Отлично, давайте детализируем первый этап разработки игры в Unity с фокусом на реализацию ключевых механик. Вот пошаговый план с практической реализацией.

### Этап 1: Базовый прототип (4-6 недель)

\*Цель:\* Рабочий каркас с основными механиками перемещения, сбора ресурсов, инвентаря и боя

---

### 1. Система карты мира (1 неделя)

\*Архитектура:\*

csharp

// WorldManager.cs

public class WorldManager : MonoBehaviour

{

public static WorldManager Instance;

[SerializeField] private int mapSize = 100;

[SerializeField] private GameObject tilePrefab;

[SerializeField] private Transform mapContainer;

public MapTile[,] Grid { get; private set; }

private void Awake() => Instance = this;

private void Start() => GenerateWorld();

private void GenerateWorld()

{

Grid = new MapTile[mapSize, mapSize];

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

{

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

{

GameObject tileObj = Instantiate(tilePrefab, new Vector3(x, y, 0), Quaternion.identity, mapContainer);

MapTile tile = tileObj.GetComponent<MapTile>();

// Определяем биом по координатам

if (y < 10) tile.Initialize(BiomeType.Water, 1);

else if (x > 80 && y > 80) tile.Initialize(BiomeType.Mountain, 4);

else tile.Initialize((BiomeType)Random.Range(0, 3), Random.Range(1, 4));

Grid[x, y] = tile;

}

}

}

}

// MapTile.cs

public class MapTile : MonoBehaviour

{

[SerializeField] private SpriteRenderer renderer;

[SerializeField] private Sprite[] biomeSprites;

public BiomeType Biome { get; private set; }

public int DangerLevel { get; private set; }

public bool HasResources { get; set; } = true;

public void Initialize(BiomeType biome, int dangerLevel)

{

Biome = biome;

DangerLevel = dangerLevel;

renderer.sprite = biomeSprites[(int)biome];

}

}

public enum BiomeType { Forest, Steppe, Water, Mountain }

---

### 2. Система перемещения игрока (3 дня)

csharp

// PlayerController.cs

public class PlayerController : MonoBehaviour

{

[SerializeField] private float moveDuration = 20f;

[SerializeField] private SpriteRenderer playerIcon;

public Vector2Int CurrentPosition { get; private set; }

private bool isMoving;

private void Start()

{

CurrentPosition = new Vector2Int(50, 50); // Старт в центре карты

transform.position = new Vector3(CurrentPosition.x, CurrentPosition.y, 0);

}

public void Move(Direction direction)

{

if (isMoving) return;

Vector2Int targetPos = CurrentPosition + direction.ToVector2Int();

if (WorldManager.Instance.IsValidPosition(targetPos))

{

StartCoroutine(MoveCoroutine(targetPos));

}

}

private IEnumerator MoveCoroutine(Vector2Int target)

{

isMoving = true;

float elapsed = 0;

Vector3 startPos = transform.position;

Vector3 endPos = new Vector3(target.x, target.y, 0);

while (elapsed < moveDuration)

{

transform.position = Vector3.Lerp(startPos, endPos, elapsed / moveDuration);

elapsed += Time.deltaTime;

yield return null;

}

transform.position = endPos;

CurrentPosition = target;

isMoving = false;

// Активируем события клетки

WorldManager.Instance.Grid[target.x, target.y].OnPlayerEnter();

}

}

public enum Direction { Up, Down, Left, Right }

public static class DirectionExtensions

{

public static Vector2Int ToVector2Int(this Direction dir)

{

return dir switch

{

Direction.Up => Vector2Int.up,

Direction.Down => Vector2Int.down,

Direction.Left => Vector2Int.left,

Direction.Right => Vector2Int.right,

\_ => Vector2Int.zero

};

}

}

---

### 3. Инвентарь и система предметов (1 неделя)

csharp

// InventorySystem.cs

public class InventorySystem : MonoBehaviour

{

[SerializeField] private float maxWeight = 50f;

[SerializeField] private Transform itemsContainer;

[SerializeField] private GameObject itemUIPrefab;

private float currentWeight;

private readonly List<Item> items = new();

public bool AddItem(ItemData data)

{

if (currentWeight + data.weight > maxWeight) return false;

items.Add(new Item(data));

currentWeight += data.weight;

UpdateUI();

return true;

}

private void UpdateUI()

{

// Очищаем контейнер

foreach (Transform child in itemsContainer) Destroy(child.gameObject);

// Создаем новые элементы UI

foreach (Item item in items)

{

GameObject itemUI = Instantiate(itemUIPrefab, itemsContainer);

itemUI.GetComponent<ItemUI>().Initialize(item.Data);

}

}

}

// Item.cs (Data container)

[System.Serializable]

public class Item

{

public ItemData Data { get; }

public int Quantity { get; set; } = 1;

public Item(ItemData data) => Data = data;

}

// Create in Resources/Items

[CreateAssetMenu(fileName = "NewItem", menuName = "Game/Item Data")]

public class ItemData : ScriptableObject

{

public string itemName;

public float weight;

public Sprite icon;

public ItemType type;

}

public enum ItemType { Resource, Weapon, Armor, Consumable }

---

### 4. Система ресурсов и сбора (4 дня)

csharp

// ResourceSystem.cs

public class ResourceSystem : MonoBehaviour

{

[SerializeField] private ResourceData[] resourcesData;

public ItemData GetResourceForBiome(BiomeType biome)

{

foreach (ResourceData data in resourcesData)

{

if (data.biome == biome)

return data.item;

}

return null;

}

}

[System.Serializable]

public struct ResourceData

{

public BiomeType biome;

public ItemData item;

}

// Добавить в MapTile.cs

public void OnPlayerEnter()

{

if (!HasResources) return;

ItemData resource = ResourceSystem.Instance.GetResourceForBiome(Biome);

if (resource != null && PlayerInventory.Instance.AddItem(resource))

{

HasResources = false;

renderer.color = new Color(0.7f, 0.7f, 0.7f); // Визуальное обозначение

}

}

---

### 5. Боевая система (1.5 недели)

csharp

// CombatSystem.cs

public class CombatSystem : MonoBehaviour

{

[SerializeField] private GameObject combatUI;

[SerializeField] private Text combatLog;

private PlayerStats player;

private Enemy currentEnemy;

private bool playerTurn = true;

public void StartCombat(Enemy enemy)

{

combatUI.SetActive(true);

currentEnemy = enemy;

player = PlayerStats.Instance;

playerTurn = true;

UpdateLog($"На вас напал {enemy.Name}!");

}

public void PlayerAttack()

{

if (!playerTurn) return;

int damage = Mathf.Max(1, player.Attack - currentEnemy.Defense);

currentEnemy.TakeDamage(damage);

UpdateLog($"Вы нанесли {damage} урона!");

if (currentEnemy.IsDead)

{

EndCombat(true);

return;

}

playerTurn = false;

Invoke("EnemyTurn", 1f);

}

private void EnemyTurn()

{

int damage = Mathf.Max(1, currentEnemy.Attack - player.Defense);

player.TakeDamage(damage);

UpdateLog($"{currentEnemy.Name} наносит {damage} урона!");

if (player.IsDead) EndCombat(false);

else playerTurn = true;

}

private void EndCombat(bool playerWon)

{

UpdateLog(playerWon ? "Победа!" : "Поражение...");

Invoke("HideCombatUI", 2f);

}

private void HideCombatUI() => combatUI.SetActive(false);

private void UpdateLog(string text) => combatLog.text += "\n" + text;

}

// Enemy.cs

public class Enemy : MonoBehaviour

{

public string Name => enemyName;

[SerializeField] private string enemyName;

[SerializeField] private int health;

[SerializeField] private int attack;

[SerializeField] private int defense;

public int Health { get; private set; }

public int Attack => attack;

public int Defense => defense;

public bool IsDead => Health <= 0;

private void Start() => Health = health;

public void TakeDamage(int damage)

{

Health = Mathf.Max(0, Health - damage);

}

}

---

### 6. Случайные события и враги (3 дня)

csharp

// Добавить в MapTile.cs

public void CheckForRandomEvent()

{

if (Random.value < DangerLevel \* 0.15f)

{

SpawnEnemy();

}

}

private void SpawnEnemy()

{

GameObject enemyObj = Instantiate(EnemyManager.Instance.GetRandomEnemy(DangerLevel));

enemyObj.transform.position = transform.position;

CombatSystem.Instance.StartCombat(enemyObj.GetComponent<Enemy>());

}

// EnemyManager.cs

public class EnemyManager : MonoBehaviour

{

[SerializeField] private Enemy[] forestEnemies;

[SerializeField] private Enemy[] steppeEnemies;

[SerializeField] private Enemy[] waterEnemies;

[SerializeField] private Enemy[] mountainEnemies;

public GameObject GetRandomEnemy(int dangerLevel)

{

Enemy[] pool = GetEnemyPool(WorldManager.Instance.Grid[PlayerPosition.x, PlayerPosition.y].Biome);

Enemy enemy = pool[Random.Range(0, pool.Length)];

return Instantiate(enemy.gameObject);

}

private Enemy[] GetEnemyPool(BiomeType biome)

{

return biome switch

{

BiomeType.Forest => forestEnemies,

BiomeType.Steppe => steppeEnemies,

BiomeType.Water => waterEnemies,

BiomeType.Mountain => mountainEnemies,

\_ => forestEnemies

};

}

}

---

### 7. UI и управление (4 дня)

\*Реализовать:\*

1. Кнопки перемещения (стрелки)

2. Панель инвентаря с отображением веса

3. Окно боя с кнопками действий

4. Таймер перемещения

5. Статус игрока (здоровье, уровень)

csharp

// UIManager.cs

public class UIManager : MonoBehaviour

{

[SerializeField] private Text positionText;

[SerializeField] private Text weightText;

[SerializeField] private Slider healthSlider;

[SerializeField] private Slider moveProgressSlider;

private void Update()

{

UpdatePlayerUI();

}

private void UpdatePlayerUI()

{

PlayerController player = PlayerController.Instance;

positionText.text = $"Position: {player.CurrentPosition}";

weightText.text = $"Weight: {InventorySystem.Instance.CurrentWeight}/{InventorySystem.Instance.MaxWeight}";

healthSlider.value = PlayerStats.Instance.HealthPercent;

}

public void UpdateMoveProgress(float progress)

{

moveProgressSlider.value = progress;

}

}

---

### 8. Настройка сцены и префабы

1. \*Игровая сцена:\*

- Grid (сетка 100x100)

- Camera (ортографическая)

- Canvas (Screen Space - Overlay)

2. \*Префабы:\*

- MapTile (SpriteRenderer + MapTile.cs)

- Player (SpriteRenderer + PlayerController.cs)

- Enemy (SpriteRenderer + Enemy.cs)

- ItemUI (Image + Button)

3. \*Настройки:\*

yaml

Project Settings:

- Physics 2D: Отключить гравитацию

- Input: Настроить кнопки управления

- Graphics: 2D URP pipeline

---

### Порядок реализации

1. Настройка проекта и базовой сцены

2. Система генерации мира

3. Контроллер игрока и перемещение

4. Система инвентаря и предметов

5. Механика сбора ресурсов

6. Система боя и враги

7. Случайные события на клетках

8. Интерфейс пользователя

9. Балансировка и тестирование

\*Ключевые скрипты:\*

1. WorldManager

2. MapTile

3. PlayerController

4. InventorySystem

5. ResourceSystem

6. CombatSystem

7. EnemyManager

8. UIManager

\*Технические требования:\*

- Unity 2021.3+ LTS

- 2D URP Pipeline

- Input System Package

- TextMeshPro

Такой подход позволяет создать работающий прототип за 4-6 недель с возможностью дальнейшего расширения. Основной фокус - на базовых механиках без глубокой детализации профессий и дома, которые пойдут во втором этапе.