**1. Chuẩn bị công cụ và tài nguyên**

1. **Lựa chọn công cụ phát triển (Game Engine)**
   * **Đề xuất:** Unity 2D (miễn phí, dễ tiếp cận, có nhiều tài liệu tiếng Việt).
   * Nếu bạn quen hoặc muốn thử engine khác (GameMaker Studio, Godot, Construct…), cũng có thể áp dụng tương tự về mặt logic.
2. **Cài đặt Unity (nếu chọn Unity)**
   * Tải Unity Hub từ trang chủ [unity3d.com](https://unity3d.com/), rồi cài Unity Editor phiên bản 2021 trở lên (chọn module “2D”).
   * Đảm bảo cài thêm gói “Visual Studio” hoặc “Visual Studio Code” để viết mã C#.
3. **Tạo thư mục dự án và công cụ hỗ trợ**
   * Mở Unity Hub → “New Project” → chọn template “2D” → đặt tên “TinhThanCaGia” → chọn đường dẫn lưu → “Create”.
   * Trong thư mục dự án, chuẩn bị ba thư mục chính trong Assets:
     + **Sprites/** (chứa tất cả hình 2D: sprite lính, ô bản đồ, icon vũ khí, UI…)
     + **Scripts/** (chứa mã C#)
     + **Prefabs/** (đưa các Prefab đã làm sẵn để dùng lại)
     + Ngoài ra có thể thêm: **Audio/** (nhạc, hiệu ứng), **UI/** (hình nút, giao diện chung).
4. **Thu thập tài nguyên đồ họa và âm thanh**
   * **Hình ảnh 2D (Sprites)**:
     + Bạn có thể dùng asset miễn phí trên Asset Store hoặc vẽ đơn giản bằng Photoshop, Aseprite, Inkscape…
     + Cần có sprite:
       - Lính (phải có hình đứng yên, di chuyển, chết – có thể dùng 3–4 frame đơn giản).
       - Ô bản đồ (tilemap) gồm: ô đất trống, ô chướng ngại (tường), ô Supply Point, ô Flag.
       - Icon vũ khí (súng, dao, lựu đạn), ô UI (nền thanh HP, nền menu).
   * **Âm thanh**:
     + Nhạc nền game (tĩnh, không lời, hơi buồn), nhạc giao tranh (tích tắc, căng thẳng) và nhạc kết thúc (theo mức hi sinh).
     + Hiệu ứng: tiếng bước chân, tiếng súng nổ, tiếng lựu đạn nổ, tiếng lính la hét ngắn, tiếng menu click.
   * **Font chữ**: Chọn một font đơn giản, dễ đọc (ví dụ: Arial, Roboto, Noto Sans) cỡ 16–18 khi hiển thị UI.

**2. Thiết kế sơ bộ dữ liệu và kho lưu trữ**

1. **Xác định các lớp (classes) cần có**
   * **Lớp Soldier**:
     + Chứa thông tin cá nhân: string name; int maxHP; int currentHP; int moveRange; int attackPower; int morale;
     + Thêm trường đánh dấu trạng thái: bool isAlive; bool hasMoved; bool hasAttacked;
   * **Lớp Tile** (ô trên bản đồ):
     + Kiểu ô: enum TileType { Empty, Obstacle, SupplyPoint, Flag }
     + Vị trí (x, y), tham chiếu tới sprite.
   * **Lớp GameController**:
     + Quản lý lượt (đếm lượt, chuyển lượt người chơi và AI).
     + Tính toán tỉ lệ hi sinh, kết thúc game.
   * **Lớp UIManager**:
     + Chịu trách nhiệm hiển thị UI: màn hình nhập lý tưởng, menu chọn tiểu đội, thanh HP, thông báo kết thúc.
   * **Lớp AIController** (phiên bản đơn giản):
     + Định nghĩa hành vi di chuyển và tấn công cơ bản khi tới lượt AI.
2. **Tạo thư mục Scripts và khởi tạo file cs**
   * Trong Assets/Scripts/, tạo các file:
     + Soldier.cs
     + Tile.cs
     + GameController.cs
     + UIManager.cs
     + AIController.cs
   * Mỗi file gồm khung sẵn:

csharp

Sao chépChỉnh sửa

using UnityEngine;

public class Soldier : MonoBehaviour

{

// Thông tin cá nhân

public string soldierName;

public int maxHP;

public int currentHP;

public int moveRange;

public int attackPower;

public int morale;

public bool isAlive = true;

public bool hasMoved = false;

public bool hasAttacked = false;

// Các hàm: Move(), Attack(), Die()...

}

1. **Chuẩn bị dữ liệu khởi tạo**
   * Trong GameController, định nghĩa trước 8 lính mẫu (tên và thông số cơ bản), ví dụ:

csharp

Sao chépChỉnh sửa

void Start()

{

Soldier[] squad = new Soldier[8];

squad[0] = new Soldier("Hòa", 100, 2, 30, 100);

// ... tạo tiếp 7 lính còn lại

}

* + Tương tự, khởi tạo tilemap (bản đồ) bằng cách lưu trước mảng 2D Tile[,] mapTiles = new Tile[12,10]; và gán loại ô.

**3. Xây dựng bản đồ chiến trường (Tilemap)**

1. **Tạo Tilemap trong Unity**
   * Mở Scene mặc định → Tạo GameObject “Grid” (2D Object → Tilemap)
   * Nhóm Grid chứa một Tilemap Renderer và Tilemap.
   * Chuẩn bị các sprite ô (spr\_empty, spr\_obstacle, spr\_supply, spr\_flag) và tạo thành Tiles (Assets → Create → Tile → kéo sprite vào).
   * Vẽ sơ bộ bản đồ (12×10 ô):
     + Đặt chướng ngại (tường) tại một số ô cố định.
     + Đặt 2 SupplyPoint ở 2 vị trí xa nhau.
     + Đặt Flag (mục tiêu chính) ở giữa hoặc các góc tùy thiết kế.
2. **Gắn component Tile.cs lên mỗi ô**
   * Tạo Prefab “TilePrefab” gồm:
     + Một GameObject có SpriteRenderer (hiển thị tile), BoxCollider2D (để phát hiện click/hover)
     + Thêm script Tile.cs để lưu thuộc tính:

csharp

Sao chépChỉnh sửa

public class Tile : MonoBehaviour

{

public int x, y;

public TileType tileType;

public bool isOccupied = false; // nếu có lính đứng trên

// Các hàm: OnMouseDown() để người chơi chọn tile di chuyển/att

}

* + Trên Tilemap, bạn có thể tạo ô động bằng cách instantiate nhiều TilePrefab, gán vị trí (x,y) tương ứng. Hoặc dùng hệ Grid Builder kết hợp với Tilemap Collider (tuy nhiên script sẽ phải tìm “Tile” tại tọa độ).

**4. Tạo và quản lý lính (Soldier)**

1. **Thiết kế Prefab cho lính**
   * Tạo một GameObject trống “SoldierPrefab”, thêm SpriteRenderer (kéo sprite nhân vật vào), BoxCollider2D (để nhận input click), Rigidbody2D (set Body Type = Kinematic).
   * Thêm component Soldier.cs lên SoldierPrefab.
   * Trong Soldier.cs, khai báo các biến public để dễ gán trong Inspector:

csharp

Sao chépChỉnh sửa

public string soldierName;

public int maxHP;

public int currentHP;

public int moveRange;

public int attackPower;

public int morale;

public bool isAlive = true;

public bool hasMoved = false;

public bool hasAttacked = false;

* + Tạo Prefab (kéo SoldierPrefab vào thư mục Prefabs).

1. **Khởi tạo tiểu đội**
   * Trong GameController, viết hàm SpawnSquad() để instantiate 6–8 SoldierPrefab lên ô hoặc gần khu Base người chơi.

csharp

Sao chépChỉnh sửa

void SpawnSquad()

{

Vector2[] teamPositions = { new Vector2(0,0), new Vector2(1,0), ... };

for (int i = 0; i < squad.Length; i++)

{

GameObject s = Instantiate(soldierPrefab, teamPositions[i], Quaternion.identity);

Soldier soldierComp = s.GetComponent<Soldier>();

soldierComp.soldierName = "Hòa"; // hoặc tên random từ mảng

soldierComp.maxHP = 100; // gán giá trị ban đầu

soldierComp.currentHP = 100;

soldierComp.moveRange = 2;

soldierComp.attackPower = 30;

soldierComp.morale = 100;

squad[i] = soldierComp;

}

}

* + Tương tự, spawn đội địch (AI) ở base đối lập.

1. **Quản lý lượt di chuyển và hành động**
   * Trong GameController, quản lý biến bool isPlayerTurn, int currentTurn, int turnLimit = 15.
   * Khi lượt người chơi:
     + Cho phép click vào Soldier của mình để chọn (đánh dấu), sau đó click vào ô trên Tilemap để di chuyển (check số ô ≤ moveRange, tile không bị chướng ngại, không có lính khác).
     + Khi di chuyển, giảm squad[i].moveRange-- (hoặc set hasMoved = true).
     + Nếu ở tầm đánh (adjacent) đối phương, hiển thị nút “Attack”, click thì trừ HP địch và set hasAttacked = true. Sau đó có thể mất morale nếu kills hay không joins.
   * Khi hết lượt (click nút “End Turn”):
     + Chuyển isPlayerTurn = false, gọi AIController chạy lượt AI tương tự (đơn giản: mọi lính AI di chuyển về hướng gần nhất với lính người chơi, tấn công khi có thể).
     + Sau khi AI xong, tăng currentTurn++, đặt isPlayerTurn = true và reset thuộc tính hasMoved = false, hasAttacked = false cho tất cả lính người chơi.
2. **Tính toán thiệt hại và morale**
   * Khi một Soldier bị trừ HP → if currentHP ≤ 0 → gọi Die():

csharp

Sao chépChỉnh sửa

public void Die()

{

isAlive = false;

// Hiệu ứng – ví dụ instantiate particle chết

Destroy(this.gameObject); // hoặc disable sau 1s

GameController.Instance.OnSoldierDeath(this);

}

* + Trong GameController.OnSoldierDeath(Soldier s):
    - Giảm morale của toàn đội: ví dụ foreach (var member in squad) member.morale -= 10;
    - Hiện thông báo nhỏ: “Hòa hy sinh để trấn giữ lối vào.”
    - Cập nhật biến int totalDeaths++; để tính tỉ lệ hi sinh.

**5. Xây dựng UI cơ bản**

1. **Màn hình nhập lý tưởng và chọn tiểu đội**
   * Tạo Canvas mới (GameObject → UI → Canvas), đặt tên “UIManager”.
   * Trong Canvas, thêm Panel “InputIdealsPanel”:
     + Hai ô InputField (UI → Input Field) để nhập “Lý tưởng tin” và “Lý tưởng ghét”.
     + Nút “Continue” (UI → Button) để chuyển sang panel chọn đội.
   * Tạo Panel “SelectSquadPanel”:
     + Hiển thị danh sách lính (8 khung nhỏ, mỗi khung hiển thị ảnh nhỏ và tên do game gán).
     + Cho phép click vào một khung để “bỏ chọn” (nếu không muốn dùng người đó).
     + Nút “Play” để bắt đầu vào màn chiến trường.
   * Khi người chơi click “Play”, khởi tạo GameController.SpawnSquad() và GameController.SpawnEnemy() rồi tắt panel này, hiển thị “BattlePanel”.
2. **Màn hình BattlePanel**
   * Canvas chính chứa:
     + **Panel Left** (cột trái, rộng 1/4 màn hình):
       - Danh sách lính (tên, HP, morale) – hiển thị dạng Text dưới nhau. Có thể highlight khi click chọn.
     + **Panel Right** (cột phải, rộng 1/4):
       - Thông báo sự kiện (“Supply Point chiếm được!”, “Lính còn 3 HP”, “Turn 5/15”).
     + **Middle (12×10 ô Tilemap)**:
       - Tilemap + Danh sách GameObject Soldier.
     + **Panel Bottom**:
       - Nút “End Turn”, nút “Restart” (ẩn đến khi kết thúc), thanh hiển thị tỉ lệ hi sinh (Text: “Tỉ lệ hi sinh: 0%”).
   * Tạo Prefab “HPBar” (UI Image) gắn vào soldier để nhìn thấy HP trên đầu – tùy chọn, có thể làm sau.
3. **Màn hình kết quả (ResultPanel)**
   * Tạo Panel “ResultPanel” (ẩn ban đầu).
   * Bên trong:
     + Text lớn hiển thị “Kết thúc: [Tên kết thúc]”.
     + Bảng thống kê: dùng UI Text hiển thị từng lính: tên – tình trạng (Alive/Dead).
     + Text hiển thị “Tỉ lệ hi sinh: X%”.
     + Text hiển thị mô tả cảm xúc tương ứng (VD “Cái giá của niềm tin…”).
     + Hai nút “Chơi lại” (Restart) và “Thoát” (Quit ứng với Application.Quit()).

**6. Lập trình tính toán kết thúc**

1. **Theo dõi số hi sinh**
   * Trong GameController, khai báo biến:

csharp

Sao chépChỉnh sửa

public int totalSoldiers; // số lính ban đầu

public int totalDeaths; // số lính chết

* + Khi spawn, totalSoldiers = squad.Length; totalDeaths = 0;
  + Trong OnSoldierDeath(...), tăng totalDeaths++;, cập nhật UI Bottom Panel:

csharp

Sao chépChỉnh sửa

float deathRate = (float)totalDeaths / totalSoldiers \* 100f;

bottomPanel.DeathRateText.text = "Tỉ lệ hi sinh: " + deathRate.ToString("F0") + "%";

1. **Kiểm tra điều kiện kết thúc**
   * Trước khi bắt đầu mỗi lượt, hoặc sau khi lượt AI xong, gọi hàm CheckEndGame():

csharp

Sao chépChỉnh sửa

void CheckEndGame()

{

// Nếu toàn bộ lính đội bạn chết → hi sinh 100% → gọi EndGame()

bool anyAlive = squad.Any(s => s.isAlive);

if (!anyAlive)

{

EndGame();

return;

}

// Nếu Flag được chiếm (target) hoặc lượt > turnLimit → EndGame()

if (flagCaptured || currentTurn > turnLimit)

{

EndGame();

}

}

* + Trong EndGame():

csharp

Sao chépChỉnh sửa

void EndGame()

{

float deathRate = (float)totalDeaths / totalSoldiers \* 100f;

string endingName;

string endingMessage;

if (deathRate <= 10f)

{

endingName = "Tia sáng cuối chân trời";

endingMessage = "Hòa bình thắng lợi với tổn thất tối thiểu…";

}

else if (deathRate <= 40f)

{

endingName = "Cái giá của niềm tin";

endingMessage = "Chiến thắng nhưng day dứt…";

}

else if (deathRate <= 70f)

{

endingName = "Chiến thắng rỗng tuếch";

endingMessage = "Hầu hết đều mất mạng, chiến thắng không còn ý nghĩa…";

}

else

{

endingName = "Nỗi nhớ và máu";

endingMessage = "Thảm họa: Không ai trở về…";

}

ShowResultPanel(endingName, endingMessage, deathRate);

}

* + ShowResultPanel(...) là hàm hiển thị “ResultPanel” với các thông số tương ứng (đã lập UI sẵn).

**7. Lập trình AI cơ bản**

1. **Khởi tạo đội AI**
   * Giống SpawnSquad(), nhưng lính AI có thể có thông số khác (sát thương cao hơn, HP thấp hơn).
   * Tạo array enemySquad chứa Soldier của đối phương.
2. **Hành vi AI đơn giản**
   * Trong AIController, viết hàm AIPlayTurn():

csharp

Sao chépChỉnh sửa

public void AIPlayTurn(Soldier[] enemyTeam, Soldier[] playerTeam)

{

foreach (var e in enemyTeam)

{

if (!e.isAlive) continue;

// Tìm lính player còn sống gần nhất

Soldier target = FindNearestAliveSoldier(e, playerTeam);

if (target == null) break;

// Tính khoảng cách Manhattan

int dx = Mathf.Abs(e.gridX - target.gridX);

int dy = Mathf.Abs(e.gridY - target.gridY);

if (dx + dy <= 1)

{

// Tấn công

e.Attack(target);

}

else

{

// Dịch chuyển từng ô về hướng target (ví dụ: di chuyển 1 ô x hướng x, rồi y)

Vector2 nextPos = GetStepTowards(e.gridX, e.gridY, target.gridX, target.gridY);

if (CanMoveTo(nextPos))

e.Move(nextPos);

}

}

}

* + Các hàm phụ: FindNearestAliveSoldier(), GetStepTowards(), CanMoveTo() đều tính toán đơn giản trên mảng mapTiles và danh sách playerTeam.

1. **Chuyển lượt**
   * Khi người chơi click “End Turn” → GameController.Instance.AIController.AIPlayTurn(enemySquad, squad); → sau khi AI kết thúc, gọi CheckEndGame() lại → hiển thị phép tính tỉ lệ hi sinh…

**8. Hoàn thiện giao diện và trải nghiệm**

1. **Hiệu ứng hình ảnh**
   * Khi lính di chuyển: có thể thêm anmation 2–3 frame, hoặc dùng tween để chuyển mượt.
   * Khi lính chết: instantiate một particle effect đơn giản (một vệt khói hoặc màu tối).
   * Khi kết thúc lượt: màn hình hơi chuyển màu xám trong 0.3s, rồi chuyển sang lượt kế.
2. **Âm thanh**
   * Trong Unity, import file mp3/wav vào thư mục Assets/Audio/.
   * Tạo một GameObject “AudioManager” với component AudioSource.
   * Viết script AudioManager để:

csharp

Sao chépChỉnh sửa

public void PlayBGM(string bgmName) { /\* Play nhạc nền \*/ }

public void PlaySFX(string sfxName) { /\* Play hiệu ứng \*/ }

* + Khi bắt đầu trận, AudioManager.PlayBGM("battle\_theme");
  + Khi lính chết, AudioManager.PlaySFX("death\_sound");
  + Khi kết thúc, dừng BGM, play SFX tùy kết thúc.

1. **Thông báo (Notifications) nhỏ**
   * Khi sự kiện Supply Point bị chiếm: hiển thị thông báo pop-up nhỏ góc phải “Supply Point chiếm được, tiếp tế thêm 5 đạn!” rồi ẩn sau 2 giây.
   * Khi lính sắp chết (HP < 30%), đổi màu tên trong UI thành đỏ nhạt.

**9. Thử nghiệm, sửa lỗi, cân bằng**

1. **Kiểm thử (Testing)**
   * Chạy thử nhiều tình huống:
     + Thắng sớm (đạt Flag)
     + Kéo dài 15 lượt, test cơ chế hi sinh (lính trừ HP dần)
     + Test AI có thể bị kẹt hoặc đi vòng.
2. **Cân bằng thông số**
   * Điều chỉnh moveRange, attackPower, maxHP phù hợp để AI không quá mạnh hoặc quá yếu.
   * Thay đổi số đạn ban đầu, số lượt, số Supply Point để tạo độ khó tương đối.
3. **Phản hồi người chơi nội bộ**
   * Mời vài người chơi thử, hỏi họ cảm nhận về độ khó, tính ám ảnh, kết thúc có đủ mạnh không.
   * Dựa vào feedback, điều chỉnh tỉ lệ hi sinh – kết thúc, thông báo, nhạc nền.

**10. Đóng gói và xuất bản**

1. **Build game**
   * Trong Unity: File → Build Settings
   * Chọn platform (PC/Windows), chọn “Development Build” để test. Nhấn “Build” và chọn thư mục xuất.
   * Chạy file .exe để kiểm tra xem game hoạt động đúng không.
2. **Tối ưu nhẹ**
   * Xoá asset không dùng đến, giảm size texture nếu cần.
   * Bật “Strip Engine Code” hoặc “Managed Stripping Level” (trong Player Settings) để giảm kích thước build.
3. **Tài liệu hướng dẫn ngắn**
   * Tạo file README.txt hoặc README.pdf đơn giản, hướng dẫn cách chạy:

markdown

Sao chépChỉnh sửa

1. Giải nén thư mục game.

2. Mở file TinhThanCaGia.exe.

3. Nhập lý tưởng, chọn lính, nhấn Play.

4. Sử dụng phím mũi tên để di chuyển, phím Space để tấn công.

5. Click “End Turn” để chuyển lượt.

6. Chọn “Restart” hoặc “Quit” khi kết thúc.

1. **Đóng gói nộp thi**
   * Nén toàn bộ thư mục containing game.exe + Data (kèm thư mục AssetsBundle nếu có).
   * Đính kèm tài liệu ý tưởng (bản PDF đã xây dựng) để ban giám khảo hiểu rõ thiết kế.

**11. Tóm tắt quy trình ngắn gọn**

1. **Chuẩn bị**:
   * Cài Unity, chuẩn bị sprite và âm thanh.
   * Tạo project 2D, cấu trúc thư mục (Sprites, Scripts, Prefabs, Audio, UI).
2. **Thiết kế dữ liệu**:
   * Khởi tạo lớp Soldier, Tile, GameController, UIManager, AIController.
3. **Xây bản đồ**:
   * Tạo Tilemap 12×10, thiết lập ô trống, chướng ngại, Supply Point, Flag.
4. **Tạo lính (Prefabs)**:
   * Quần lính 6–8 người, gán tên, HP, sát thương, di chuyển.
5. **Lập trình lượt**:
   * Người chơi di chuyển, tấn công, sau đó tới lượt AI.
6. **Hệ thống thiệt hại & morale**:
   * Khi lính chết, cập nhật tỉ lệ hi sinh, morale giảm.
7. **UI/UX**:
   * Màn nhập lý tưởng, chọn tiểu đội, màn chiến trường, màn kết thúc.
8. **Hệ thống kết thúc đa dạng**:
   * Tính tỉ lệ hi sinh, quyết định 4 kết thúc, hiển thị thông điệp.
9. **Cải tiến**:
   * Thêm sự kiện ngẫu nhiên, nâng cấp vũ khí, achievement, phân tích replay.
10. **Thử nghiệm & đóng gói**:
    * Test, cân bằng, build, nén nộp thi + PDF ý tưởng.

**Lời khuyên cuối cùng**

* **Bắt đầu từng bước**: Tạo trước khung di chuyển (Soldier Move → Blade), sau đó tích hợp tấn công, rồi morale, cuối cùng mới làm UI kết quả.
* **Sử dụng placeholder**: Ban đầu vẽ ô, nhân vật rất đơn giản (hình vuông, hình tròn). Khi logic đã ổn, mới thay bằng sprite hoàn chỉnh.
* **Chia nhỏ công việc**: Mỗi tính năng (di chuyển, tấn công, morale, UI) nên làm trong một nhánh nhỏ rồi gộp dần, tránh code chồng chéo khó sửa.