**HỆ THỐNG ĐIỂM & PHẦN THƯỞNG: ANIMAL RESCUE ADVENTURE**

**TỔNG QUAN HỆ THỐNG**

Hệ thống điểm thưởng được thiết kế để:

1. **Tạo động lực** cho người chơi tiếp tục chơi và khám phá
2. **Khuyến khích học tập** thông qua việc thưởng cho kiến thức về động vật và môi trường
3. **Cân bằng giữa thử thách và thành tựu** cho các nhóm tuổi khác nhau
4. **Tạo cảm giác tiến bộ** rõ ràng qua hệ thống cấp độ Ranger

**CÁC LOẠI ĐIỂM & TIỀN TỆ**

**1. Điểm Kinh nghiệm (XP)**

* **Mục đích**: Tăng cấp độ Ranger
* **Cách kiếm**:
  + Hoàn thành chạy (10 XP mỗi 100m)
  + Giải cứu động vật (50-250 XP dựa vào độ hiếm)
  + Hoàn thành nhiệm vụ hàng ngày (100-300 XP)
  + Mở khóa kiến thức mới về động vật (25 XP mỗi mục)
  + Xây dựng khu bảo tồn (50 XP mỗi nâng cấp)
* **Thang tiến XP**:
* Cấp 1: 0 XPCấp 2: 500 XPCấp 3: 1,000 XPCấp 4: 2,000 XPCấp 5: 3,500 XP...Cấp N: Cấp N-1 + (N-1)\*500 XP

**2. Xu (Coins)**

* **Mục đích**: Tiền tệ chính để mở khóa và nâng cấp
* **Cách kiếm**:
  + Thu thập trong chạy (1 xu mỗi đồng xu)
  + Hoàn thành mini-game (10-50 xu dựa vào độ khó và thời gian)
  + Hoàn thành nhiệm vụ hàng ngày (20-100 xu)
  + Thưởng hàng ngày khi đăng nhập (20 xu, tăng theo chuỗi ngày đăng nhập)
  + Lên cấp (50 xu \* cấp độ mới)
* **Cách sử dụng**:
  + Mua vật phẩm trang trí cho khu bảo tồn
  + Mở khóa trang phục cho nhân vật
  + Mua power-up và vật phẩm tiêu hao
  + Nâng cấp khu vực trong khu bảo tồn

**3. Ngôi Sao Bảo Tồn (Conservation Stars)**

* **Mục đích**: Tiền tệ cao cấp cho vật phẩm đặc biệt
* **Cách kiếm**:
  + Giải cứu động vật quý hiếm (1-3 sao dựa vào độ hiếm)
  + Hoàn thành thử thách hàng tuần (2-5 sao)
  + Hoàn thành bộ sưu tập động vật theo biome (5 sao)
  + Đạt mốc cấp độ quan trọng (Cấp 10, 25, 50, v.v. - 3-10 sao)
  + Thành tích đặc biệt (5-15 sao)
* **Cách sử dụng**:
  + Mở khóa biome mới sớm
  + Mở khóa các loài động vật đặc biệt
  + Mua vật phẩm trang trí cao cấp
  + Mở khóa tính năng đặc biệt trong khu bảo tồn

**4. Điểm Bảo Tồn (Mỗi loài động vật)**

* **Mục đích**: Theo dõi mức độ chăm sóc đối với từng loài
* **Cách kiếm**:
  + Cho động vật ăn (+1 điểm)
  + Tương tác với động vật trong khu bảo tồn (+1 điểm)
  + Nâng cấp môi trường sống (+3 điểm)
  + Học về động vật và hoàn thành câu đố liên quan (+2 điểm)
* **Lợi ích**:
  + Mở khóa hành vi mới của động vật (10/25/50 điểm)
  + Tăng tỷ lệ xuất hiện các loài liên quan
  + Mở khóa nâng cấp đặc biệt trong khu bảo tồn

**HỆ THỐNG ĐIỂM SỐ TRONG ENDLESS RUNNER**

**1. Điểm cơ bản**

* **Quãng đường**: 1 điểm mỗi mét
* **Thu thập xu**: 5 điểm mỗi xu
* **Thu thập ngọc**: 25 điểm mỗi ngọc

**2. Điểm cộng thêm**

* **Combo thu thập**: Tăng 10% điểm sau mỗi 5 vật phẩm thu thập liên tiếp
* **Perfect Timing**: Nhảy/trượt đúng lúc +10 điểm
* **Near Miss**: Tránh chướng ngại vật trong gang tấc +15 điểm
* **Lane Change Bonus**: Đổi làn để tránh chướng ngại vật +20 điểm

**3. Hệ số nhân**

* **Streak không va chạm**:
  + 100m: x1.1
  + 300m: x1.2
  + 500m: x1.5
  + 1000m: x2.0
* **Power-up nhân điểm**: x1.5 - x3.0 trong thời gian giới hạn
* **Tăng theo độ khó**: Mỗi biome khó hơn có hệ số điểm cao hơn
  + Rừng (Jungle): x1.0 (cơ bản)
  + Đại dương (Ocean): x1.2
  + Savanna: x1.5
  + Vùng cực (Arctic): x1.8
  + Núi cao (Mountain): x2.0

**4. Điểm thưởng cho giải câu đố**

* **Hoàn thành cơ bản**: 100 điểm
* **Hoàn thành nhanh**: Tối đa 200 điểm thưởng thêm (giảm dần theo thời gian)
* **Độ chính xác**: +10 điểm cho mỗi câu trả lời/hành động đúng
* **Thưởng liên tục**: Chuỗi các câu đố liên tiếp hoàn thành tăng điểm 10% mỗi lần

**HỆ THỐNG BỘ CHỌN MỨC THƯỞNG (REWARD POOL SELECTOR)**

// RewardManager.cs (một phần)

public class RewardManager : MonoBehaviour

{

[System.Serializable]

public class RewardPool

{

public string poolName;

public RewardTier tier;

public List<RewardItem> possibleRewards;

public float dropChance; // Tỷ lệ rơi (0-100%)

}

[System.Serializable]

public class RewardItem

{

public string itemId;

public RewardType type;

public int minAmount;

public int maxAmount;

public float weight; // Trọng số trong pool (càng cao càng có khả năng rơi)

}

public enum RewardTier

{

Common,

Uncommon,

Rare,

Epic,

Legendary

}

public enum RewardType

{

Coins,

XP,

ConservationStars,

PowerUp,

AnimalUnlock,

Cosmetic

}

[SerializeField] private List<RewardPool> rewardPools;

// Chọn pool phần thưởng dựa vào thành tích người chơi và yếu tố ngẫu nhiên

public RewardPool SelectRewardPool(float score, int animalRarity)

{

// Tính toán tier cơ bản dựa vào điểm số

RewardTier baseTier = CalculateBaseTier(score);

// Điều chỉnh dựa vào độ hiếm của động vật

RewardTier adjustedTier = AdjustTierByRarity(baseTier, animalRarity);

// Lọc các pool phù hợp với tier

List<RewardPool> eligiblePools = rewardPools

.Where(pool => pool.tier == adjustedTier)

.ToList();

if (eligiblePools.Count == 0)

return null;

// Chọn ngẫu nhiên từ các pool phù hợp, có xét đến tỷ lệ rơi

float totalChance = eligiblePools.Sum(pool => pool.dropChance);

float randomValue = Random.Range(0, totalChance);

float accumulatedChance = 0;

foreach (RewardPool pool in eligiblePools)

{

accumulatedChance += pool.dropChance;

if (randomValue <= accumulatedChance)

{

return pool;

}

}

// Mặc định trả về pool đầu tiên nếu có lỗi

return eligiblePools[0];

}

// Chọn phần thưởng cụ thể từ pool

public List<RewardItem> SelectRewardsFromPool(RewardPool pool, int numRewards = 1)

{

List<RewardItem> selectedRewards = new List<RewardItem>();

List<RewardItem> availableRewards = new List<RewardItem>(pool.possibleRewards);

for (int i = 0; i < numRewards; i++)

{

if (availableRewards.Count == 0)

break;

// Tính tổng trọng số

float totalWeight = availableRewards.Sum(item => item.weight);

float randomWeight = Random.Range(0, totalWeight);

// Chọn item dựa trên trọng số

float accumulatedWeight = 0;

for (int j = 0; j < availableRewards.Count; j++)

{

accumulatedWeight += availableRewards[j].weight;

if (randomWeight <= accumulatedWeight)

{

RewardItem selectedItem = availableRewards[j];

// Xác định số lượng ngẫu nhiên trong khoảng min-max

int amount = Random.Range(selectedItem.minAmount, selectedItem.maxAmount + 1);

// Tạo bản sao để thiết lập số lượng

RewardItem rewardCopy = new RewardItem

{

itemId = selectedItem.itemId,

type = selectedItem.type,

minAmount = amount,

maxAmount = amount,

weight = selectedItem.weight

};

selectedRewards.Add(rewardCopy);

// Xóa phần thưởng đã chọn (để tránh lặp lại)

availableRewards.RemoveAt(j);

break;

}

}

}

return selectedRewards;

}

private RewardTier CalculateBaseTier(float score)

{

// Ví dụ đơn giản về cách xác định tier dựa vào điểm số

if (score < 1000)

return RewardTier.Common;

else if (score < 2500)

return RewardTier.Uncommon;

else if (score < 5000)

return RewardTier.Rare;

else if (score < 10000)

return RewardTier.Epic;

else

return RewardTier.Legendary;

}

private RewardTier AdjustTierByRarity(RewardTier baseTier, int animalRarity)

{

// Tăng tier dựa vào độ hiếm của động vật

int tierAdjustment = animalRarity - 1; // animalRarity từ 1-5

int tierIndex = (int)baseTier + tierAdjustment;

tierIndex = Mathf.Clamp(tierIndex, 0, 4); // Giới hạn trong khoảng 0-4 (Common đến Legendary)

return (RewardTier)tierIndex;

}

// Áp dụng phần thưởng cho người chơi

public void ApplyRewards(List<RewardItem> rewards)

{

foreach (RewardItem reward in rewards)

{

switch (reward.type)

{

case RewardType.Coins:

DataManager.Instance.AddCoins(reward.minAmount);

break;

case RewardType.XP:

DataManager.Instance.AddXP(reward.minAmount);

break;

case RewardType.ConservationStars:

DataManager.Instance.AddConservationStars(reward.minAmount);

break;

case RewardType.PowerUp:

DataManager.Instance.AddPowerUp(reward.itemId, reward.minAmount);

break;

case RewardType.AnimalUnlock:

DataManager.Instance.UnlockAnimal(reward.itemId);

break;

case RewardType.Cosmetic:

DataManager.Instance.UnlockCosmetic(reward.itemId);

break;

}

}

}

}

**LƯU TRỮ VÀ HIỂN THỊ THÀNH TÍCH**

**1. Bảng xếp hạng**

* **Bảng xếp hạng cá nhân**: Lưu top 10 thành tích cao nhất của người chơi
* **Bảng xếp hạng bạn bè**: Tùy chọn kết nối xã hội nếu triển khai
* **Bảng xếp hạng theo độ tuổi**: Người chơi chỉ tranh đua với nhóm tuổi tương tự
* **Thống kê theo biome**: Điểm cao nhất trên từng môi trường

**2. Hệ thống thành tựu**

* **Thành tựu chung**: Đạt mốc về điểm, khoảng cách, động vật giải cứu
* **Thành tựu biome**: Hoàn thành các mục tiêu trong từng môi trường cụ thể
* **Thành tựu chuyên gia**: Cứu đủ bộ động vật trong một nhóm
* **Thành tựu hành trình**: Hoàn thành các nhiệm vụ theo trình tự

**3. Lưu trữ tiến trình**

* **Định dạng lưu**: JSON
* **Thông tin lưu**:
* { "player": { "name": "PlayerName", "level": 10, "xp": 5000, "coins": 2500, "stars": 15, "ageGroup": "8-11" }, "progress": { "highestScore": 12500, "totalDistance": 25000, "animalsRescued": 35, "biomeProgress": { "jungle": 0.8, "ocean": 0.4, "savanna": 0.1, "arctic": 0.0, "mountain": 0.0 } }, "inventory": { "powerups": [ {"id": "shield", "amount": 5}, {"id": "magnet", "amount": 3} ], "cosmetics": ["ranger\_hat", "explorer\_outfit"] }, "sanctuary": { "unlocked": true, "upgrades": { "jungle": 2, "ocean": 1 }, "animals": [ {"id": "monkey", "conservationPoints": 15}, {"id": "parrot", "conservationPoints": 8} ] }, "achievements": [ {"id": "first\_rescue", "completed": true}, {"id": "jungle\_master", "completed": false, "progress": 0.7} ]}

**HỆ THỐNG NHIỆM VỤ & THƯỞNG**

**1. Nhiệm vụ hàng ngày (3 mỗi ngày)**

* **Loại nhiệm vụ**:
  + Chạy quãng đường nhất định
  + Giải cứu X động vật
  + Thu thập Y xu
  + Hoàn thành Z mini-game
  + Tương tác với T động vật trong khu bảo tồn
* **Phần thưởng**:
  + 50-200 XP
  + 20-100 Xu
  + 0-1 Ngôi sao bảo tồn

**2. Nhiệm vụ hàng tuần (1 mỗi tuần)**

* **Loại nhiệm vụ**:
  + Giải cứu nhóm động vật cụ thể
  + Đạt điểm số nhất định trên biome cụ thể
  + Hoàn thành chuỗi mini-game không bị lỗi
  + Nâng cấp khu vực trong khu bảo tồn
* **Phần thưởng**:
  + 500-1000 XP
  + 200-500 Xu
  + 2-5 Ngôi sao bảo tồn

**3. Nhiệm vụ hành trình (Sự kiện dài hạn)**

* **Cấu trúc**: Chuỗi 5-10 nhiệm vụ liên quan
* **Chủ đề**: Bảo vệ loài cụ thể, phục hồi hệ sinh thái, etc.
* **Phần thưởng tích lũy**: Mỗi bước hoàn thành tăng giá trị phần thưởng
* **Phần thưởng cuối cùng**: Động vật hoặc tính năng đặc biệt, trang phục độc quyền

**POWER-UPS & VẬT PHẨM TIÊU HAO**

**1. Power-ups trong Runner**

* **Loại**:
  + **Lá chắn bảo vệ**: Bất tử trong 10 giây
  + **Nam châm xu**: Thu hút xu trong 15 giây
  + **Tăng tốc**: Tăng tốc độ và điểm 2x trong 10 giây
  + **Jetpack**: Bay qua chướng ngại vật trong 5 giây
  + **Bong bóng bảo vệ**: Cho phép va chạm 1 lần không bị ngã
* **Thu thập**:
  + Xuất hiện ngẫu nhiên trong game
  + Mua bằng xu
  + Phần thưởng từ nhiệm vụ

**2. Vật phẩm hỗ trợ trong Puzzle**

* **Gợi ý**: Hiển thị phần giải pháp
* **Thêm thời gian**: +15 giây
* **Bỏ qua**: Bỏ qua puzzle hiện tại
* **Đơn giản hóa**: Giảm độ khó của puzzle

**3. Vật phẩm khu bảo tồn**

* **Thức ăn đặc biệt**: Tăng điểm bảo tồn nhanh
* **Đồ chơi thú cưng**: Tăng hạnh phúc của động vật
* **Dụng cụ trang trí**: Nâng cấp thẩm mỹ và chức năng

**CODE MẪU CHO TÍNH NĂNG CHÍNH**

**1. Cập nhật điểm số trong Runner**

// ScoreManager.cs (một phần)

public class ScoreManager : MonoBehaviour

{

[SerializeField] private TextMeshProUGUI scoreText;

[SerializeField] private TextMeshProUGUI distanceText;

[SerializeField] private TextMeshProUGUI coinText;

private float currentScore = 0f;

private float distance = 0f;

private int coins = 0;

private int combo = 0;

private float comboMultiplier = 1.0f;

private float biomeMultiplier = 1.0f;

private float streakMultiplier = 1.0f;

private float powerupMultiplier = 1.0f;

private Transform playerTransform;

private Vector3 lastPosition;

private void Start()

{

playerTransform = GameObject.FindGameObjectWithTag("Player").transform;

lastPosition = playerTransform.position;

// Đặt hệ số nhân biome từ GameManager hoặc DataManager

biomeMultiplier = GameManager.Instance.GetCurrentBiomeMultiplier();

UpdateUI();

}

private void Update()

{

// Cập nhật khoảng cách

float distanceMoved = playerTransform.position.z - lastPosition.z;

if (distanceMoved > 0)

{

distance += distanceMoved;

// Cộng điểm cho khoảng cách

float distancePoints = distanceMoved \* 1.0f; // 1 điểm / mét

AddScore(distancePoints);

// Kiểm tra streak thưởng

CheckDistanceStreaks();

}

lastPosition = playerTransform.position;

UpdateUI();

}

public void AddCoins(int amount)

{

coins += amount;

// Cộng điểm cho coin (5 điểm / coin)

float coinPoints = amount \* 5.0f;

AddScore(coinPoints);

// Tăng combo

combo++;

if (combo >= 5)

{

// Mỗi 5 combo tăng 10% hệ số

comboMultiplier = 1.0f + (combo / 5) \* 0.1f;

}

UpdateUI();

}

public void ResetCombo()

{

combo = 0;

comboMultiplier = 1.0f;

}

public void AddPerfectTimingBonus()

{

AddScore(10);

}

public void AddNearMissBonus()

{

AddScore(15);

}

public void AddLaneChangeBonus()

{

AddScore(20);

}

public void SetPowerupMultiplier(float multiplier, float duration)

{

StartCoroutine(ApplyPowerupMultiplier(multiplier, duration));

}

private IEnumerator ApplyPowerupMultiplier(float multiplier, float duration)

{

float originalMultiplier = powerupMultiplier;

powerupMultiplier = multiplier;

yield return new WaitForSeconds(duration);

powerupMultiplier = originalMultiplier;

}

private void AddScore(float points)

{

// Áp dụng tất cả hệ số nhân

float totalMultiplier = comboMultiplier \* biomeMultiplier \* streakMultiplier \* powerupMultiplier;

float adjustedPoints = points \* totalMultiplier;

currentScore += adjustedPoints;

}

private void CheckDistanceStreaks()

{

// Thiết lập hệ số nhân streak dựa vào khoảng cách

if (distance >= 1000)

streakMultiplier = 2.0f;

else if (distance >= 500)

streakMultiplier = 1.5f;

else if (distance >= 300)

streakMultiplier = 1.2f;

else if (distance >= 100)

streakMultiplier = 1.1f;

else

streakMultiplier = 1.0f;

}

private void UpdateUI()

{

if (scoreText != null)

scoreText.text = Mathf.FloorToInt(currentScore).ToString();

if (distanceText != null)

distanceText.text = Mathf.FloorToInt(distance) + "m";

if (coinText != null)

coinText.text = coins.ToString();

}

// Gọi khi kết thúc lượt chơi

public void FinalizeScore()

{

int finalScore = Mathf.FloorToInt(currentScore);

// Lưu điểm cao

if (finalScore > DataManager.Instance.GetPlayerData().highScore)

{

DataManager.Instance.SetHighScore(finalScore);

}

// Thêm XP (1 XP mỗi 10 điểm)

int xpEarned = Mathf.FloorToInt(finalScore / 10.0f);

DataManager.Instance.AddXP(xpEarned);

// Lưu số xu

DataManager.Instance.AddCoins(coins);

// Hiển thị màn hình kết quả

UIManager.Instance.ShowRunResults(finalScore, Mathf.FloorToInt(distance), coins, xpEarned);

}

}

**2. Hệ thống cấp độ Ranger**

// RangerLevelSystem.cs

public class RangerLevelSystem : MonoBehaviour

{

[System.Serializable]

public class RangerLevel

{

public int level;

public int requiredXP;

public string title;

public Sprite badge;

public Reward[] rewards;

}

[System.Serializable]

public class Reward

{

public RewardManager.RewardType type;

public string itemId;

public int amount;

}

[SerializeField] private List<RangerLevel> rangerLevels = new List<RangerLevel>();

private int currentLevel = 1;

private int currentXP = 0;

private void Start()

{

// Tải từ DataManager

PlayerData playerData = DataManager.Instance.GetPlayerData();

currentLevel = playerData.level;

currentXP = playerData.xp;

}

public void AddXP(int amount)

{

int oldLevel = currentLevel;

currentXP += amount;

// Kiểm tra lên cấp

CheckLevelUp(oldLevel);

// Lưu vào DataManager

DataManager.Instance.GetPlayerData().xp = currentXP;

DataManager.Instance.GetPlayerData().level = currentLevel;

DataManager.Instance.SavePlayerData();

}

private void CheckLevelUp(int oldLevel)

{

// Kiểm tra có đủ XP để lên cấp không

while (true)

{

int nextLevelIndex = currentLevel; // Index trong mảng là level - 1

// Nếu đã đạt cấp độ tối đa

if (nextLevelIndex >= rangerLevels.Count)

break;

// Lấy thông tin cấp độ tiếp theo

RangerLevel nextLevel = rangerLevels[nextLevelIndex];

// Nếu đủ XP để lên cấp

if (currentXP >= nextLevel.requiredXP)

{

currentLevel++;

// Không trừ XP khi lên cấp, XP tích lũy

}

else

{

break;

}

}

// Nếu có lên cấp, cấp phát phần thưởng

if (currentLevel > oldLevel)

{

for (int level = oldLevel + 1; level <= currentLevel; level++)

{

RangerLevel rangerLevel = rangerLevels[level - 1];

GrantLevelUpRewards(rangerLevel);

// Hiển thị thông báo lên cấp

UIManager.Instance.ShowLevelUpNotification(level, rangerLevel.title, rangerLevel.badge, rangerLevel.rewards);

}

}

}

private void GrantLevelUpRewards(RangerLevel level)

{

foreach (Reward reward in level.rewards)

{

switch (reward.type)

{

case RewardManager.RewardType.Coins:

DataManager.Instance.AddCoins(reward.amount);

break;

case RewardManager.RewardType.ConservationStars:

DataManager.Instance.AddConservationStars(reward.amount);

break;

case RewardManager.RewardType.PowerUp:

DataManager.Instance.AddPowerUp(reward.itemId, reward.amount);

break;

case RewardManager.RewardType.AnimalUnlock:

DataManager.Instance.UnlockAnimal(reward.itemId);

break;

case RewardManager.RewardType.Cosmetic:

DataManager.Instance.UnlockCosmetic(reward.itemId);

break;

}

}

}

public int GetCurrentLevel()

{

return currentLevel;

}

public int GetCurrentXP()

{

return currentXP;

}

public int GetXPForNextLevel()

{

int nextLevelIndex = currentLevel;

if (nextLevelIndex >= rangerLevels.Count)

return -1; // Đã đạt cấp độ tối đa

return rangerLevels[nextLevelIndex].requiredXP;

}

public string GetCurrentRangerTitle()

{

int titleIndex = Mathf.Clamp(currentLevel - 1, 0, rangerLevels.Count - 1);

return rangerLevels[titleIndex].title;

}

public float GetLevelProgress()

{

int currentLevelXP = 0;

if (currentLevel > 1)

{

currentLevelXP = rangerLevels[currentLevel - 2].requiredXP;

}

int nextLevelXP = GetXPForNextLevel();

if (nextLevelXP == -1) // Đã đạt cấp độ tối đa

return 1.0f;

float progress = (float)(currentXP - currentLevelXP) / (nextLevelXP - currentLevelXP);

return Mathf.Clamp01(progress);

}

}

**3. Hệ thống nhiệm vụ hàng ngày**

// DailyQuestSystem.cs

public class DailyQuestSystem : MonoBehaviour

{

[System.Serializable]

public class Quest

{

public string id;

public string title;

public string description;

public QuestType type;

public int targetAmount;

public int currentAmount;

public Reward[] rewards;

public bool isCompleted;

}

[System.Serializable]

public class Reward

{

public RewardManager.RewardType type;

public string itemId;

public int amount;

}

public enum QuestType

{

RunDistance,

CollectCoins,

RescueAnimals,

CompletePuzzles,

InteractWithAnimals

}

[SerializeField] private List<Quest> questTemplates = new List<Quest>();

[SerializeField] private int dailyQuestCount = 3;

private List<Quest> activeQuests = new List<Quest>();

private DateTime lastQuestRefreshTime;

private void Start()

{

LoadQuestData();

CheckForQuestRefresh();

}

private void LoadQuestData()

{

// Tải từ PlayerPrefs

string questData = PlayerPrefs.GetString("ActiveQuests", "");

string lastRefreshString = PlayerPrefs.GetString("LastQuestRefresh", "");

if (!string.IsNullOrEmpty(lastRefreshString))

{

lastQuestRefreshTime = DateTime.Parse(lastRefreshString);

}

else

{

lastQuestRefreshTime = DateTime.Now.AddDays(-1); // Đảm bảo refresh ngay lập tức

}

if (!string.IsNullOrEmpty(questData))

{

activeQuests = JsonUtility.FromJson<QuestList>(questData).quests;

}

}

private void SaveQuestData()

{

QuestList questList = new QuestList { quests = activeQuests };

string questData = JsonUtility.ToJson(questList);

PlayerPrefs.SetString("ActiveQuests", questData);

PlayerPrefs.SetString("LastQuestRefresh", lastQuestRefreshTime.ToString());

PlayerPrefs.Save();

}

private void CheckForQuestRefresh()

{

DateTime now = DateTime.Now;

// Kiểm tra nếu ngày đã thay đổi

if (now.Date > lastQuestRefreshTime.Date)

{

RefreshDailyQuests();

lastQuestRefreshTime = now;

SaveQuestData();

}

}

private void RefreshDailyQuests()

{

activeQuests.Clear();

// Tạo danh sách các quest templates để chọn ngẫu nhiên

List<Quest> availableTemplates = new List<Quest>(questTemplates);

// Chọn ngẫu nhiên các quest cho ngày mới

for (int i = 0; i < dailyQuestCount; i++)

{

if (availableTemplates.Count == 0)

break;

int randomIndex = Random.Range(0, availableTemplates.Count);

Quest selectedTemplate = availableTemplates[randomIndex];

availableTemplates.RemoveAt(randomIndex);

// Tạo bản sao của template

Quest newQuest = new Quest

{

id = selectedTemplate.id,

title = selectedTemplate.title,

description = selectedTemplate.description,

type = selectedTemplate.type,

targetAmount = selectedTemplate.targetAmount,

currentAmount = 0,

rewards = selectedTemplate.rewards,

isCompleted = false

};

// Điều chỉnh mục tiêu dựa vào cấp độ người chơi

int playerLevel = DataManager.Instance.GetPlayerData().level;

float difficultyMultiplier = 1.0f + (playerLevel - 1) \* 0.05f; // Tăng 5% mỗi cấp

newQuest.targetAmount = Mathf.RoundToInt(newQuest.targetAmount \* difficultyMultiplier);

activeQuests.Add(newQuest);

}

// Báo cho UI cập nhật

UIManager.Instance.UpdateQuestUI(activeQuests);

}

public void UpdateQuestProgress(QuestType type, int amount)

{

bool anyQuestUpdated = false;

foreach (Quest quest in activeQuests)

{

if (quest.type == type && !quest.isCompleted)

{

quest.currentAmount += amount;

// Kiểm tra hoàn thành

if (quest.currentAmount >= quest.targetAmount)

{

quest.currentAmount = quest.targetAmount;

if (!quest.isCompleted)

{

CompleteQuest(quest);

}

}

anyQuestUpdated = true;

}

}

if (anyQuestUpdated)

{

SaveQuestData();

UIManager.Instance.UpdateQuestUI(activeQuests);

}

}

private void CompleteQuest(Quest quest)

{

quest.isCompleted = true;

// Cấp phát phần thưởng

foreach (Reward reward in quest.rewards)

{

switch (reward.type)

{

case RewardManager.RewardType.Coins:

DataManager.Instance.AddCoins(reward.amount);

break;

case RewardManager.RewardType.XP:

DataManager.Instance.AddXP(reward.amount);

break;

case RewardManager.RewardType.ConservationStars:

DataManager.Instance.AddConservationStars(reward.amount);

break;

case RewardManager.RewardType.PowerUp:

DataManager.Instance.AddPowerUp(reward.itemId, reward.amount);

break;

}

}

// Hiển thị thông báo hoàn thành

UIManager.Instance.ShowQuestCompletedNotification(quest);

}

public List<Quest> GetActiveQuests()

{

return activeQuests;

}

}

// Lớp hỗ trợ để serialize danh sách quest

[System.Serializable]

public class QuestList

{

public List<DailyQuestSystem.Quest> quests;

}

**CÁCH TÍCH HỢP VÀO GAMEPLAY**

**1. Thu thập xu trong Endless Runner**

* **Vị trí**: Đặt theo đường, hình chữ U, vòng tròn
* **Giá trị**: Mặc định 1 xu, xu đặc biệt 5-10 xu
* **Hiệu ứng**: Animation bay vào UI khi thu thập
* **Âm thanh**: Phản hồi âm thanh rõ ràng
* **Code sử dụng**:
* // Trong PlayerController.csprivate void OnTriggerEnter(Collider other){ if (other.CompareTag("Coin")) { int coinValue = other.GetComponent<CoinController>().GetValue(); ScoreManager.Instance.AddCoins(coinValue); // Hiệu ứng và âm thanh other.GetComponent<CoinController>().Collect(); // Cập nhật nhiệm vụ DailyQuestSystem.Instance.UpdateQuestProgress( DailyQuestSystem.QuestType.CollectCoins, coinValue); }}

**2. Hoàn thành mini-game với hệ thống điểm và thưởng**

* **Điểm gốc**: Dựa vào độ khó và thời gian hoàn thành
* **Điểm thưởng cho trẻ nhỏ**: Tăng mức thưởng cơ bản
* **Hiển thị kết quả**: Đánh giá bằng sao (1-3 sao)
* **Code mẫu**:
* // Trong MinigameController.cspublic void CompletePuzzle(bool success, float timeRemaining){ if (success) { // Tính điểm cơ bản int baseScore = 100; // Thưởng dựa trên thời gian còn lại float timeBonus = (timeRemaining / initialTime) \* 200; // Điều chỉnh theo độ tuổi string ageGroup = DataManager.Instance.GetPlayerData().ageGroup; float ageMultiplier = 1.0f; if (ageGroup == "4-7") ageMultiplier = 1.2f; // Thưởng thêm cho trẻ nhỏ int finalScore = Mathf.RoundToInt((baseScore + timeBonus) \* ageMultiplier); // Xác định số sao int stars = 1; if (timeRemaining > initialTime \* 0.7f) stars = 3; else if (timeRemaining > initialTime \* 0.4f) stars = 2; // Phần thưởng DataManager.Instance.AddXP(finalScore / 10); DataManager.Instance.AddCoins(stars \* 10); if (stars == 3) DataManager.Instance.AddConservationStars(1); // Hiển thị kết quả UIManager.Instance.ShowPuzzleResults(finalScore, stars); // Cập nhật nhiệm vụ DailyQuestSystem.Instance.UpdateQuestProgress( DailyQuestSystem.QuestType.CompletePuzzles, 1); } else { // Thất bại - vẫn cho một ít XP để khuyến khích DataManager.Instance.AddXP(10); UIManager.Instance.ShowPuzzleResults(0, 0); }}

**3. Hệ thống thưởng đăng nhập hàng ngày**

// DailyRewardSystem.cs

public class DailyRewardSystem : MonoBehaviour

{

[System.Serializable]

public class DailyReward

{

public int day; // Ngày trong chuỗi (1-7)

public RewardManager.RewardType type;

public string itemId;

public int amount;

}

[SerializeField] private List<DailyReward> weeklyRewards = new List<DailyReward>();

private DateTime lastRewardClaimTime;

private int currentStreak = 0;

private bool canClaimToday = false;

private void Start()

{

LoadDailyRewardData();

CheckCanClaimToday();

}

private void LoadDailyRewardData()

{

string lastClaimString = PlayerPrefs.GetString("LastRewardClaim", "");

currentStreak = PlayerPrefs.GetInt("CurrentRewardStreak", 0);

if (!string.IsNullOrEmpty(lastClaimString))

{

lastRewardClaimTime = DateTime.Parse(lastClaimString);

}

else

{

lastRewardClaimTime = DateTime.Now.AddDays(-2); // Đảm bảo ngày đầu tiên có thể claim

}

}

private void SaveDailyRewardData()

{

PlayerPrefs.SetString("LastRewardClaim", lastRewardClaimTime.ToString());

PlayerPrefs.SetInt("CurrentRewardStreak", currentStreak);

PlayerPrefs.Save();

}

private void CheckCanClaimToday()

{

DateTime now = DateTime.Now;

// Kiểm tra xem đã nhận phần thưởng hôm nay chưa

if (now.Date > lastRewardClaimTime.Date)

{

canClaimToday = true;

// Kiểm tra nếu bỏ lỡ ngày (reset streak)

if ((now.Date - lastRewardClaimTime.Date).Days > 1)

{

// Bỏ lỡ ít nhất 1 ngày

currentStreak = 0;

}

}

else

{

canClaimToday = false;

}

// Cập nhật UI

UIManager.Instance.UpdateDailyRewardUI(canClaimToday, currentStreak);

}

public void ClaimDailyReward()

{

if (!canClaimToday)

return;

// Tăng streak và cập nhật thời gian

currentStreak++;

lastRewardClaimTime = DateTime.Now;

canClaimToday = false;

// Xác định phần thưởng dựa vào streak

int dayInWeek = (currentStreak - 1) % 7 + 1; // 1-7

DailyReward reward = weeklyRewards.Find(r => r.day == dayInWeek);

if (reward != null)

{

// Cấp phát phần thưởng

switch (reward.type)

{

case RewardManager.RewardType.Coins:

DataManager.Instance.AddCoins(reward.amount);

break;

case RewardManager.RewardType.XP:

DataManager.Instance.AddXP(reward.amount);

break;

case RewardManager.RewardType.ConservationStars:

DataManager.Instance.AddConservationStars(reward.amount);

break;

case RewardManager.RewardType.PowerUp:

DataManager.Instance.AddPowerUp(reward.itemId, reward.amount);

break;

case RewardManager.RewardType.AnimalUnlock:

DataManager.Instance.UnlockAnimal(reward.itemId);

break;

case RewardManager.RewardType.Cosmetic:

DataManager.Instance.UnlockCosmetic(reward.itemId);

break;

}

// Hiển thị thông báo

UIManager.Instance.ShowDailyRewardClaimed(dayInWeek, reward);

}

SaveDailyRewardData();

UIManager.Instance.UpdateDailyRewardUI(canClaimToday, currentStreak);

}

public bool CanClaimToday()

{

return canClaimToday;

}

public int GetCurrentStreak()

{

return currentStreak;

}

public int GetDayInWeek()

{

return (currentStreak % 7) + 1;

}

}

**CÁC CHỈ SỐ CẦN THIẾT ĐỂ CÂN BẰNG HỆ THỐNG**

**1. Tốc độ tăng cấp độ**

* Cấp 1-10: Trung bình 30 phút chơi mỗi cấp
* Cấp 11-25: Trung bình 1 giờ chơi mỗi cấp
* Cấp 26-50: Trung bình 2 giờ chơi mỗi cấp
* Cấp 50+: Trung bình 4-5 giờ chơi mỗi cấp

**2. Tỷ lệ kiếm tiền**

* Trung bình 200-300 xu cho 10 phút chơi
* 50-100 xu từ phần thưởng hàng ngày
* 100-500 xu từ thưởng cấp độ

**3. Tỷ lệ kiếm Ngôi sao bảo tồn**

* Trung bình 1-2 sao mỗi ngày chơi ~30 phút
* 2-5 sao từ nhiệm vụ hàng tuần
* 3-10 sao từ thưởng cấp độ lớn

**4. Giá cả vật phẩm (ví dụ)**

* **Trang phục nhân vật**: 500-2000 xu
* **Môi trường mới**: 10-25 sao bảo tồn
* **Power-up (5 cái)**: 200-500 xu
* **Nâng cấp khu bảo tồn**: 300-1500 xu hoặc 1-5 sao

**BẢNG PHÂN BỐ PHẦN THƯỞNG THEO ĐỘ TUỔI**

| **Nhóm tuổi** | **Xu/phút chơi** | **XP/phút chơi** | **Hệ số độ khó mini-game** | **Hỗ trợ trực quan** |
| --- | --- | --- | --- | --- |
| 4-7 tuổi | 35-45 | 20-25 | 0.7 (dễ hơn) | Cao (hướng dẫn, gợi ý) |
| 8-11 tuổi | 25-35 | 15-20 | 1.0 (chuẩn) | Trung bình |
| 12-15 tuổi | 20-30 | 10-15 | 1.3 (khó hơn) | Thấp (ít hỗ trợ) |

**KẾT LUẬN VÀ ĐỀ XUẤT TRIỂN KHAI**

Hệ thống điểm và phần thưởng được thiết kế để tạo một vòng lặp tương tác hấp dẫn cho người chơi, đồng thời cân bằng được yếu tố giáo dục và giải trí. Dưới đây là kế hoạch triển khai theo thứ tự ưu tiên:

**Giai đoạn 1 (Prototype)**

1. Triển khai hệ thống điểm cơ bản trong Runner
2. Tạo hệ thống cấp độ Ranger đơn giản
3. Thêm cơ chế kiếm và sử dụng xu cơ bản

**Giai đoạn 2 (Beta)**

1. Hoàn thiện hệ thống điểm thưởng trong mini-game
2. Thêm hệ thống thành tựu và nhiệm vụ hàng ngày
3. Triển khai power-ups cơ bản
4. Thêm Ngôi sao bảo tồn và cơ chế kiếm/sử dụng

**Giai đoạn 3 (Phiên bản đầy đủ)**

1. Mở rộng hệ thống nhiệm vụ với các sự kiện đặc biệt
2. Triển khai cửa hàng và vật phẩm đa dạng
3. Thêm tương tác xã hội và bảng xếp hạng
4. Tích hợp hệ thống phân tích để cân bằng liên tục

Hệ thống này được thiết kế để linh hoạt, có thể mở rộng và điều chỉnh dựa trên phản hồi từ người dùng thực tế sau khi ra mắt phiên bản beta.