LevelMap Module Documentation

Обзор

LevelMap Module - это готовое решение для Unity, которое предоставляет полнофункциональную систему карты уровней для мобильных и десктопных игр. Модуль реализован с использованием архитектурного паттерна MVC (Model-View-Controller) и обеспечивает отделение игровой логики от представления.

Возможности

- **Прогрессия уровней**: Автоматическое разблокирование следующего уровня при завершении текущего
- **Система звезд**: Отслеживание рейтинга прохождения уровней (1-3 звезды)
- **Сохранение прогресса**: Автоматическое сохранение состояния в PlayerPrefs
- **Настраиваемый UI**: Гибкая настройка внешнего вида через ScriptableObject
- **События**: Система событий для интеграции с другими модулями игры
- **Отзывчивый интерфейс**: Поддержка различных разрешений экрана

Архитектура

MVC Pattern

Модуль следует архитектуре MVC:

- **Model** (`LevelMapModel`): Управляет данными и бизнес-логикой
- **View** (`LevelMapView`): Отвечает за отображение UI
- **Controller** (`LevelMapController`): Координирует взаимодействие между Model и View

Основные компоненты

1. Данные

LevelData - представляет информацию об отдельном уровне:

```csharp
public class LevelData
{
    public int levelId;           // Уникальный ID уровня
    public bool isUnlocked;       // Разблокирован ли уровень
    public bool isCompleted;      // Завершен ли уровень
    public int stars;             // Количество звезд (0-3)
    public Vector2 mapPosition;   // Позиция на карте
    public string levelName;      // Название уровня
    public Sprite levelIcon;      // Иконка уровня
}
```

LevelMapData - содержит всю информацию о карте уровней:

```csharp
public class LevelMapData
{
    public List<LevelData> levels;        // Список всех уровней
    public int currentLevel;              // Текущий выбранный уровень
    public int maxUnlockedLevel;          // Максимальный разблокированный уровень
}
```

2. Интерфейсы

ILevelMapController - основной интерфейс для работы с модулем:

```csharp
public interface ILevelMapController
{
    void Initialize();                                  // Инициализация модуля
    void ShowLevelMap();                                // Показать карту уровней
    void HideLevelMap();                                // Скрыть карту уровней
    void CompleteLevel(int levelId, int stars);         // Завершить уровень
    void UnlockLevel(int levelId);                      // Разблокировать уровень
    LevelData GetLevel(int levelId);                    // Получить данные уровня
}
```

3. События

Модуль предоставляет следующие события:

В Controller:

- `OnLevelSelected` - Вызывается при выборе уровня

- `OnBackPressed` - Вызывается при нажатии кнопки "Назад"

В Model:

- `OnLevelUnlocked` - Вызывается при разблокировке уровня
- `OnLevelCompleted` - Вызывается при завершении уровня
- `OnCurrentLevelChanged` - Вызывается при изменении текущего уровня

Установка и настройка

1. Импорт модуля

1. Скопируйте все файлы модуля в папку `Assets/Scripts/LevelMapModule/`
2. Убедитесь, что все скрипты находятся в namespace `LevelMapModule`

2. Настройка UI

Создание префаба кнопки уровня

1. Создайте GameObject с компонентом `Button`
2. Добавьте дочерние объекты:
   - `Text` - для отображения номера уровня
   - `Image` - для иконки уровня
   - `Stars Container` - контейнер со звездами (1-3 объекта)
   - `Lock Icon` - иконка блокировки
   - `Completed Icon` - иконка завершения
3. Добавьте компонент `LevelButton` и настройте ссылки на UI элементы

Создание главного UI

1. Создайте Canvas для карты уровней
2. Добавьте контейнер для кнопок уровней (например, с `GridLayoutGroup`)
3. Добавьте кнопку "Назад"
4. На главном GameObject карты добавьте компонент `LevelMapView`

3. Настройка LevelMapViewSettings

Создайте и настройте `LevelMapViewSettings`:

csharp

```csharp
[SerializeField] private LevelMapViewSettings settings = new LevelMapViewSettings
{
    levelButtonPrefab = prefabReference,      // Ссылка на префаб кнопки
    levelButtonsContainer = containerTransform, // Контейнер для кнопок
    backButton = backButtonReference,         // Кнопка "Назад"
    mapCanvas = canvasReference,              // Canvas карты уровней
    buttonSpacing = 100f,                     // Расстояние между кнопками
    buttonsPerRow = 5                         // Количество кнопок в ряду
};
```

Использование

Инициализация модуля

csharp

```csharp
// Создание экземпляров
var model = new LevelMapModel();
var view = FindObjectOfType<LevelMapView>();
var controller = new LevelMapController(model, view);

// Подписка на события
controller.OnLevelSelected += OnLevelSelected;
controller.OnBackPressed += OnBackPressed;

// Инициализация
controller.Initialize();
```

Основные операции

Показать/скрыть карту уровней

csharp

```csharp
controller.ShowLevelMap();  // Показать
controller.HideLevelMap();  // Скрыть
```

Завершение уровня

csharp

```csharp
// Завершить уровень с 3 звездами
controller.CompleteLevel(levelId: 1, stars: 3);

// При завершении уровня автоматически разблокируется следующий
```

Разблокировка уровня

csharp

```csharp
// Разблокировать конкретный уровень
controller.UnlockLevel(levelId: 5);
```

Получение информации о уровне

csharp

```csharp
LevelData levelData = controller.GetLevel(levelId: 1);
if (levelData != null)
{
    Debug.Log($"Level {levelData.levelId}: Unlocked={levelData.isUnlocked}, Stars={levelData.st
}
```

Обработка событий

csharp

```csharp
private void OnLevelSelected(int levelId)
{
    Debug.Log($"Player selected level {levelId}");
    // Загрузить выбранный уровень
    SceneManager.LoadScene($"Level_{levelId}");
}

private void OnBackPressed()
{
    Debug.Log("Back button pressed");
    // Вернуться в главное меню
    SceneManager.LoadScene("MainMenu");
}
```

Расширение функциональности

Кастомизация данных уровня

Вы можете расширить `LevelData` дополнительными полями:

```csharp
[Serializable]
public class ExtendedLevelData : LevelData
{
    public string description;
    public float bestTime;
    public int coinsCollected;

    public ExtendedLevelData(int id) : base(id) { }
}
```

Кастомные условия разблокировки

Переопределите логику в `LevelMapModel`:

```csharp
public class CustomLevelMapModel : LevelMapModel
{
    public override void CompleteLevel(int levelId, int stars = 0)
    {
        base.CompleteLevel(levelId, stars);

        // Кастомная логика разблокировки
        if (stars >= 2 && levelId % 5 == 0) // Каждый 5-й уровень требует минимум 2 звезды
        {
            UnlockLevel(levelId + 1);
        }
    }
}
```

Анимации и эффекты

Добавьте анимации в `LevelMapView`:

```csharp
public void UpdateLevel(LevelData levelData)
{
    if (levelButtons.TryGetValue(levelData.levelId, out var button))
    {
        button.UpdateData(levelData);

        // Добавить анимацию разблокировки
        if (levelData.isUnlocked)
        {
            PlayUnlockAnimation(button);
        }
    }
}
```

Лучшие практики

1. Управление памятью

Всегда вызывайте `Dispose()` при уничтожении контроллера:

```csharp
private void OnDestroy()
{
    controller?.Dispose();
}
```

2. Обработка ошибок

Модуль включает базовую обработку ошибок для сохранения/загрузки данных. Для production-кода рекомендуется добавить дополнительные проверки.

3. Производительность

- Используйте объектные пулы для кнопок уровней при большом количестве уровней (>100)
- Рассмотрите ленивую загрузку UI элементов для оптимизации времени загрузки

4. Тестирование

Создайте тестовые сценарии для проверки:

- Корректного сохранения/загрузки прогресса
- Правильной работы событий
- Обработки граничных случаев (несуществующие уровни, некорректные данные)

Заключение

LevelMap Module предоставляет гибкое и расширяемое решение для создания карт уровней в Unity играх. Модуль следует принципам SOLID и позволяет легко кастомизировать как внешний вид, так и игровую логику под конкретные требования проекта.