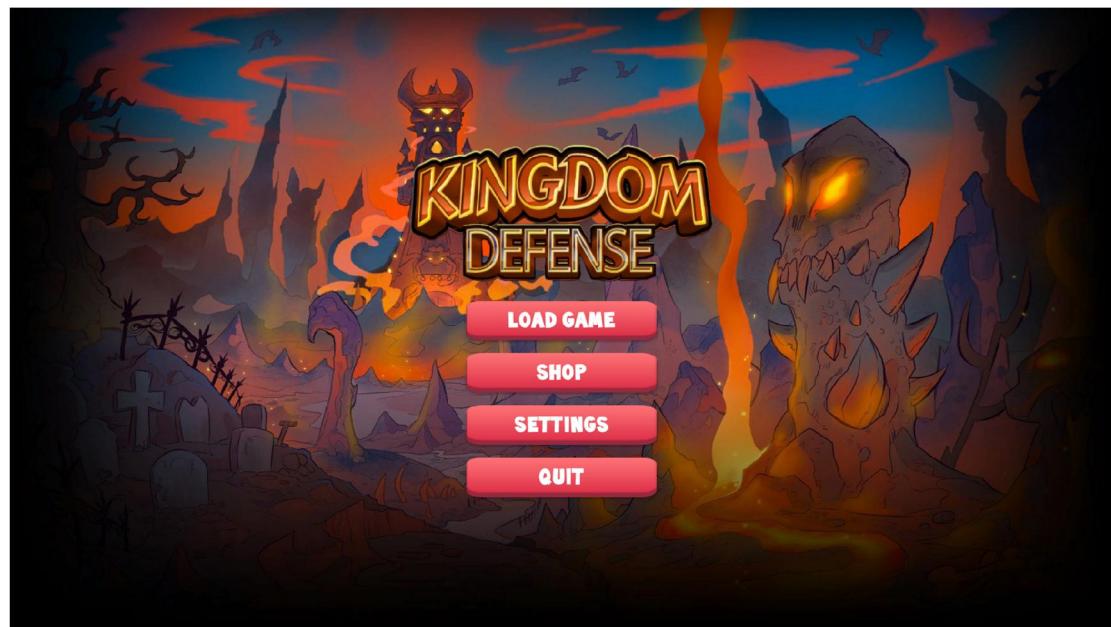
Hình 4.33 Code để lưu và chỉnh âm thanh trong game

Khi vào đến màn hình cài đặt của game, người chơi có thể chỉnh âm thanh cho toàn game nếu cảm thấy âm thanh trong game quá lớn hoặc quá nhỏ, chưa phù hợp với trải nghiệm trong game. Khi bắt đầu chỉnh âm thanh trong game, hàm setVolume sẽ hoạt động, âm thanh trong game sẽ được chỉnh dựa trên một Audio Mixer, audioMixer này sẽ hoạt động như một output của nguồn âm thanh, khi người chơi chỉnh giá trị của nguồn âm thanh này lên hoặc xuống, nó sẽ ảnh hưởng đến tất cả các âm thanh có trong trò chơi. Hai hàm Save và Load sẽ được dùng để lưu và lấy dữ liệu về sự điều chỉnh âm thanh của người chơi, do chỉ cần lưu một biến float là volume nên sẽ sử dụng PlayerPrefs, một chức năng có sẵn của Unity dùng cho việc lưu trữ

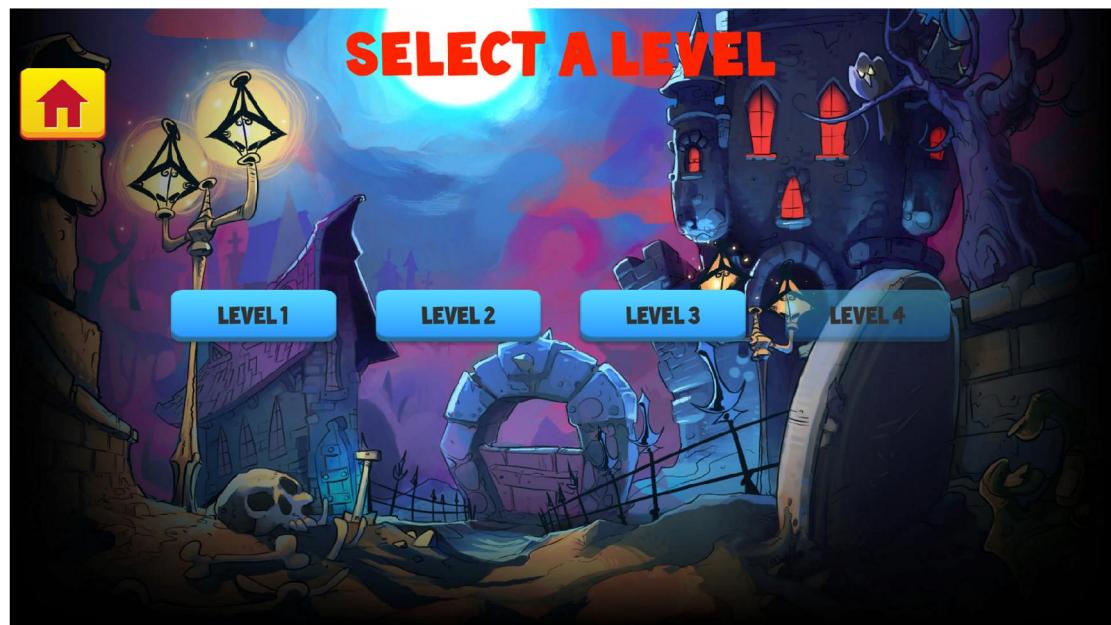
nhưng khá hạn chế. Khi người chơi bấm thoát khỏi màn hình cài đặt, hàm Save sẽ được gọi để lưu lại các thay đổi của người chơi.

✧ Kết quả của thực nghiệm hệ thống

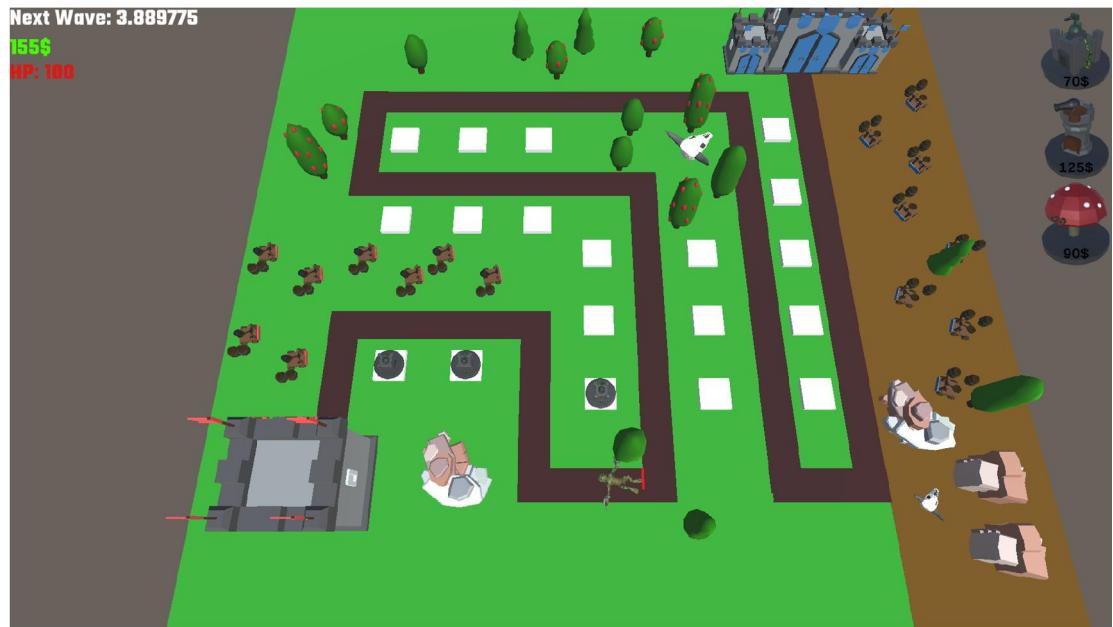
4.7 Màn hình chính của game



4.8 Màn hình chọn màn chơi



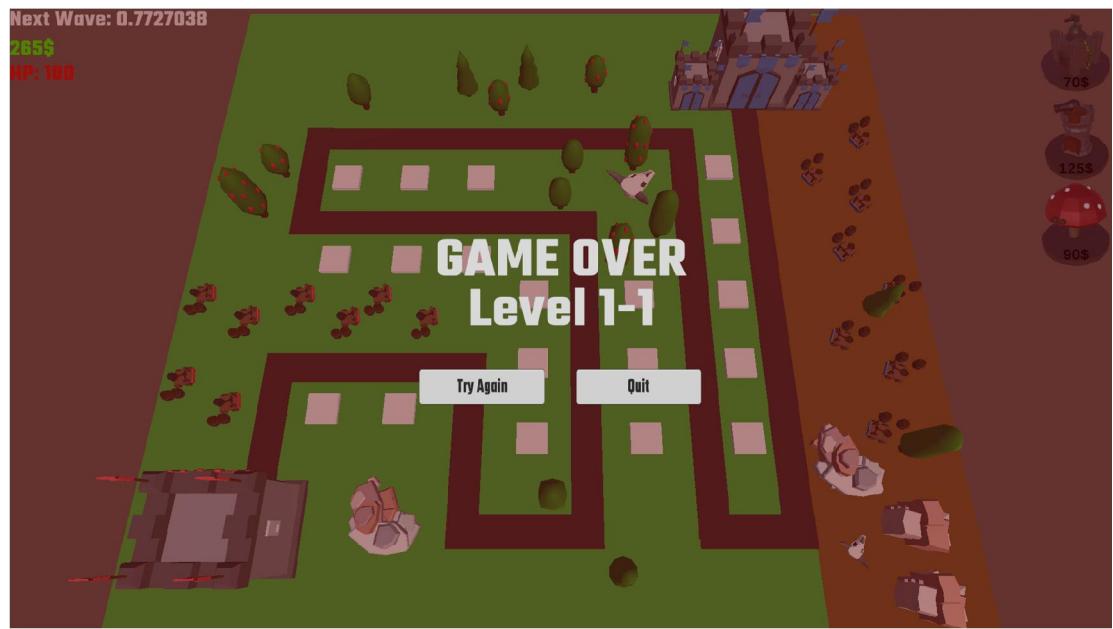
4.9 Màn hình một vòng chơi của game



4.10 PauseUI đơn giản của game



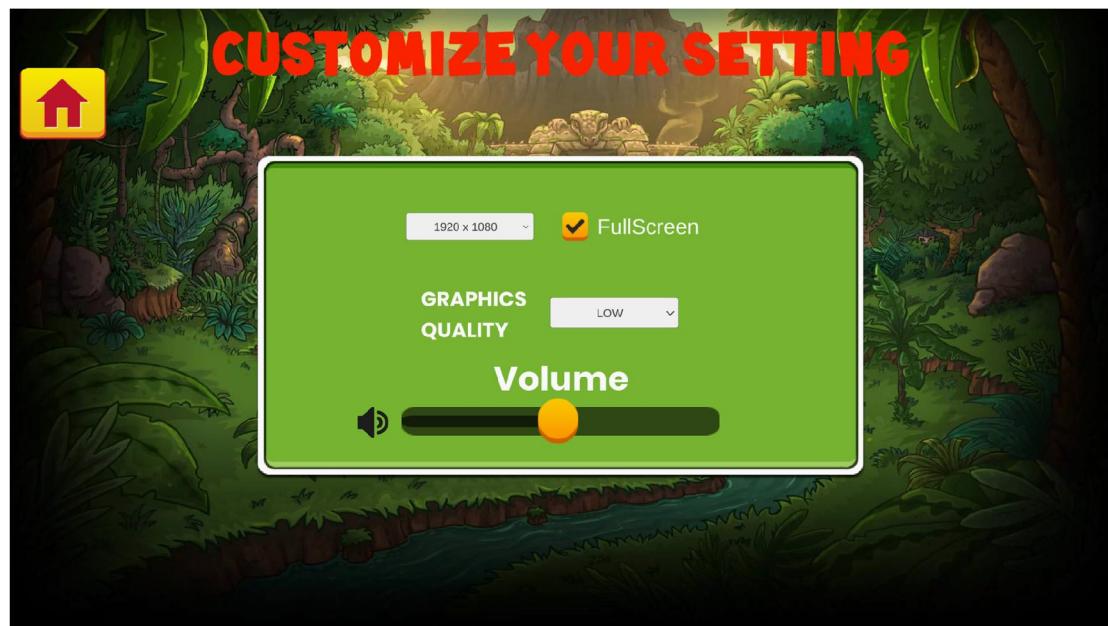
4.11 Màn hình game khi chơi thua



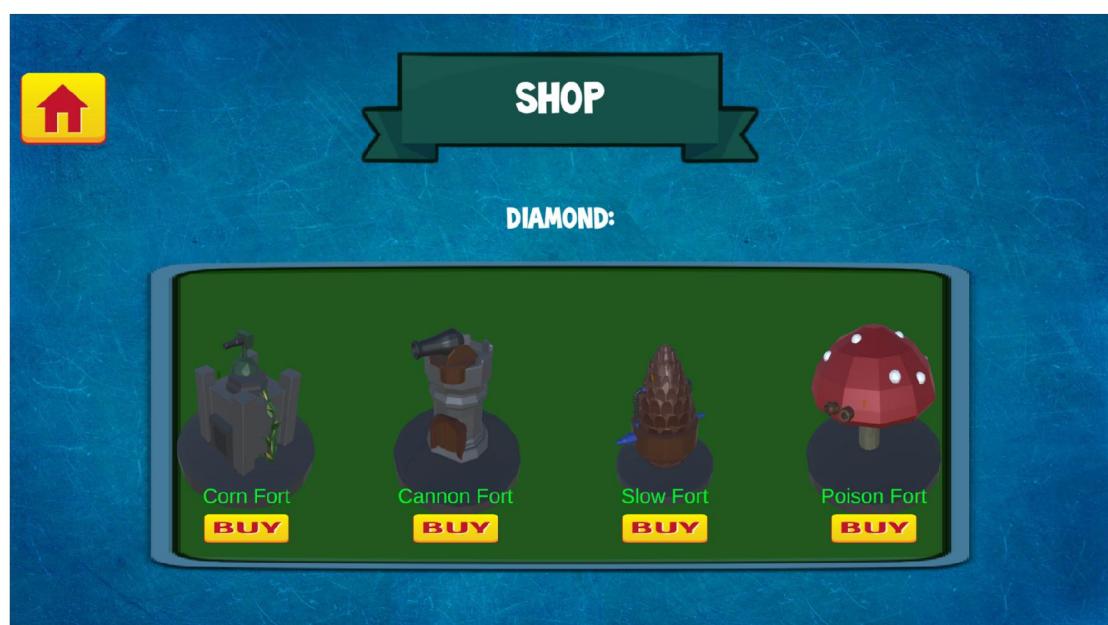
4.12 Khi chiến thắng một màn chơi



4.13 Menu cài đặt của game



4.14 Menu shop trong game



CHƯƠNG 5. KẾT LUẬN

5.1 Đã đạt được

Về cơ bản, nhóm đã làm được gần hết các phần dự tính trong game concept đã nộp từ trước.

Tạo được một menu để cho người chơi tương tác, menu này không quá phức tạp, cho người dùng có thể tùy chỉnh chất lượng hình ảnh, độ phân giải màn hình và âm thanh game.

Tạo được map cho game, map không làm chắp vá cho có, bối cảnh map phù hợp với bối cảnh của game.

Tạo được chuyển cảnh đơn giản cho game.

Tạo được animation cho trụ súng, animation khi đặt trụ xuống vị trí mong muốn. Chính được AI cho trụ ngắm và bắn. Làm một UI đơn giản dành cho việc bán và nâng cấp trụ.

Các quái vật có AI để di chuyển theo đường đi, tạo được animation đánh cho quái khi tới đích đến.

Tạo một menu shop, đơn vị tiền tệ của shop này là kim cương, người chơi sẽ được thưởng một lượng kim cương nhất định khi hoàn thành một màn chơi, người chơi sẽ sử dụng chúng để mở khóa trụ súng mới, giúp việc đương đầu với các màn chơi khó hơn.

Nhóm cũng đã làm được lưu trữ trong game, không chỉ sử dụng PlayerPrefs có sẵn trong Unity, nhóm đã có thể lưu các dữ liệu trong game thành các file .sav, các file này chứa các dữ liệu của người chơi và đã được mã hóa, giúp việc bảo mật file save trở nên tốt hơn.

5.2 Chưa đạt được

Trong game concept nhóm đã nộp, có còn phần làm ra các kỹ năng cơ bản cho người chơi như giật sét hay đóng băng quái, nhóm chưa làm được.

Các hình ảnh trong game đều được lấy miễn phí nên một số hình ảnh, các animation của trụ và quái không được đẹp cho lắm.

Game chưa đa dạng các loại trụ và quái để thu hút người chơi lâu dài.

Game còn hạn chế về mặt UI dành cho người chơi.

Game đang khá mất cân bằng, màn chơi có thể dễ quá hoặc quá khó để chơi.

5.3 Hướng phát triển

Định hướng tương lai là làm thêm kỹ năng cho người chơi để qua màn dễ hơn.

Thêm đa dạng các loại trụ súng, đặc biệt là trụ nhà lính, sẽ cho ra các binh sĩ để cản đường quái vật, câu thời gian cho trụ súng bắn thêm sát thương.

Làm thêm các cutscene khi chuyển cảnh giữa các màn chơi để cốt truyện game được rõ ràng hơn.

Thêm hiệu ứng cho trụ súng và thêm các animation cho quái.

Giải quyết các vấn đề tồn đọng trong phần chưa đạt được của sản phẩm.

TÀI LIỆU THAM KHẢO

1. <https://www.iostream.co/article/am-thanh-trong-unity-tf3UK1>
2. <https://caodang.fpt.edu.vn/tin-tuc-poly/tin-da-nang/tim-hieu-ve-trigger-trong-lap-trinh-game-unity.html#:~:text=Trong%20c%C3%B4ng%20c%E1%BB%A5%20l%E1%BA%ADp%20tr%C3%A1Cnh,c%C3%A1c%20%C4%91%E1%BB%91i%20t%C6%B0%E1%BB%A3ng%20g%E1%BA%B7p%20nhau>
3. <https://viblo.asia/p/co-ban-ve-engine-vat-ly-trong-unity-3OEqGj1PM9bL>
4. <https://www.youtube.com/playlist?list=PLPV2KyIb3jR4u5jX8za5iU1cqnQPmbzG0>