BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙡•🙣



**BÀI TẬP NHÓM MÔN HỌC LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**CÀI ĐẶT GAME TOP DOWN SHOOTING**

**Giảng viên hướng dẫn: ThS. Phạm Thị Kim Ngoan**

**Sinh viên thực hiện:**

* + - * **64132832 Nguyễn Hoàng Tuấn**
      * **64130880 Phạm Gia Huy**
      * **64130603 Nguyễn Nhật Hào**

**Khánh Hòa, tháng 5/2024**

**MỤC LỤC**

[I. Nêu bài toán: 3](#_Toc121475168)

[1. Giới thiệu về game : 3](#_Toc121475169)

[2. Yêu cầu thực hiện: 3](#_Toc121475170)

[II. Cách giải quyết: 4](#_Toc121475171)

[1. Hướng giải quyết 4](#_Toc121475172)

[2. Quy trình thiết kế ứng dụng 4](#_Toc121475173)

[III. Cài đặt game 4](#_Toc121475174)

[Tài liệu tham khảo 24](#_Toc121475175)

[ĐÁNH GIÁ CÁC THÀNH VIÊN (A, B, C, D) 24](#_Toc121475176)

**I. Nêu bài toán:**

***1. Giới thiệu về game bắn súng top-down:***

Game bắn súng top-down là thể loại game hành động trong đó người chơi điều khiển nhân vật từ góc nhìn từ trên xuống. Lối chơi thường tập trung vào việc di chuyển nhân vật, nhắm mục tiêu và tiêu diệt kẻ thù bằng súng hoặc các loại vũ khí khác.

***2.Yêu cầu thực hiện:***

* Hiển thị thanh máu và thông tin khác cho người chơi.hiển thị điểm số và thông tin khác,tạo menu tạm dừng và menu kết thúc trò chơi.
* Sử dụng phần mềm đồ họa 2D để tạo sprite cho nhân vật, kẻ thù, vũ khí, vật phẩm, power-up, cảnh quan và các yếu tố khác trong game.
* Tạo nhạc nền và hiệu ứng âm thanh cho game.
* Tạo Di chuyển nhân vật của người chơi theo hướng nhập (phím WASD hoặc gamepad),hiển thị trạng thái sức khỏe của người chơi,xử lý việc thu thập vật phẩm và power-up,xử lý việc bắn súng và các hành động khác của người chơi.
* Tạo và di chuyển Enemy theo các mẫu di chuyển khác nhau,xử lý hành vi của kẻ thù, chẳng hạn như tấn công người chơi và né tránh đạn,thanh theo dõi trạng thái sức khỏe của Enemy,xử lý khi Enemy bị hạ và rơi ra vật phẩm
* Tạo và quản lý các loại vũ khí khác nhau,xử lý việc bắn đạn, bao gồm đường đạn, va chạm và sát thương.
* Lập trình hành vi thông minh hơn cho kẻ địch, chẳng hạn như ẩn nấp, phối hợp tấn công và phản ứng với hành động của người chơi.

**II. Cách giải quyết:**

***1.Hướng giải quyết:***

- Lựa chọn lập trình trên Winform và Code của C# ( Visual studio): Tiết kiệm thời gian, dễ thao tác.

***2. Quy trình thiết kế ứng dụng:***

* B1. Thiết kế giao diện cơ bản
* B2. Lập trình tính năng xử lý cơ bản
* B3. Lập trình tính năng xử lý bổ sung
* B4. Hoàn tất thiết kế giao diện

**III. Cài đặt game:**

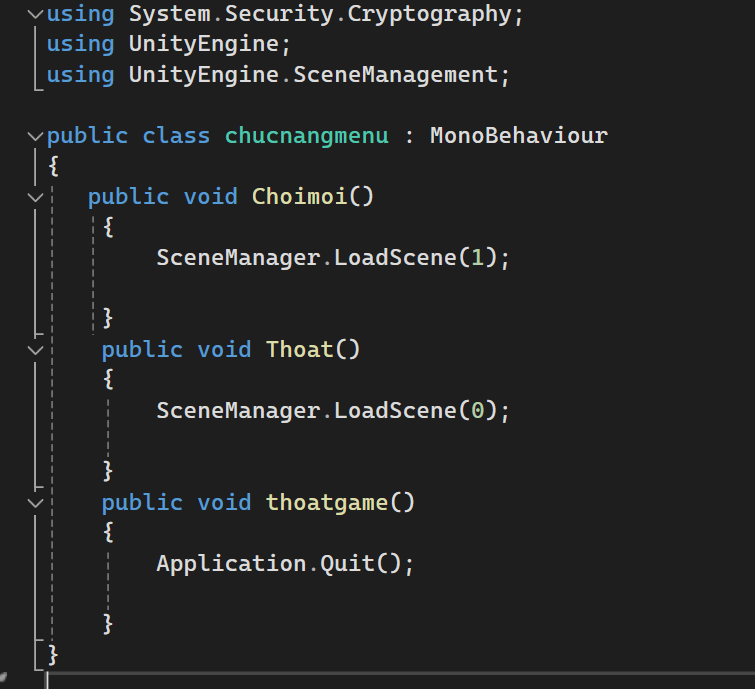
1. **Ý tưởng:**

* Một game bắn súng lấy bối cảnh trong một thế giới phép thuật với các sinh vật huyền bí và ma thuật mạnh mẽ. người chơi sẽ hóa thân thành nhân vật chính tập trung vào việc né tránh và bắn trả lại kẻ thù.
* Người chơi sẽ được hòa mình vào không khí nhộn nhịp kịch tính, kích thích và thú vị, rèn luyện kỹ năng phản xạ khi né và bắn trả kẻ thù.

1. **Tạo menu và nhân vật:**

* Menu:



****

Đây là một script C# có ba phương thức được sử dụng để quản lý các chức năng trong menu của một game Unity:

* Choimoi():

Phương thức này được gọi khi người dùng muốn bắt đầu một trò chơi mới.

Nó sử dụng SceneManager.LoadScene(1) để tải cảnh (scene) thứ 2 của game. Thông thường, cảnh đầu tiên sẽ là menu chính và cảnh thứ 2 sẽ là cảnh chính của trò chơi.

* Thoat():

Phương thức này được gọi khi người dùng muốn quay lại menu chính.

Nó sử dụng SceneManager.LoadScene(0) để tải lại cảnh đầu tiên, tức là menu chính.

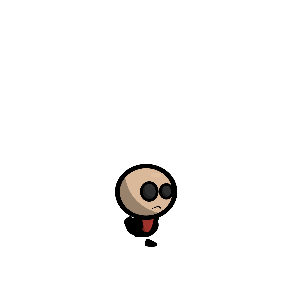
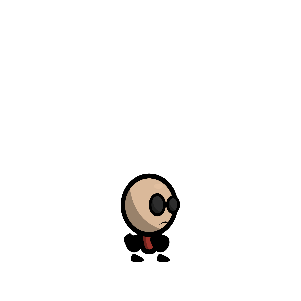
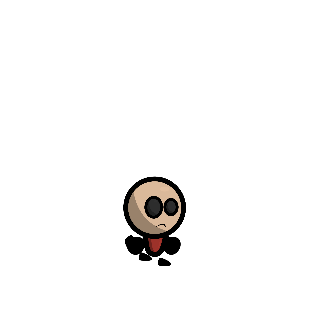
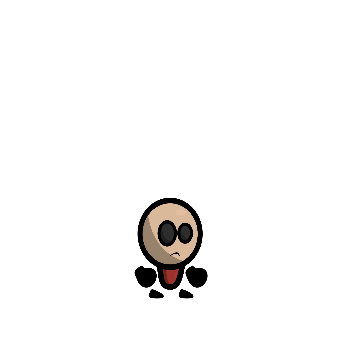
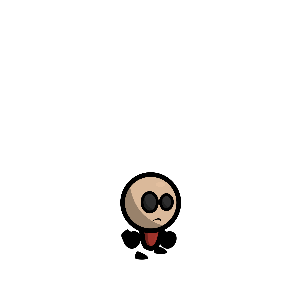
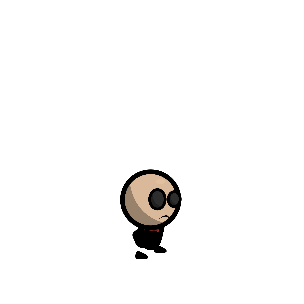
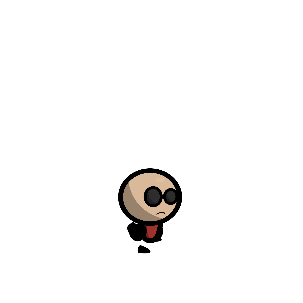
* thoatgame():

Phương thức này được gọi khi người dùng muốn thoát khỏi game.

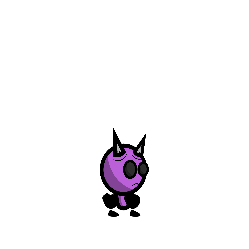
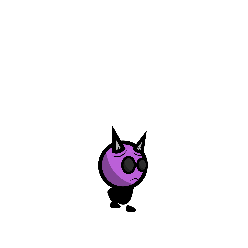
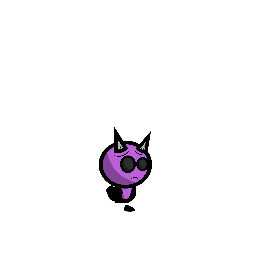
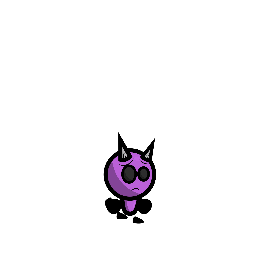
Nó sử dụng Application.Quit() để đóng ứng dụng game.

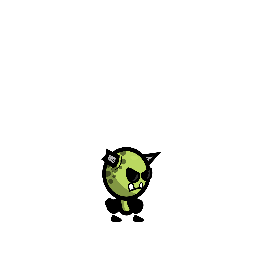
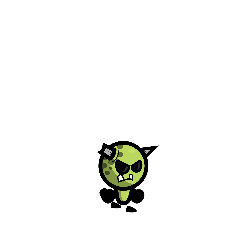
Cách thức hoạt động của script như sau:

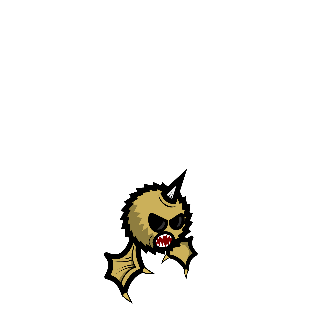
* Khi người dùng nhấn vào nút "Chơi mới" trong menu, phương thức Choimoi() sẽ được gọi, tải cảnh chính của game.
* Khi người dùng nhấn vào nút "Thoát" trong menu, phương thức Thoat() sẽ được gọi, tải lại menu chính.
* Khi người dùng nhấn vào nút "Thoát game", phương thức thoatgame() sẽ được gọi, đóng ứng dụng game.
* Nhân vật (tạo animation cho nhân vật). Mô phỏng dáng vẻ khi chạy của nhân vật.

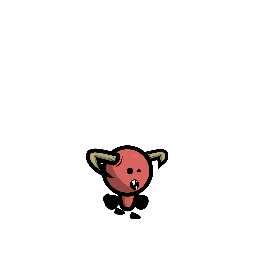
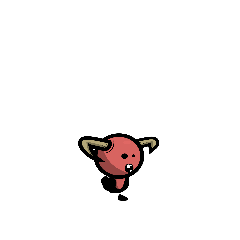
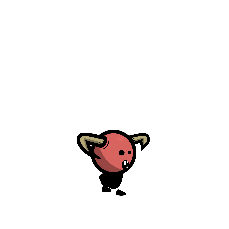
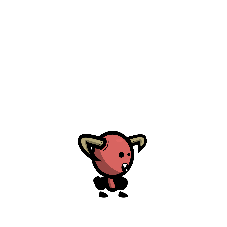
**    **

* **Kẻ địch:**

**   **

**   **

****

****

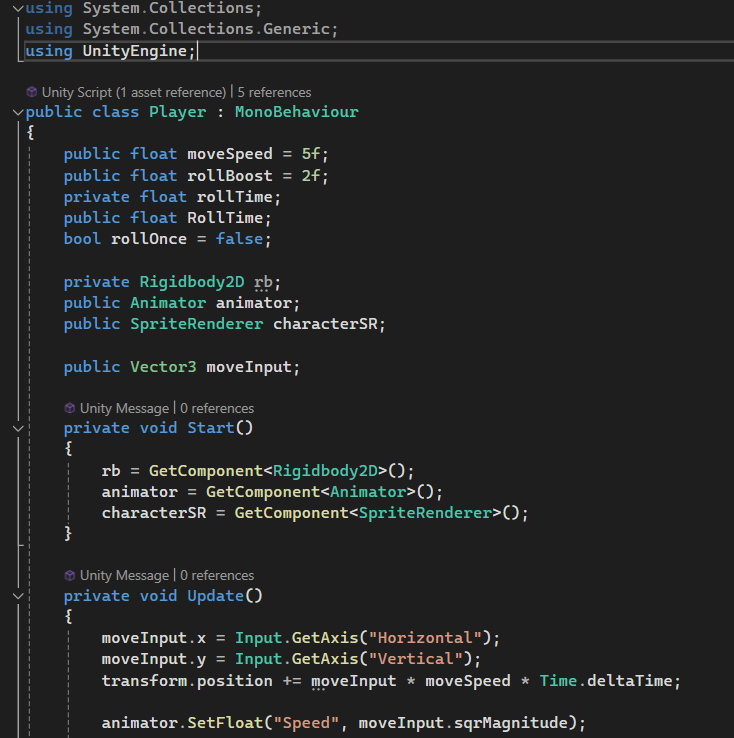
* **Vũ khí:**

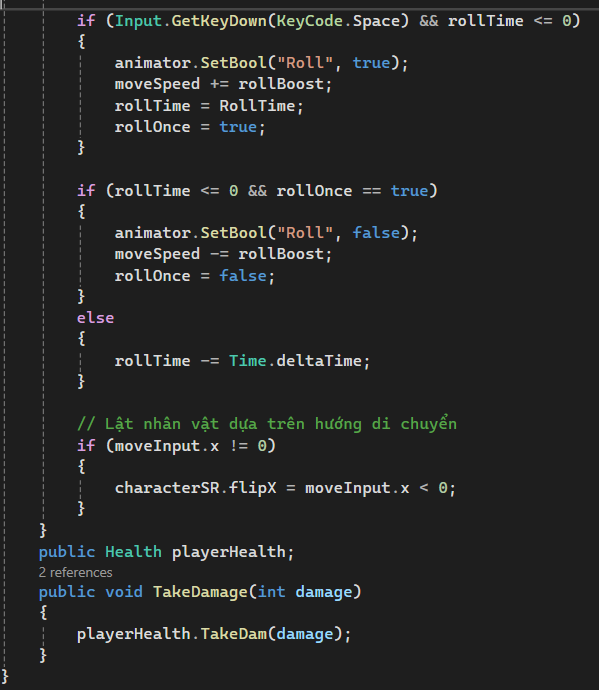
****

1. **Xử lý di chuyển:**
   1. **Xử lí di chuyển:**

* Viết hàm xử lí di chuyển:

- Viết hàm tạo chuyển động trong class move cho các Object

****

****

bắt đầu với một khối điều kiện kiểm tra hai điều kiện: xem liệu phím Space được nhấn (Input.GetKeyDown(KeyCode.Space)) và biến rollTime có nhỏ hơn hoặc bằng không không. Nếu cả hai điều kiện đều được thỏa mãn, một số hành động sẽ được thực hiện:

Một tham số boolean "Roll" của animator được đặt thành true (animator.SetBool("Roll", true)).

Biến moveSpeed được đặt thành rollBoost.

Biến rollOnce được đặt thành true.

Sau khối này, một điều kiện khác kiểm tra nếu rollTime nhỏ hơn hoặc bằng không và rollOnce là true. Nếu có, nó thực hiện các hành động sau:

Nó đặt tham số boolean "Roll" của animator thành false (animator.SetBool("Roll", false)).

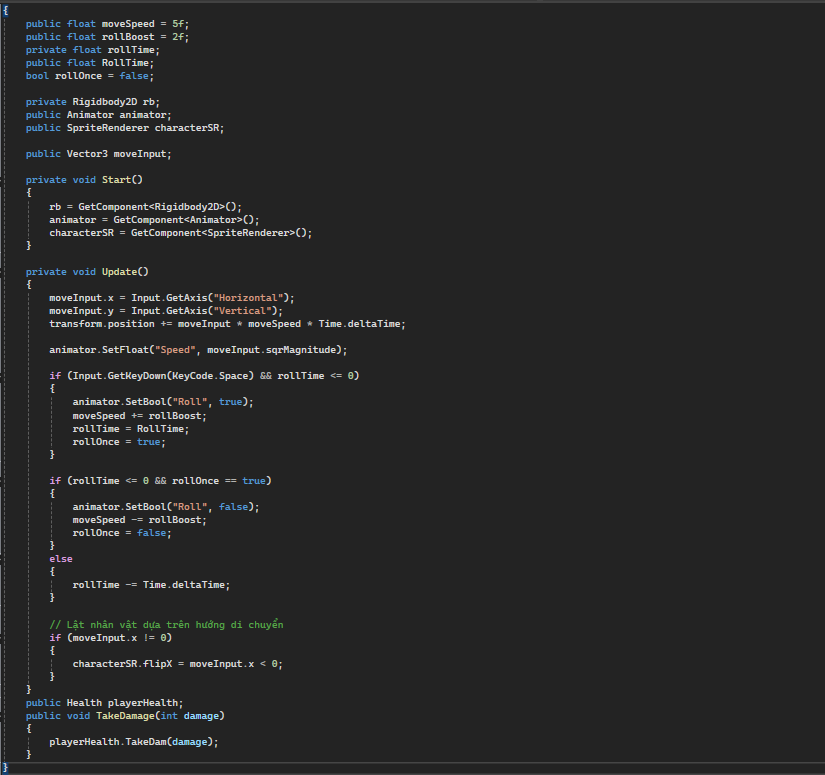
Nó đặt lại moveSpeed thành -rollBoost.

Nó đặt rollOnce thành false.

Nếu không có điều kiện nào ở trên được thỏa mãn, rollTime được giảm đi Time.deltaTime.

Dưới đoạn mã này, có một chú thích bằng tiếng Việt dịch sang tiếng Anh là "Flip the character based on the direction of movement." Sau đó là một câu lệnh điều kiện để lật hình ảnh của nhân vật trên trục x nếu moveInput.x nhỏ hơn không (characterSR.flipX = moveInput.x < 0).

-di chuyển nhân vật người chơi:

****

* đây là một script điều khiển nhân vật người chơi trong một trò chơi Unity. Nó quản lý di chuyển, lăn và hệ thống máu của nhân vật. Các tính năng chính bao gồm di chuyển, lăn và giảm máu.
* Các biến và thành phần:
* moveSpeed, rollBoost, rollTime, RollTime, rollOnce: Các biến để quản lý tốc độ di chuyển, tốc độ lăn, thời gian lăn và trạng thái lăn.
* rb (Rigidbody2D), animator, characterSR (SpriteRenderer): Các thành phần Unity được truy cập và sử dụng trong script.
* moveInput: Lưu trữ đầu vào di chuyển của người chơi.
* playerHealth: Một biến tham chiếu đến thành phần Health của nhân vật.
* Khởi tạo và cập nhật:
* Trong Start(), các thành phần Unity được lấy và gán vào các biến tương ứng.
* Trong Update(), script xử lý các hoạt động sau:
  + - * Lấy đầu vào di chuyển từ người chơi và cập nhật moveInput.
      * Cập nhật hoạt ảnh animator dựa trên moveInput.
      * Xử lý hành động lăn:
    - Nếu người chơi ấn phím Space và rollTime bằng 0, bắt đầu lăn bằng cách bật cờ rollOnce và tăng moveSpeed.
    - Nếu rollTime bằng 0 và rollOnce là true, kết thúc lăn bằng cách tắt cờ rollOnce và giảm moveSpeed.
    - Nếu rollTime lớn hơn 0, giảm rollTime theo thời gian.
* Lật hướng nhân vật dựa trên hướng di chuyển.
* Chức năng TakeDamage:

TakeDamage(int damage): Một phương thức để giảm máu của người chơi bằng cách gọi phương thức TakeDam() của thành phần Health.

-di chuyển Enemy:



**đây là một script điều khiển AI của Enemy, bao gồm các tính năng như di chuyển, tìm đường đi, bắn đạn và tìm kiếm người chơi. Nó sử dụng thư viện Pathfinding để tìm đường đi ngắn nhất và các biến để kiểm soát các hành vi của Enemy.**

**1.Các biến và thành phần:**

* roaming: Xác định xem Enemy có đi loanh quanh hay không.
* moveSpeed: Tốc độ di chuyển của Enemy.
* nextWPDistance: Khoảng cách tới điểm tiếp theo trên đường đi.
* updateContinuesPath: Xác định xem có cần cập nhật đường đi liên tục hay không.
* seeker: Thành phần Pathfinding để tìm đường đi.
* isShootable: Xác định xem Enemy có thể bắn đạn hay không.
* bullet: Đối tượng đạn của Enemy.
* bulletSpeed: Tốc độ của viên đạn.
* timeBtwFire: Thời gian giữa các lần bắn.
* fireCoolDown: Thời gian hồi chiêu của Enemy.
* reachDestination: Cờ để kiểm tra xem Enemy đã đến đích chưa.
* path: Đường đi mà Enemy sẽ đi.
* movecoroutine: Biến lưu trữ coroutine di chuyển.

**2. Khởi tạo và cập nhật:**

* Trong Start(), script bắt đầu tính toán đường đi cho Enemy.
* Trong Update(), script giảm thời gian hồi chiêu của Enemy và gọi phương thức EnemyFireBullet() nếu thời gian hồi chiêu đã hết.

**3. Chức năng bắn đạn:**

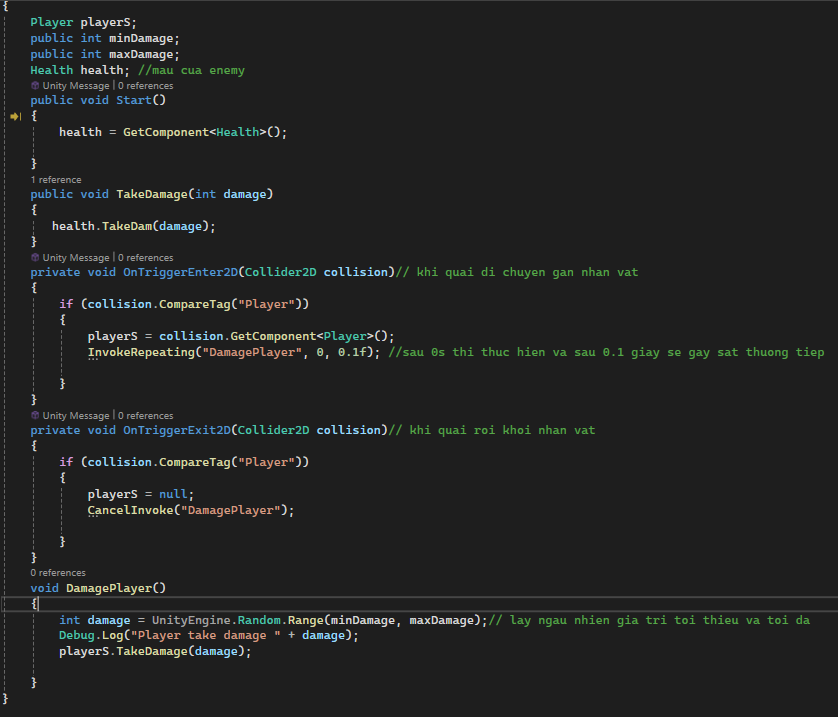
* EnemyFireBullet(): Tạo ra một viên đạn mới, lấy vị trí của người chơi, tính toán hướng và thêm lực vào viên đạn.

**4.Tính toán đường đi:**

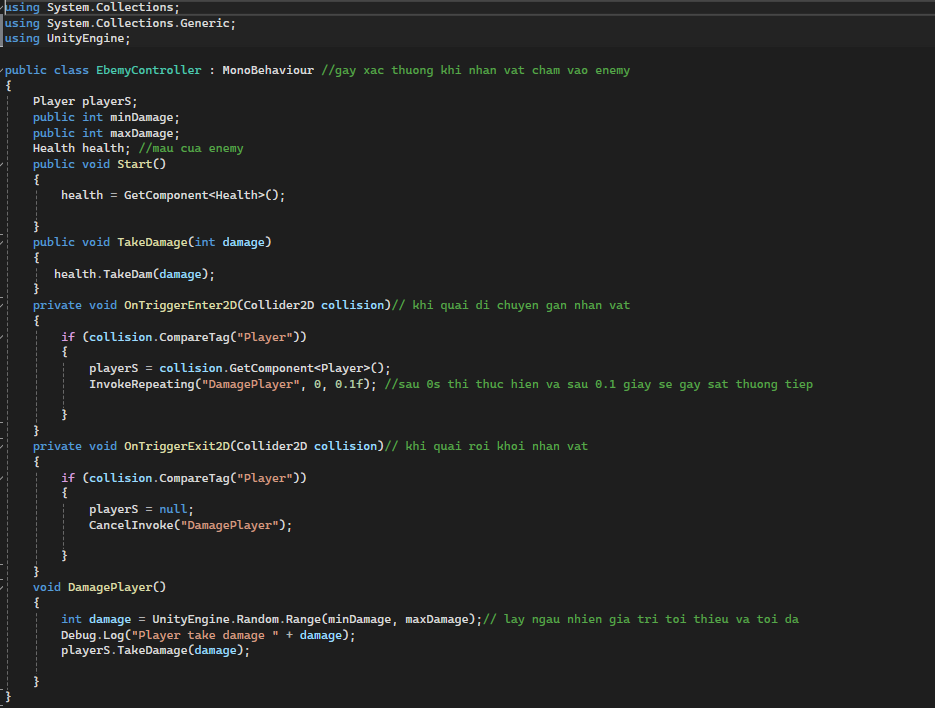
* CalculatePath(): Tìm đích mới và bắt đầu tính toán đường đi mới nếu seeker đã hoàn thành đường đi trước đó.
* OnPathComplete(Path p): Được gọi khi đường đi được tính toán xong. Nếu không có lỗi, lưu đường đi và gọi MoveToTarget().
* MoveToTarget(): Bắt đầu coroutine MoveToTargetCoroutine().
* MoveToTargetCoroutine(): Di chuyển Enemy theo đường đi đã tính toán. Cập nhật vị trí, kiểm tra xem đã đến điểm tiếp theo hay chưa, và kết thúc khi đến điểm cuối.

**5. Tìm điểm đích:**

* FindTarget(): Tìm điểm đích mới dựa trên chế độ "roaming" (đi loanh quanh) hoặc đi thẳng tới người chơi.
  1. **xử lí va chạm gây sát thương:**

****

Script này mô phỏng hành vi của một Enemy trong trò chơi, khi Player tiếp xúc với Enemy, sẽ gây ra sát thương cho Player theo một khoảng giá trị ngẫu nhiên đã được xác định trước.



Đây là một script C# được sử dụng để điều khiển Enemy trong một game Unity. Nó có các chức năng chính sau:

1.Gây sát thương cho Người chơi khi tiếp xúc:

* Khi Enemy va chạm với Người chơi (khi vào vùng Trigger), script sẽ bắt đầu gây sát thương cho Người chơi.
* Sát thương được tính ngẫu nhiên trong khoảng minDamage và maxDamage.
* Sát thương được gây cho Người chơi mỗi 0.1 giây bằng cách gọi playerS.TakeDamage(damage).

2.Xử lý sát thương cho Enemy:

* Trong Start(), script lấy thành phần Health của chính Enemy.
* Phương thức TakeDamage(int damage) được sử dụng để giảm máu của Enemy bằng cách gọi health.TakeDam(damage).

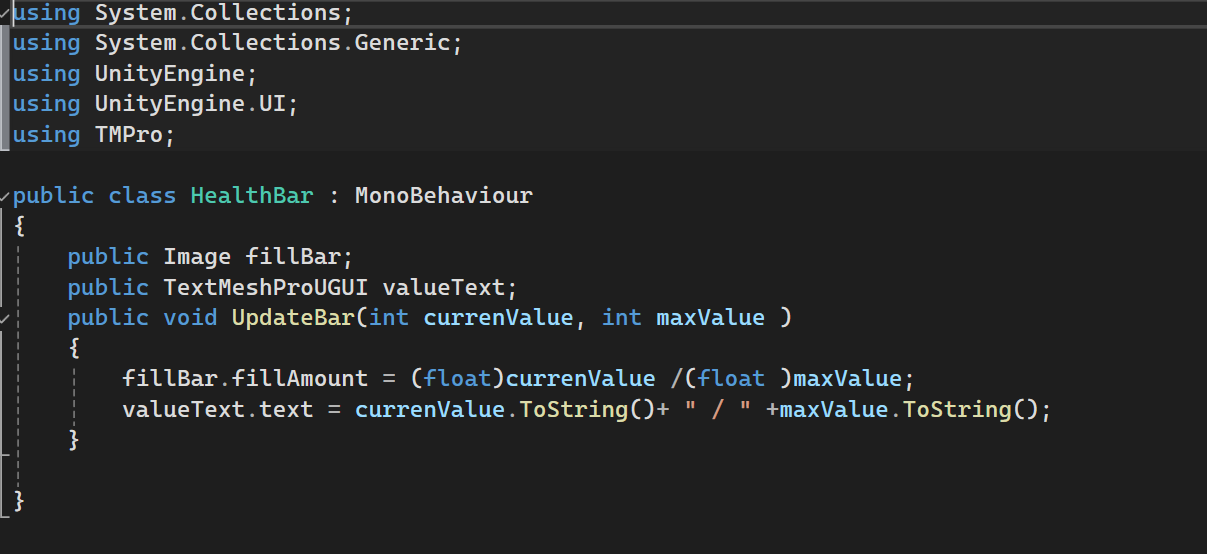
3.Quản lý va chạm:

* Trong OnTriggerEnter2D(Collider2D collision), khi Enemy va chạm với Người chơi, script sẽ lưu trữ tham chiếu đến thành phần Player và bắt đầu gọi DamagePlayer() mỗi 0.1 giây.
* Trong OnTriggerExit2D(Collider2D collision), khi Enemy rời khỏi vùng va chạm với Người chơi, script sẽ ngừng gọi DamagePlayer().

Cách thức hoạt động của script như sau:

1. Khi Enemy va chạm với Người chơi, script sẽ lưu trữ tham chiếu đến thành phần Player và bắt đầu gọi DamagePlayer() mỗi 0.1 giây.
2. Trong DamagePlayer(), script sẽ tính toán một giá trị sát thương ngẫu nhiên trong khoảng minDamage và maxDamage, sau đó gọi playerS.TakeDamage(damage) để giảm máu của Người chơi.
3. Khi Enemy rời khỏi vùng va chạm với Người chơi, script sẽ ngừng gọi DamagePlayer().
4. Nếu Enemy bị tấn công, phương thức TakeDamage(int damage) sẽ được gọi để giảm máu của Enemy bằng cách sử dụng thành phần Health.

**3.3thanh máu của nhân vật và tương tác với thanh máu:**

****

Đây là một script C# được sử dụng để hiển thị thanh máu (health bar) trong một game Unity. Bài phân tích này sẽ tập trung vào các thành phần chính của script:

1.Các biến và thành phần:

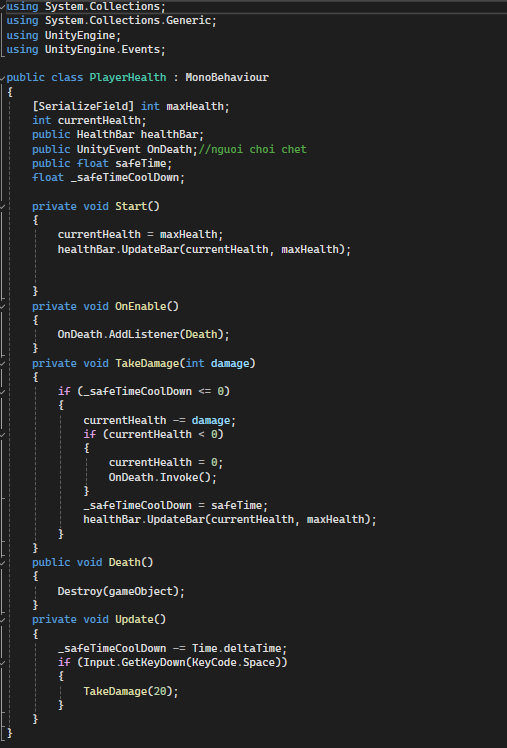
* fillBar: Một biến Image để hiển thị thanh máu.
* valueText: Một biến TextMeshProUGUI để hiển thị giá trị máu hiện tại và tổng máu.

2.Phương thức UpdateBar(int currenValue, int maxValue):

* Phương thức này được sử dụng để cập nhật trạng thái của thanh máu và giá trị máu.
* Nó nhận vào hai tham số: currenValue (giá trị máu hiện tại) và maxValue (giá trị máu tối đa).
* Trong phương thức này, script sẽ thực hiện các việc sau:
* Cập nhật fillAmount của fillBar bằng cách tính tỷ lệ currenValue so với maxValue. Điều này sẽ điều chỉnh độ dài của thanh máu.
* Cập nhật văn bản valueText bằng cách ghép chuỗi currenValue và maxValue với định dạng "currenValue / maxValue".

Cách hoạt động của script như sau:

* 1. Khi cần cập nhật trạng thái của thanh máu, script sẽ được gọi và nhận vào hai tham số: currenValue và maxValue.
  2. Dựa vào hai tham số này, script sẽ tính toán và cập nhật độ dài của fillBar và giá trị văn bản hiển thị trên thanh máu.
  3. Việc cập nhật thanh máu và giá trị văn bản sẽ được thực hiện ngay lập tức, đảm bảo rằng thanh máu và giá trị hiển thị luôn phản ánh đúng trạng thái của nhân vật.



Đây là một script C# được sử dụng để quản lý sức khỏe (health) của một nhân vật người chơi trong game (script này quản lý sức khỏe của nhân vật, cập nhật thanh máu, xử lý khi nhân vật bị tấn công và khi nhân vật chết). Dưới đây là phân tích chi tiết về các thành phần chính của script:

1.Các biến và thành phần:

* maxHealth: Lượng máu tối đa của nhân vật.
* currentHealth: Lượng máu hiện tại của nhân vật.
* healthBar: Một tham chiếu đến thành phần HealthBar để cập nhật thanh máu.
* OnDeath: Một sự kiện UnityEvent được kích hoạt khi nhân vật chết.
* safeTime: Thời gian miễn nhiễm sau khi bị thương.
* \_safeTimeCoolDown: Thời gian miễn nhiễm còn lại.

2.Khởi tạo và cập nhật:

* Trong Start(), lượng máu hiện tại được đặt bằng maxHealth và thanh máu được cập nhật.
* Trong OnEnable(), sự kiện OnDeath được đăng ký để xử lý sự kiện khi nhân vật chết.
* Trong Update(), \_safeTimeCoolDown được giảm theo thời gian. Khi nhấn phím Space, TakeDamage(20) được gọi để giảm máu của nhân vật.

3.Nhận sát thương và xử lý tử vong:

* TakeDamage(int damage): Phương thức này được gọi khi nhân vật bị tấn công. Nó kiểm tra xem thời gian miễn nhiễm còn lại hay không, sau đó giảm máu hiện tại. Nếu máu về 0 hoặc thấp hơn, sự kiện OnDeath được kích hoạt.
* Death(): Phương thức này được gọi khi sự kiện OnDeath được kích hoạt. Nó hủy đối tượng của nhân vật.

Cách thức hoạt động của script như sau:

1. Khi game bắt đầu, lượng máu hiện tại của nhân vật được đặt bằng maxHealth và thanh máu được cập nhật.
2. Khi nhân vật bị tấn công, TakeDamage(int damage) được gọi. Nếu thời gian miễn nhiễm đã qua, máu hiện tại sẽ bị giảm và thanh máu được cập nhật.
3. Nếu máu hiện tại bằng 0 hoặc thấp hơn, sự kiện OnDeath được kích hoạt và phương thức Death() được gọi để hủy đối tượng của nhân vật.
4. Trong Update(), thời gian miễn nhiễm còn lại được giảm và khi nhấn phím Space, nhân vật sẽ bị tấn công.

Hình ảnh thanh máu trong game:



**3.4viên đạn gây sát thương:**



Đây là một script C# được sử dụng để xử lý các viên đạn trong một game Unity. Dưới đây là phân tích chi tiết về các thành phần chính của script:

1.Các biến và thành phần:

* minDamage: Sát thương tối thiểu của viên đạn.
* maxDamage: Sát thương tối đa của viên đạn.
* goodSizeBullet: Cờ để xác định xem viên đạn là "lớn" hay "nhỏ".

2.Xử lý va chạm:

* Trong OnTriggerEnter2D(Collider2D collision), script sẽ kiểm tra xem viên đạn có va chạm với đối tượng nào không.
* Nếu viên đạn va chạm với Người chơi và không phải là "lớn", nó sẽ tính toán một giá trị sát thương ngẫu nhiên trong khoảng minDamage và maxDamage, sau đó gọi Player.TakeDamage(damage) để giảm máu của Người chơi.
* Nếu viên đạn va chạm với Enemy và là "lớn", nó sẽ tính toán một giá trị sát thương ngẫu nhiên trong khoảng minDamage và maxDamage, sau đó gọi EbemyController.TakeDamage(damage) để giảm máu của Enemy.
* Nếu viên đạn va chạm với Boss và là "lớn", nó sẽ tính toán một giá trị sát thương ngẫu nhiên trong khoảng minDamage và maxDamage, sau đó gọi BossHealth.TakeDamage(damage) để giảm máu của Boss.
* Sau khi gây sát thương, viên đạn sẽ bị hủy bỏ bằng cách gọi Destroy(gameObject).

Cách thức hoạt động của script như sau:

1. Khi viên đạn va chạm với một đối tượng khác (Người chơi, Enemy hoặc Boss), OnTriggerEnter2D(Collider2D collision) sẽ được gọi.
2. Script sẽ kiểm tra xem đối tượng va chạm có phải là Người chơi, Enemy hoặc Boss không.
3. Nếu là Người chơi và viên đạn không phải là "lớn", script sẽ tính toán một giá trị sát thương ngẫu nhiên và gọi Player.TakeDamage(damage) để giảm máu của Người chơi.
4. Nếu là Enemy hoặc Boss và viên đạn là "lớn", script sẽ tính toán một giá trị sát thương ngẫu nhiên và gọi EbemyController.TakeDamage(damage) hoặc BossHealth.TakeDamage(damage) để giảm máu của đối tượng tương ứng.
5. Sau khi gây sát thương, viên đạn sẽ bị hủy bỏ.

**3.5 hàm triệu hồi quái (spawner):**

****

Đây là một script C# được sử dụng để quản lý việc triển khai và mất máu của kẻ địch trong một game Unity. Dưới đây là phân tích chi tiết về các thành phần chính của script:

1.Các biến và thành phần:

* enemyPrefab: Prefab của kẻ địch.
* startTimeBtwSpawn: Thời gian giữa các lần triển khai kẻ địch.
* playerTransform: Vị trí của người chơi.
* timeBtwSpawn: Đếm ngược thời gian để triển khai kẻ địch kế tiếp.
* currentEnemies: Số lượng kẻ địch hiện có trên sân.
* maxEnemies: Số lượng tối đa kẻ địch có thể có trên sân cùng lúc.
* spawnRadius: Khoảng cách gần người chơi mà kẻ địch có thể xuất hiện.
* playerAlive: Cờ để kiểm tra xem người chơi có còn sống hay không.

2.Cập nhật và triển khai:

* Trong Update(), script kiểm tra xem có thể triển khai kẻ địch mới không (dựa trên timeBtwSpawn, currentEnemies, maxEnemies và playerAlive).
* Nếu có thể triển khai, phương thức SpawnEnemyNearPlayer() sẽ được gọi để tạo ra một kẻ địch mới.
* SpawnEnemyNearPlayer() sẽ tính toán vị trí triển khai dựa trên vị trí của người chơi và spawnRadius, sau đó tạo ra một đối tượng kẻ địch mới.
* Sau khi triển khai, currentEnemies được tăng lên 3.

3.Xử lý khi kẻ địch bị tiêu diệt:

* Phương thức EnemyDied() được gọi khi một kẻ địch bị tiêu diệt.
* Trong phương thức này, currentEnemies được giảm đi 1.
* Nếu số lượng kẻ địch hiện tại là 7 hoặc ít hơn, phương thức SpawnEnemyNearPlayer() sẽ được gọi để triển khai thêm kẻ địch mới.

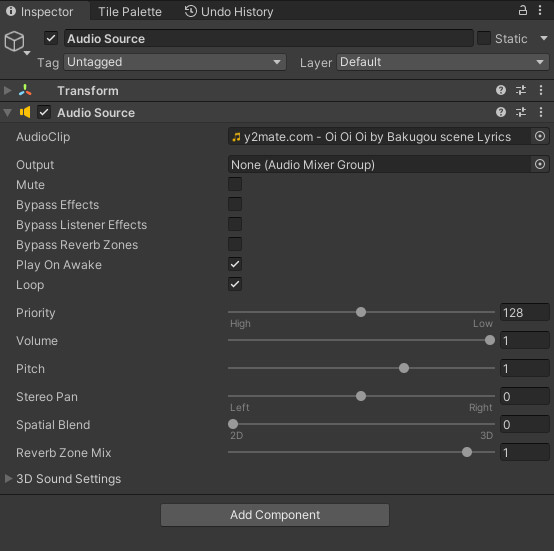
4.Xử lý khi người chơi chết:

* Phương thức PlayerDied() được gọi khi người chơi chết.
* Trong phương thức này, playerAlive được đặt thành false, ngăn không cho triển khai thêm kẻ địch mới.

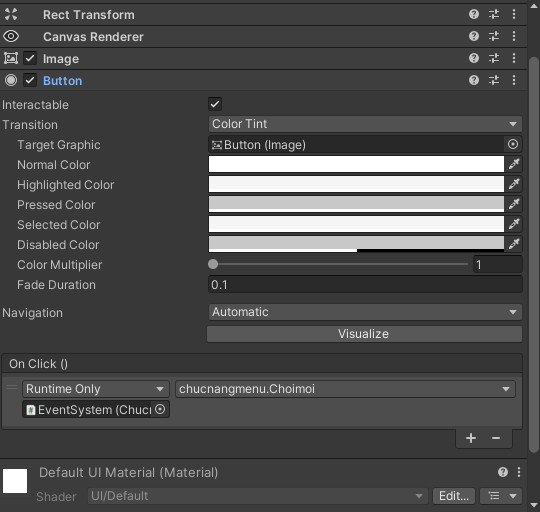
Cách thức hoạt động của script như sau:

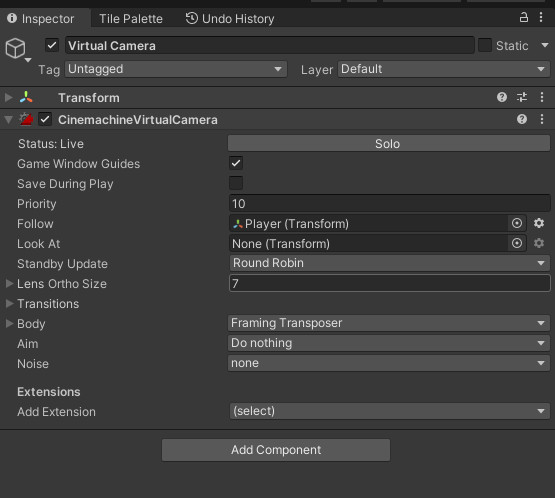
1. Trong Update(), script sẽ kiểm tra xem có thể triển khai kẻ địch mới không (dựa trên các điều kiện như timeBtwSpawn, currentEnemies, maxEnemies và playerAlive).
2. Nếu có thể triển khai, SpawnEnemyNearPlayer() sẽ được gọi để tạo ra một kẻ địch mới, gần vị trí của người chơi.
3. Khi một kẻ địch bị tiêu diệt, EnemyDied() sẽ được gọi, giảm currentEnemies và có thể triển khai thêm kẻ địch mới nếu số lượng hiện tại ít hơn 7.
4. Khi người chơi chết, PlayerDied() sẽ được gọi, đặt playerAlive thành false để ngăn không cho triển khai thêm kẻ địch mới.

Nhạc nền:



Setup nút bấm:



Camera theo người chơi:  


Tài liệu tham khảo:

- Kiến thức lập trình OOP with C# (Đã được học tại trường)

- Phân tích và hướng dẫn tự làm game: Top down shooting

<https://www.youtube.com/watch?v=vjzEiA1bkiw&list=PLHEyg4GEx-GDCpuS3Hm69bB1quBaT8gdS>

**PHÂN CÔNG CÔNG VIỆC CHO CÁC THÀNH VIÊN**

1. Lựa chọn và phân tích đề tài:

- Nguyễn Hoàng Tuấn

- Nguyễn Nhật Hào

2. Ý tưởng về thiết kế giao diện, chức năng:

- Nguyễn Hoàng Tuấn

- Nguyễn Nhật Hào

- Phạm Gia Huy

3. Giải quyết bài toán:

- Nguyễn Hoàng Tuấn

- Nguyễn Nhật Hào

- Phạm Gia Huy

4. Tiến hành code mã nguồn:

- Nguyễn Hoàng Tuấn

5. Tiến hành kiểm thử: Cả nhóm

6. Viết báo cáo và tổng hợp:

- Nguyễn Nhật Hào

- Nguyễn Hoàng Tuấn

- Phạm Gia Huy

# ĐÁNH GIÁ CÁC THÀNH VIÊN (A, B, C, D)

- Nguyễn Hoàng Tuấn:A

- Nguyễn Nhật Hào:B

- Phạm Gia Huy:B

Link tải game:

<https://drive.google.com/file/d/1zJ7s_OYUaT082DDdbA9lYQVYGiJX-96F/view?usp=sharing>

(do dung lượng cao nên không thể up lên github)