Отлично, давайте детально разберем второй этап разработки игры в Unity с фокусом на расширенных механиках. Этот этап займет 8-12 недель разработки.

### Этап 2: Расширение функционала и полировка

\*Цель:\* Полноценная игра с системой профессий, домом, торговлей и продвинутыми механиками

---

### 1. Система профессий и навыков (2 недели)

csharp

// ProfessionManager.cs

public enum Profession { Alchemist, Warrior, Gatherer, Merchant, Assassin }

public class ProfessionManager : MonoBehaviour

{

public Profession currentProfession;

private Dictionary<Profession, int> professionLevels = new();

private Dictionary<Profession, SkillTree> skillTrees = new();

public void UnlockProfession(Profession profession)

{

if (currentProfession != Profession.None) return;

currentProfession = profession;

professionLevels[profession] = 1;

skillTrees[profession] = new SkillTree();

ApplyProfessionBonuses();

}

private void ApplyProfessionBonuses()

{

switch (currentProfession)

{

case Profession.Alchemist:

PlayerStats.Instance.AddPotionEffectiveness(0.2f);

break;

case Profession.Warrior:

PlayerStats.Instance.AddDamageBonus(15);

break;

case Profession.Gatherer:

ResourceSystem.Instance.AddGatheringBonus(0.3f);

break;

}

}

public void GainProfessionXP(int amount)

{

professionLevels[currentProfession] += amount;

// Проверка нового уровня

}

}

// SkillTree.cs

[System.Serializable]

public class SkillTree

{

public List<SkillNode> nodes = new();

[System.Serializable]

public class SkillNode

{

public string skillName;

public string description;

public int requiredLevel;

public bool unlocked;

public SkillEffect effect;

}

public bool UnlockSkill(string skillName)

{

SkillNode node = nodes.Find(n => n.skillName == skillName);

if (node != null && !node.unlocked && professionLevel >= node.requiredLevel)

{

node.unlocked = true;

node.effect.Apply();

return true;

}

return false;

}

}

// SkillEffect.cs

public abstract class SkillEffect : ScriptableObject

{

public abstract void Apply();

}

[CreateAssetMenu(menuName = "Skills/AlchemistSkill")]

public class AdvancedPotionCrafting : SkillEffect

{

public override void Apply()

{

CraftingSystem.Instance.UnlockAdvancedPotions();

}

}

---

### 2. Система крафта и алхимии (1.5 недели)

csharp

// CraftingSystem.cs

public class CraftingSystem : MonoBehaviour

{

public List<Recipe> recipes = new();

private bool advancedPotionsUnlocked;

public void UnlockAdvancedPotions() => advancedPotionsUnlocked = true;

public bool CraftItem(Recipe recipe, Inventory inventory)

{

if (!CanCraft(recipe, inventory)) return false;

foreach (var ingredient in recipe.ingredients)

{

inventory.RemoveItem(ingredient.item, ingredient.quantity);

}

inventory.AddItem(recipe.result.item, recipe.result.quantity);

return true;

}

private bool CanCraft(Recipe recipe, Inventory inventory)

{

if (recipe.requiredProfession != Profession.None &&

PlayerProfession.currentProfession != recipe.requiredProfession) return false;

foreach (var ingredient in recipe.ingredients)

{

if (!inventory.HasItem(ingredient.item, ingredient.quantity)) return false;

}

return true;

}

}

// Recipe.cs

[System.Serializable]

public class Recipe

{

public string recipeName;

public List<Ingredient> ingredients;

public ItemStack result;

public Profession requiredProfession;

public int requiredLevel;

}

[System.Serializable]

public struct Ingredient

{

public ItemData item;

public int quantity;

}

[System.Serializable]

public struct ItemStack

{

public ItemData item;

public int quantity;

}

// Добавить в ItemData

public class CraftingMaterial : ItemData

{

public MaterialType materialType;

}

public enum MaterialType { Herb, Ore, Cloth, MonsterPart }

---

### 3. Система строительства и дома (1.5 недели)

csharp

// HousingSystem.cs

public class HousingSystem : MonoBehaviour

{

[SerializeField] private Transform houseLocation;

[SerializeField] private GameObject housePrefab;

[SerializeField] private CraftingStation[] availableStations;

private GameObject currentHouse;

private List<CraftingStation> placedStations = new();

public void BuildHouse()

{

if (currentHouse != null) return;

currentHouse = Instantiate(housePrefab, houseLocation.position, Quaternion.identity);

PlayerInventory.Instance.RemoveItem("Wood", 50);

PlayerInventory.Instance.RemoveItem("Stone", 100);

}

public void PlaceCraftingStation(CraftingStationType stationType)

{

CraftingStation stationPrefab = availableStations.FirstOrDefault(s => s.stationType == stationType);

if (stationPrefab == null) return;

CraftingStation station = Instantiate(stationPrefab, currentHouse.transform);

station.transform.localPosition = GetNextStationPosition();

placedStations.Add(station);

}

private Vector3 GetNextStationPosition()

{

return new Vector3(placedStations.Count \* 2f, 0, 0);

}

}

// CraftingStation.cs

public enum CraftingStationType { AlchemyLab, Forge, TanningRack }

public class CraftingStation : MonoBehaviour, IInteractable

{

public CraftingStationType stationType;

public Recipe[] availableRecipes;

public void Interact(Player player)

{

CraftingUI.Instance.OpenStation(this);

}

}

---

### 4. Расширенная система зелий и эффектов (1 неделя)

csharp

// PotionSystem.cs

public class PotionSystem : MonoBehaviour

{

public void ApplyPotionEffect(ConsumableItem potion)

{

StartCoroutine(ApplyEffectCoroutine(potion));

}

private IEnumerator ApplyEffectCoroutine(ConsumableItem potion)

{

PlayerStats stats = PlayerStats.Instance;

// Применение эффекта

stats.AddTemporaryEffect(potion.effect);

// Визуальные эффекты

VFXManager.Instance.PlayPotionEffect(potion.effectType);

// Длительность эффекта

yield return new WaitForSeconds(potion.duration);

// Снятие эффекта

stats.RemoveTemporaryEffect(potion.effect);

}

}

// StatusEffect.cs

[System.Serializable]

public class StatusEffect

{

public EffectType effectType;

public float duration;

public StatModifier[] modifiers;

}

[System.Serializable]

public struct StatModifier

{

public StatType statType;

public float value;

public ModifierType modifierType; // Flat или Percentage

}

public enum EffectType { Strength, Agility, Invisibility, Regeneration }

public enum StatType { Health, Damage, Defense, MoveSpeed }

public enum ModifierType { Flat, Percentage }

// Расширяем ItemData для зелий

public class ConsumableItem : ItemData

{

public StatusEffect effect;

public float duration = 60f;

}

---

### 5. Экономика и торговля (1 неделя)

csharp

// EconomySystem.cs

public class EconomySystem : MonoBehaviour

{

public Dictionary<string, float> itemBasePrices = new();

public float globalPriceMultiplier = 1.0f;

private void Awake()

{

// Инициализация базовых цен

itemBasePrices.Add("Herbs", 2f);

itemBasePrices.Add("IronOre", 5f);

itemBasePrices.Add("HealthPotion", 20f);

}

public float GetItemPrice(ItemData item, int quantity, bool isBuying)

{

if (!itemBasePrices.ContainsKey(item.itemName)) return 0f;

float basePrice = itemBasePrices[item.itemName];

float price = basePrice \* quantity \* globalPriceMultiplier;

// Модификаторы для покупателя/продавца

if (isBuying) price \*= 1.2f; // Наценка при покупке

else price \*= 0.8f; // Скидка при продаже

// Бонусы профессии

if (PlayerProfession.currentProfession == Profession.Merchant)

{

price \*= PlayerProfession.IsSkillUnlocked("Negotiation") ? 0.9f : 0.95f;

}

return price;

}

}

// MerchantNPC.cs

public class MerchantNPC : MonoBehaviour, IInteractable

{

[SerializeField] private List<ItemStack> buyItems = new();

[SerializeField] private List<ItemStack> sellItems = new();

public void Interact(Player player)

{

TradingUI.Instance.OpenMerchant(this);

}

}

// TradingUI.cs

public class TradingUI : MonoBehaviour

{

public void BuyItem(ItemData item, int quantity)

{

float price = EconomySystem.Instance.GetItemPrice(item, quantity, true);

if (PlayerInventory.Instance.Gold >= price)

{

PlayerInventory.Instance.RemoveGold(price);

PlayerInventory.Instance.AddItem(item, quantity);

}

}

public void SellItem(ItemData item, int quantity)

{

if (PlayerInventory.Instance.HasItem(item, quantity))

{

float price = EconomySystem.Instance.GetItemPrice(item, quantity, false);

PlayerInventory.Instance.RemoveItem(item, quantity);

PlayerInventory.Instance.AddGold(price);

}

}

}

---

### 6. Улучшенная генерация мира и чанков (1 неделя)

csharp

// WorldManager.cs (расширение)

private void GenerateWorld()

{

// Используем шум Перлина для генерации биомов

float[,] noiseMap = GenerateNoiseMap(mapSize, mapSize, scale, octaves, persistence, lacunarity);

for (int x = 0; x < mapSize; x++)

{

for (int y = 0; y < mapSize; y++)

{

float noiseValue = noiseMap[x, y];

BiomeType biome = GetBiomeFromNoise(noiseValue);

int dangerLevel = CalculateDangerLevel(x, y, biome);

Grid[x, y].Initialize(biome, dangerLevel);

}

}

PlaceSpecialLocations();

}

private BiomeType GetBiomeFromNoise(float noiseValue)

{

if (noiseValue < 0.3f) return BiomeType.Ocean;

if (noiseValue < 0.4f) return BiomeType.Beach;

if (noiseValue < 0.6f) return BiomeType.Plains;

if (noiseValue < 0.8f) return BiomeType.Forest;

return BiomeType.Mountains;

}

private void PlaceSpecialLocations()

{

// Размещаем города, деревни и особые локации

PlaceLocation("Starting Town", 50, 50, LocationType.Town);

PlaceLocation("Dark Forest", 30, 70, LocationType.Dungeon);

PlaceLocation("Iron Mine", 70, 30, LocationType.ResourceArea);

}

// ChunkLoader.cs для оптимизации

public class ChunkLoader : MonoBehaviour

{

[SerializeField] private int chunkSize = 10;

[SerializeField] private int loadRadius = 3;

private Vector2Int lastChunk = new(-1, -1);

private void Update()

{

Vector2Int currentChunk = GetCurrentChunk();

if (currentChunk != lastChunk)

{

LoadChunksAround(currentChunk);

UnloadDistantChunks(currentChunk);

lastChunk = currentChunk;

}

}

private Vector2Int GetCurrentChunk()

{

Vector3 playerPos = PlayerController.Instance.transform.position;

return new Vector2Int(

Mathf.FloorToInt(playerPos.x / chunkSize),

Mathf.FloorToInt(playerPos.y / chunkSize)

);

}

private void LoadChunksAround(Vector2Int center)

{

for (int x = center.x - loadRadius; x <= center.x + loadRadius; x++)

{

for (int y = center.y - loadRadius; y <= center.y + loadRadius; y++)

{

WorldManager.Instance.LoadChunk(x, y);

}

}

}

}

---

### 7. Система квестов и заданий (1 неделя)

csharp

// QuestSystem.cs

public class QuestSystem : MonoBehaviour

{

public List<Quest> activeQuests = new();

public List<Quest> completedQuests = new();

public void AcceptQuest(Quest quest)

{

activeQuests.Add(quest);

quest.OnQuestAccepted();

}

public void CompleteQuest(Quest quest)

{

if (activeQuests.Contains(quest) && quest.CheckCompletion())

{

activeQuests.Remove(quest);

completedQuests.Add(quest);

quest.OnQuestCompleted();

}

}

}

// Quest.cs

public abstract class Quest : ScriptableObject

{

public string questID;

public string title;

public string description;

public QuestReward reward;

public abstract bool CheckCompletion();

public abstract void OnQuestAccepted();

public abstract void OnQuestCompleted();

}

[CreateAssetMenu(menuName = "Quests/GatheringQuest")]

public class GatheringQuest : Quest

{

public ItemData requiredItem;

public int requiredAmount;

public override bool CheckCompletion()

{

return PlayerInventory.Instance.GetItemCount(requiredItem) >= requiredAmount;

}

public override void OnQuestAccepted()

{

// Регистрируем события

GameEvents.OnItemCollected += OnItemCollected;

}

private void OnItemCollected(ItemData item, int amount)

{

if (item == requiredItem)

{

// Обновляем UI квеста

}

}

}

// QuestGiverNPC.cs

public class QuestGiverNPC : MonoBehaviour, IInteractable

{

[SerializeField] private Quest[] availableQuests;

public void Interact(Player player)

{

QuestUI.Instance.ShowQuests(availableQuests);

}

}

---

### 8. Система сохранений и загрузки (3 дня)

csharp

// SaveSystem.cs

public class SaveSystem : MonoBehaviour

{

private const string SAVE\_KEY = "GameSave";

public void SaveGame()

{

SaveData data = new SaveData

{

playerPosition = PlayerController.Instance.CurrentPosition,

inventoryData = PlayerInventory.Instance.GetSaveData(),

professionData = ProfessionManager.Instance.GetSaveData(),

worldData = WorldManager.Instance.GetSaveData()

};

string json = JsonUtility.ToJson(data);

PlayerPrefs.SetString(SAVE\_KEY, json);

}

public void LoadGame()

{

if (PlayerPrefs.HasKey(SAVE\_KEY))

{

string json = PlayerPrefs.GetString(SAVE\_KEY);

SaveData data = JsonUtility.FromJson<SaveData>(json);

PlayerController.Instance.TeleportTo(data.playerPosition);

PlayerInventory.Instance.LoadFromData(data.inventoryData);

ProfessionManager.Instance.LoadFromData(data.professionData);

WorldManager.Instance.LoadFromData(data.worldData);

}

}

}

[System.Serializable]

public class SaveData

{

public Vector2Int playerPosition;

public InventoryData inventoryData;

public ProfessionData professionData;

public WorldData worldData;

}

[System.Serializable]

public class WorldData

{

public Dictionary<Vector2Int, MapTileData> tilesData;

}

[System.Serializable]

public class MapTileData

{

public bool resourcesCollected;

public bool eventsTriggered;

}

---

### 9. Оптимизация и полировка (1 неделя)

1. \*Оптимизации:\*

- Object Pooling для врагов и ресурсов

- Texture Atlasing для спрайтов

- LOD для тайлов карты

- Асинхронная загрузка чанков

2. \*Полировка:\*

- Система частиц для сбора ресурсов и боя

- Анимации UI

- Звуковые эффекты для всех действий

- Реакция окружения на игрока

3. \*Балансировка:\*

- Тестирование и настройка экономики

- Баланс боевой системы

- Настройка скоростей прокачки

csharp

// ObjectPool.cs

public class ObjectPool : MonoBehaviour

{

[System.Serializable]

public class Pool

{

public string tag;

public GameObject prefab;

public int size;

}

public List<Pool> pools;

public Dictionary<string, Queue<GameObject>> poolDictionary;

private void Start()

{

poolDictionary = new Dictionary<string, Queue<GameObject>>();

foreach (Pool pool in pools)

{

Queue<GameObject> objectPool = new Queue<GameObject>();

for (int i = 0; i < pool.size; i++)

{

GameObject obj = Instantiate(pool.prefab);

obj.SetActive(false);

objectPool.Enqueue(obj);

}

poolDictionary.Add(pool.tag, objectPool);

}

}

public GameObject SpawnFromPool(string tag, Vector3 position)

{

if (!poolDictionary.ContainsKey(tag)) return null;

GameObject objectToSpawn = poolDictionary[tag].Dequeue();

objectToSpawn.SetActive(true);

objectToSpawn.transform.position = position;

poolDictionary[tag].Enqueue(objectToSpawn);

return objectToSpawn;

}

}

---

### Порядок реализации второго этапа:

1. \*Неделя 1-2:\* Система профессий и навыков

2. \*Неделя 3:\* Расширенный крафт и алхимия

3. \*Неделя 4:\* Строительство дома и станции

4. \*Неделя 5:\* Система зелий и эффектов

5. \*Неделя 6:\* Экономика и торговля

6. \*Неделя 7:\* Улучшенная генерация мира

7. \*Неделя 8:\* Система квестов

8. \*Неделя 9:\* Сохранения и загрузка

9. \*Неделя 10:\* Оптимизация и полировка

\*Ключевые скрипты:\*

1. ProfessionManager

2. SkillTree

3. CraftingSystem

4. HousingSystem

5. PotionSystem

6. EconomySystem

7. MerchantNPC

8. ChunkLoader

9. QuestSystem

10. SaveSystem

11. ObjectPool

\*Технические улучшения:\*

- Использование Addressable Assets для управления ресурсами

- Реализация UI через UI Toolkit

- Интеграция DOTween для плавных анимаций

- Использование Odin Inspector для удобной настройки данных

- Добавление локализации через I2 Localization

\*Тестирование:\*

1. Юнит-тесты для критических систем (инвентарь, бой, экономика)

2. Баланс-тесты для боевой системы и экономики

3. Стресс-тесты для системы чанков

4. Юзабилити-тестирование интерфейса

Этот план позволяет создать полноценную RPG с глубокой системой профессий, экономикой и строительством. Основной фокус на модульности систем - каждая механика реализована как независимый модуль, что упрощает тестирование и балансировку.