МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Донецкий национальный технический университет»

Факультет ИСП

Кафедра ПИ им Л.П.Фельдмана

Лабораторная работа № 4

на тему: «Создание самодокументирующегося кода»

по курсу: «Профессиональная практика программной инженерии»

Проверил:

асс. каф. ПИ им. Л.П.Фельдмана Филипишин Д.А.

Выполнил:

ст. гр. ПИ-21В

Чащин А.Ю.

Донецк-2024

Цель работы – научиться добавлять в программный код специальным образом оформление докблок-комментарии, для последующей автоматической генерации API reference, а также познакомиться с форматом оформления документации DocBook.

Вариант 6. Видео игра RPG (Аналогичная DarkSouls);

Ссылка на аккаунт: <https://github.com/0Anchan0>

Ссылка на репозиторий: <https://github.com/0Anchan0/Fallen-Kingdom-PPPI>

// --- Character Progression ---  
using UnityEngine;  
  
/// <summary>  
/// Класс <c>CharacterProgression</c> управляет прогрессией персонажа в игре Fallen Kingdom.  
/// Он отвечает за атрибуты персонажа, такие как сила и ловкость, и предоставляет методы для их изменения.  
/// </summary>  
public class CharacterProgression : MonoBehaviour  
{  
 /// <summary>  
 /// Сила персонажа, влияющая на физическую мощь и урон в ближнем бою.  
 /// </summary>  
 public int strength = 10;  
  
 /// <summary>  
 /// Ловкость персонажа, влияющая на скорость передвижения и уклонение от атак.  
 /// </summary>  
 public int agility = 10;  
  
 /// <summary>  
 /// Увеличивает атрибуты персонажа при повышении уровня.  
 /// Каждый раз, когда вызывается этот метод, сила и ловкость увеличиваются на 2,  
 /// что отражает прогрессию персонажа в хардкорной RPG.  
 /// </summary>  
 public void LevelUp()  
 {  
 strength += 2;  
 agility += 2;  
 Debug.Log($"New stats - Strength: {strength}, Agility: {agility}");  
 }  
  
 /// <summary>  
 /// Устанавливает значение силы персонажа.  
 /// Используется для настройки параметров персонажа в зависимости от выбранного билда или класса.  
 /// </summary>  
 /// <param name="newStrength">Новое значение силы.</param>  
 public void SetStrength(int newStrength)  
 {  
 strength = newStrength;  
 Debug.Log($"Strength set to: {strength}");  
 }  
  
 /// <summary>  
 /// Устанавливает значение ловкости персонажа.  
 /// Позволяет адаптировать персонажа к различным стилям игры и тактикам.  
 /// </summary>  
 /// <param name="newAgility">Новое значение ловкости.</param>  
 public void SetAgility(int newAgility)  
 {  
 agility = newAgility;  
 Debug.Log($"Agility set to: {agility}");  
 }  
  
 /// <summary>  
 /// Возвращает текущие атрибуты персонажа в виде строки.  
 /// Используется для отображения информации о состоянии персонажа в интерфейсе игры.  
 /// </summary>  
 /// <returns>Строка с текущими значениями силы и ловкости.</returns>  
 public string GetStats()  
 {  
 return $"Current stats - Strength: {strength}, Agility: {agility}";  
 }  
  
 /// <summary>  
 /// Увеличивает силу персонажа на заданное количество.  
 /// Это может быть полезно при использовании предметов или навыков, которые временно увеличивают силу.  
 /// </summary>  
 /// <param name="amount">Количество, на которое нужно увеличить силу.</param>  
 public void IncreaseStrength(int amount)  
 {  
 strength += amount;  
 Debug.Log($"Strength increased by {amount}. New strength: {strength}");  
 }  
  
 /// <summary>  
 /// Увеличивает ловкость персонажа на заданное количество.  
 /// Это может быть полезно для адаптации персонажа к различным ситуациям в бою.  
 /// </summary>  
 /// <param name="amount">Количество, на которое нужно увеличить ловкость.</param>  
 public void IncreaseAgility(int amount)  
 {  
 agility += amount;  
 Debug.Log($"Agility increased by {amount}. New agility: {agility}");  
 }  
}

Рисунок 1 – Пример файл с ДокБлоками CharacterProgression.cs

Doxygen — это инструмент для документирования исходного кода, который поддерживает множество языков программирования, включая C, C++, Java, Python и другие. Он позволяет разработчикам генерировать документацию в различных форматах, таких как HTML, LaTeX и RTF, на основе комментариев, встроенных в код. Doxygen анализирует структуру кода, включая классы, функции и их взаимосвязи, что позволяет создавать подробные и структурированные документы, облегчающие понимание и использование программного обеспечения.

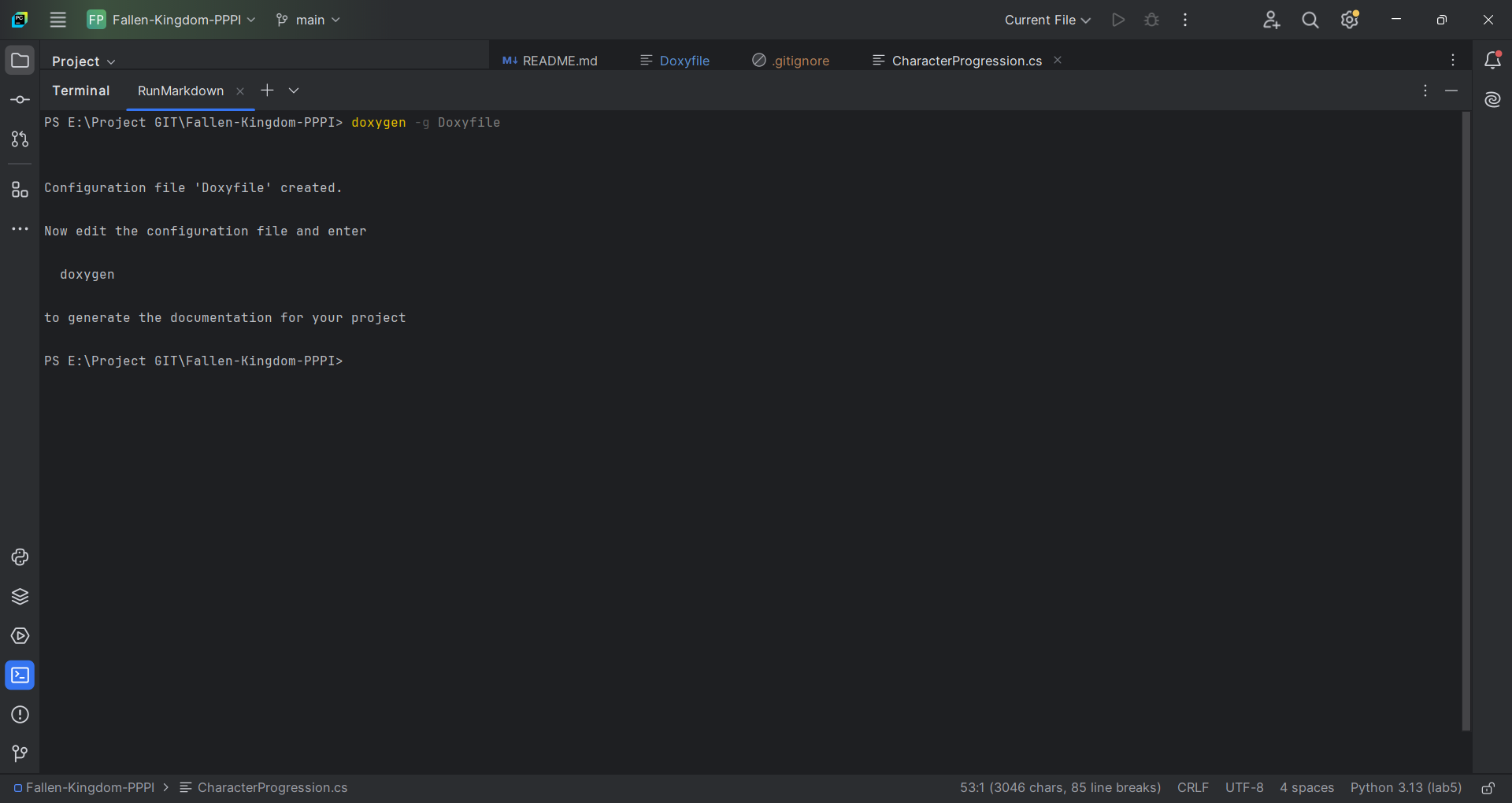


Рисунок 2 – Пример начала работы с Doxygen

Структура полученных файлов представлена на рисунке 3.

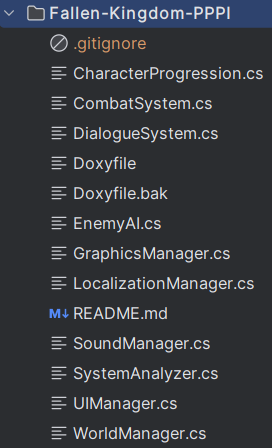


Рисунок 3 – Структура проекта

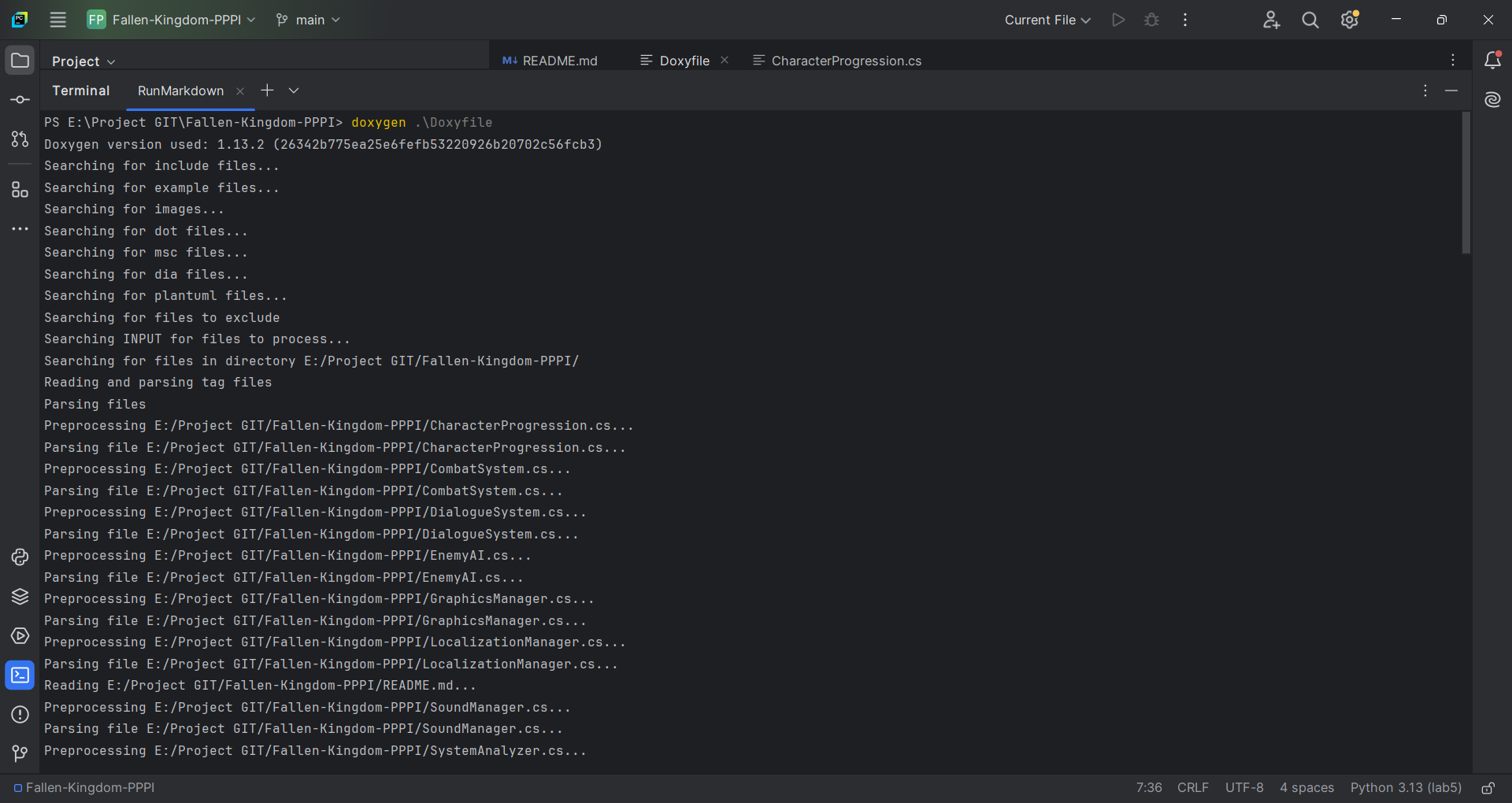


Рисунок 4- Пример создания автоматической документации через doxygen

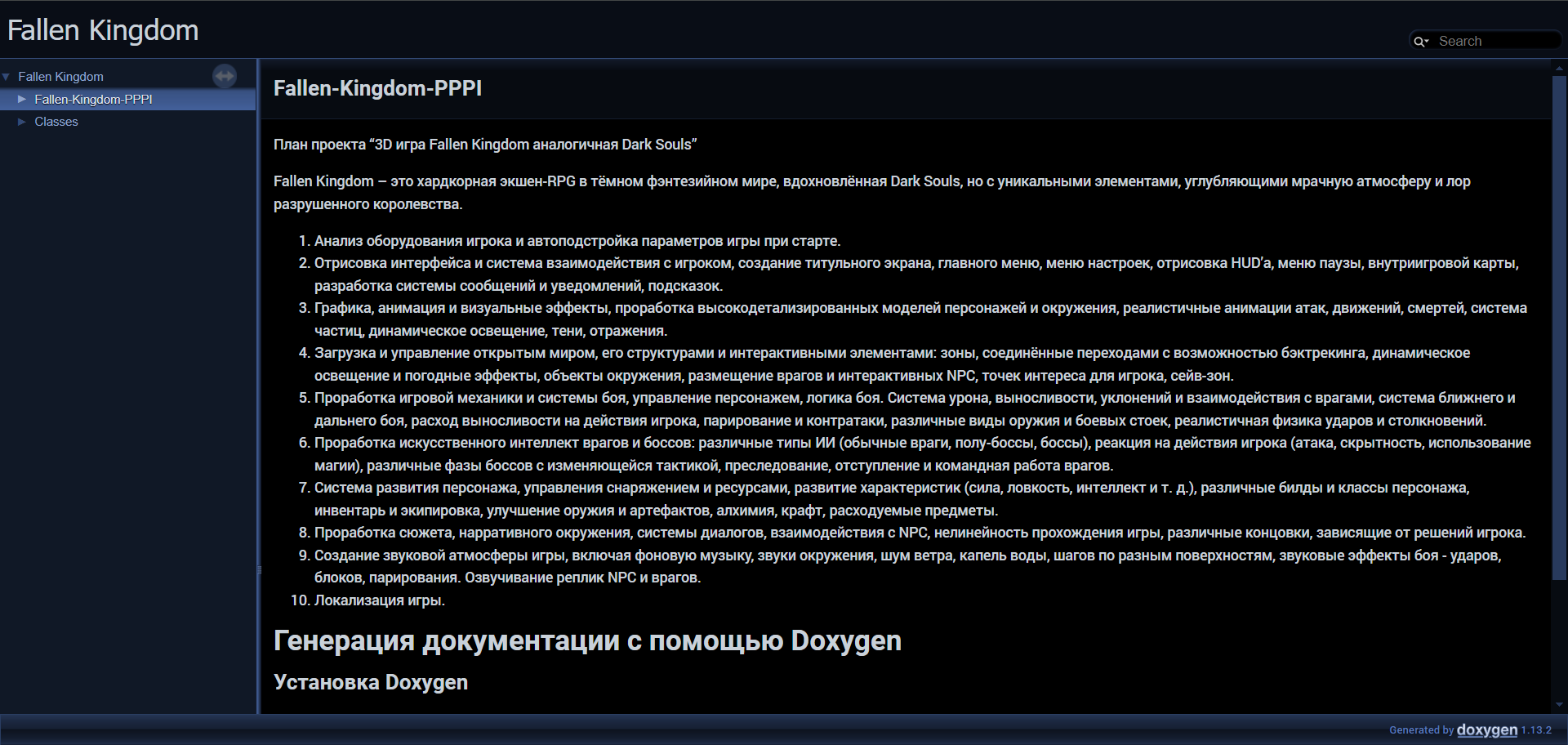


Рисунок 5 – Главная страница получившейся документации

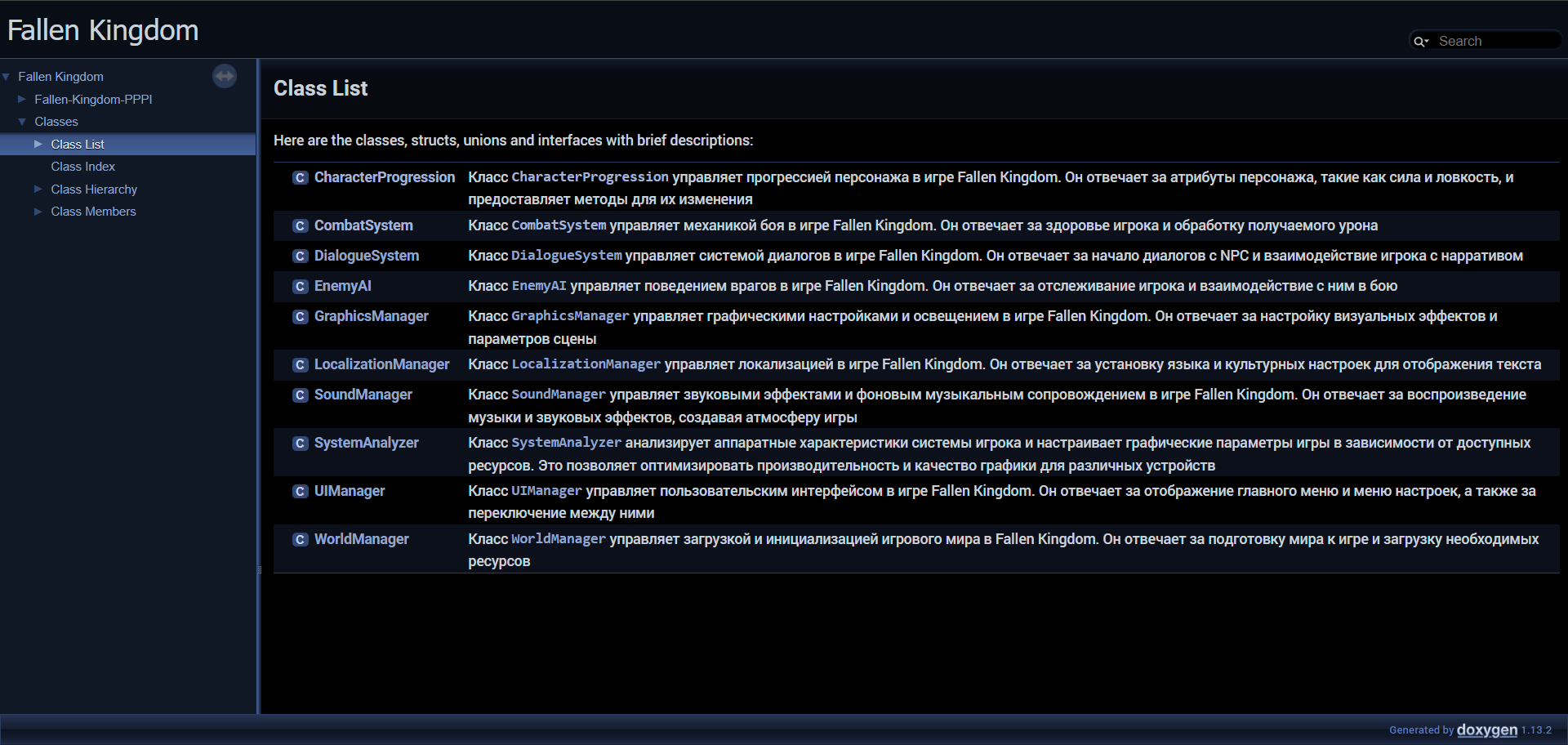


Рисунок 6 – Список классов для которых была создана документация



Рисунок 7 – Пример отображения документации для класса CharacterProgression