|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на выполнение работ по созданию

компьютерной игры.

Выполнил: студент учебной группы

ИСПк-301-51-00

Целищев Вадим

Тюлькин Данила

Преподаватель:

Долженкова Мария

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

2022

Оглавление

[1 Введение 3](#_Toc120000022)

[1.1 Наименование программного продукта 3](#_Toc120000023)

[1.2 Краткая характеристика области применения 3](#_Toc120000024)

[2 Основание для разработки. 4](#_Toc120000025)

[3 Назначение для разработки. 5](#_Toc120000026)

[3.1 Функциональная назначение 5](#_Toc120000027)

[3.2 Эксплуатационное назначение 5](#_Toc120000028)

[4 Требования к программе или программному изделию. 6](#_Toc120000029)

[4.1 Требования к функциональным характеристикам 6](#_Toc120000030)

[4.2 Требования к надёжности 6](#_Toc120000031)

[4.2.1 Требования к обеспечению надежного (устойчивого) функционального функционирования программы 6](#_Toc120000032)

[4.2.2 Время восстановления после отказа 6](#_Toc120000033)

[4.2.3 Отказы из-за некорректных действий оператора 6](#_Toc120000034)

[4.3 Условия эксплуатации 7](#_Toc120000035)

[4.3.1 Климатические условия эксплуатации 7](#_Toc120000036)

[4.3.2 Требования к видам обслуживания 7](#_Toc120000037)

[4.3.3 Требования к численности и квалификации персонала 7](#_Toc120000038)

[4.4 Требования к составу и параметрам технических средств 7](#_Toc120000039)

[4.5 Требования к информационной и программной совместимости 8](#_Toc120000040)

[4.5.1 Требования к защите информации и программ 8](#_Toc120000041)

[4.5.2 Требование к маркировке и упаковке 8](#_Toc120000042)

[4.5.3 Требования к транспортированию и хранению 8](#_Toc120000043)

[4.5.4 Специальные требования 8](#_Toc120000044)

[5 Технико-экономические показатели 9](#_Toc120000045)

[6 Стадии и этапы разработки. 10](#_Toc120000046)

[6.1 Стадии разработки: 10](#_Toc120000047)

[6.2 Этапы разработки. 10](#_Toc120000048)

[7 Порядок контроля и приемки. 12](#_Toc120000049)

# Введение

## Наименование программного продукта

Наименование программы: Компьютерная игра на игровом движке Unity: “ Dungeon Master.”

## Краткая характеристика области применения

Область применения – игровая индустрия для развлечения.

# Основание для разработки.

Учебный план специальности 09.02.07 Информационные системы и программирование, утверждённый в 2022 году.

Разработка идёт на основании составленного раннее анализа предметной области, который был утверждён Долженковой Марией Львовной 1.09.2022.

Период выполнения:

Начало разработки:

Конец разработки: 21.11.2022

Заказчик: коллектив преподавателей ВятГУ в составе: Долженкова Мария Львовна.

Исполнитель: студент колледжа ВятГУ ИСПк-301-51-00 Целищев Вадим Сергеевич.

# Назначение для разработки.

## Функциональная назначение

Программа является продуктом развлечения в сфере компьютерных развлечений. Программа предназначена для получения, повышения и применения навыков, полученных на дисциплинах: учебная практика, анализ и разработка ТЗ.

## Эксплуатационное назначение

Программа будет эксплуатироваться как приложение для персональных компьютеров.

# Требования к программе или программному изделию.

## Требования к функциональным характеристикам

Основными требования являются функции, которые в общем сформируют общую картину программного продукта. Все функции и интерфейс изображён и описан в руководстве пользователя, а их реализация находятся в руководстве программиста.

## Требования к надёжности

### Требования к обеспечению надежного (устойчивого) функционального функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. Организация бесперебойного питания технических средств;
2. Содержание технических средств в рабочем состоянии.

### Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 10 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

### Отказы из-за некорректных действий оператора

Программа не должна непредвиденно прерывать свою работу. После запуска приложения отказ приложения вследствие непредвиденной ошибки –исключен.

## Условия эксплуатации

Уверенный пользователь персональный компьютер.

### Климатические условия эксплуатации

Условия эксплуатации должны соответствовать стандартным требованиям к бытовым помещениям.

### Требования к видам обслуживания

Программа не требует проведения каких-либо видов обслуживания.

### Требования к численности и квалификации персонала

Пользователь программного обеспечения должен обладать базовыми навыками работы с ПК.

## Требования к составу и параметрам технических средств

Для работы программного продукта необходима следующая программно-аппаратная конфигурация:

1. Windows 7, Windows Server 2003 Service Pack 2, Windows Server 2008, Windows Server 2008 R2, Windows Vista, Windows Vista Service Pack 1, Windows XP Service Pack 2, Windows XP Service Pack 3;
2. 32-разрядные системы: компьютер, оборудованный процессором Intel или совместимым процессором с тактовой частотой 1 ГГц или выше (рекомендуется 2 ГГц или выше, поддерживается только один процессор);
3. 64-разрядные системы: процессор с тактовой частотой 1,4 ГГц или выше (рекомендуется 2 ГГц или выше, поддерживается только один процессор);
4. минимальный объем ОЗУ 512 МБ (рекомендуется 1 ГБ или более);
5. 250 МБ свободного места на диске;

## Требования к информационной и программной совместимости

### Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

### Требование к маркировке и упаковке

Программа поставляет в виде программного изделия на внешнем носителе, где должны содержаться: программная документация, исполняемые файлы, прочие файлы, необходимые для работы программы. Требования к маркировке и упаковке не предусмотрены. Требования к защите информации и программ не предъявляются

### Требования к транспортированию и хранению

Дополнительные требования к транспортированию и хранению не предъявляются.

### Специальные требования

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

# Технико-экономические показатели

Расчёт экономической эффективности не предусмотрен. Экономическое преимущество разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами отсутствует.

# Стадии и этапы разработки.

## Стадии разработки:

Разработка должна быть проведена в три стадии:    
1. разработка технического задания;    
2. рабочее проектирование;    
3. внедрение.

## Этапы разработки.

На стадии «Техническое задание» должен быть выполнен этап разработки, согласования и утверждения технического задания.

На стадии «рабочее проектирование» должны быть выполнены перечисленные ниже этапы работ:

1. Разработка программы;
2. Разработка программной документации;
3. Испытания программы;

На стадии «Внедрение» должен быть выполнен быть выполнен этап разработки «Подготовка и передача программы».

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;
2. определение и уточнение требований к техническим средствам;
3. определение требований к программе;
4. определение стадий, этапов и сроков разработки программы и документации на неё;
5. согласование и утверждение технического задания.    
   На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.    
   На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями к составу документации.    
   На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:
6. разработка, согласование и утверждение и методики испытаний;
7. проведение приемо-сдаточных испытаний;
8. корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена подготовка и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

# Порядок контроля и приемки.

Испытания представленной программы и контроль качества ее работы провести на базе компьютерного класса. Во время испытаний проверить работу системы по следующим позициям:   
1) ПО должно корректно проходить весь набор функциональных тестов;

2) корректное функционирование заданных в ТЗ функций;

3) возможность функционирования на ЭВМ с указанными минимальными системными требованиями;

4) демонстрация возможности установки на указанную систему.

Протокол приема сдаточных испытаний

Тест-кейс 1

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 1** | TC\_1 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Проигрыш |
| **Резюме испытания** | При тестировании нужно достигнуть факта пройгрыша |
| **Шаги тестирования** | 1. Открыть игру 2. Довести до состояния единицы здоровья персонажа 3. Провести ещё ход 4. Начать игру с начала |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Проигрыш |
| **Фактический результат** | Проигрыш |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Единица здоровья персонажа |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 1:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public int HP = 1;

public int HP\_M = 20;

bool a = true;

[Test]

public void NewTestScriptSimplePasses()

{

}

[UnityTest]

public IEnumerator NewTestScriptWithEnumeratorPasses()

{

HP -= 1;

if (HP <= 0)

{

HP\_M = 100;

HP = 100;

}

if ((HP\_M == 100)&&(HP ==100))

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 2

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 2** | TC\_2 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Победа над противником |
| **Резюме испытания** | При тестировании нужно достигнуть факта победы над противником |
| **Шаги тестирования** | 1. Открыть игру 2. Довести до состояния количества здоровья противника равного или меньше урона персонажа 3. Провести ещё ход 4. Получить +1 счёта |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Победа над противником |
| **Фактический результат** | Победа над противником |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Количество здоровья противника равного или меньше урона персонажа |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 2:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript1

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 1;

public int Level = 0;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 0;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void qq()

{

Debug.Log("Damadge");

HP\_M -= damg;

HP -= 1;

}

qq();

if (HP\_M <= 0)

{

mana += 10;

Level += 1;

HP\_M = Monstr;

Monstr += 10;

}

if (Level>0)

{

a = true;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 3

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 3** | TC\_3 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Нанесение урона противнику |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие кнопки урона |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку урона |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Нанесение урона |
| **Фактический результат** | Нанесение урона |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 3:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript2

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 1;

public int Level = 0;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 0;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

int HP\_M\_old = HP\_M;

void qq()

{

Debug.Log("Damadge");

HP\_M -= damg;

HP -= 1;

}

qq();

if (HP\_M\_old>HP\_M)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 4

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 4** | TC\_4 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Неудачное использование мощного удара |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие мощного удара с недостаточным его количеством |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку мощного удара |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Отсутствие реакции от персонажа |
| **Фактический результат** | Отсутствие реакции от персонажа |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 4:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript3

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 0;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 0;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void tt()

{

Debug.Log("Magic");

if (mana >= 50)

{

HP\_M -= 50;

mana -= 50;

HP -= 1;

}

}

tt();

if (HP\_M == 100)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 5

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 5** | TC\_5 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Удачное использование мощного удара |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие мощного удара с достаточным его количеством |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку мощного удара |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Нанесение мощного удара |
| **Фактический результат** | Нанесение мощного удара |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 5:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript4

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 0;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 50;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void tt()

{

Debug.Log("Magic");

if (mana >= 50)

{

HP\_M -= 50;

mana -= 50;

HP -= 1;

}

}

tt();

if (HP\_M < 100)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 6

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 6** | TC\_6 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Неудачное улучшение урона |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие улучшения урона с недостаточным количеством счёта |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку улучшения урона |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Отсутствие реакции от персонажа |
| **Фактический результат** | Отсутствие реакции от персонажа |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 6:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript5

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 0;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 50;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void rr()

{

Debug.Log("Sword+");

if (Level >= scoreS)

{

Level -= scoreS;

scoreS += 1;

damg += 5;

}

}

rr();

if (damg == 10)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 7

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 7** | TC\_7 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Удачное улучшение урона |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие улучшения урона с достаточным количеством счёта |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку улучшения урона |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Улучшение оружия |
| **Фактический результат** | Улучшение оружия |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 7:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript6

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 1;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 50;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void rr()

{

Debug.Log("Sword+");

if (Level >= scoreS)

{

Level -= scoreS;

scoreS += 1;

damg += 5;

}

}

rr();

if ((damg > 10) && (Level == 0))

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 8

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 8** | TC\_8 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Неудачное улучшение здоровья |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие улучшения здоровья с недостаточным количеством счёта |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку улучшения здоровья |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Отсутствие реакции от персонажа |
| **Фактический результат** | Отсутствие реакции от персонажа |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 8:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript7

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 0;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 50;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void aptech()

{

Debug.Log("HP+");

if (Level >= scoreH)

{

Level -= scoreH;

HP += 10;

}

}

aptech();

if (HP == 100)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 9

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 9** | TC\_9 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Удачное улучшение здоровья |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие улучшения здоровья с достаточным количеством счёта |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку улучшения здоровья |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Улучшение здоровья |
| **Фактический результат** | Улучшение здоровья |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 9:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript8

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 1;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 50;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void aptech()

{

Debug.Log("HP+");

if (Level >= scoreH)

{

Level -= scoreH;

HP += 10;

}

}

aptech();

if (HP > 100)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

Тест-кейс 10

|  |  |
| --- | --- |
| **Наименование** | **Описание** |
| **Test Case 10** | TC\_10 |
| **Приоритет тестирования** *(Малый/Средний/высокий)* | Высокий |
| **Название тестирования/Имя** | Работоспособность кнопки защиты |
| **Резюме испытания** | При тестировании нужно достигнуть факта нажатие кнопки защиты |
| **Шаги тестирования** | 1. Открыть игру 2. Нажать на кнопку защиты |
| **Данные тестирования** | Игра |
| **Ожидаемый результат** | Защита от получения урона |
| **Фактический результат** | Защита от получения урона |
| **Предпосылки** | На устройстве должно присутствовать игра |
| **Постусловия** | Нажатие кнопки |
| **Статус** *(Pass/Fail)* | Pass |
| **Комментарии** | Тест пройден |

Тест-кейс 10:

using System.Collections;

using System.Collections.Generic;

using NUnit.Framework;

using UnityEngine;

using UnityEngine.TestTools;

public class NewTestScript9

{

public GameObject Text\_hp;

public GameObject Text\_hp\_M;

public GameObject Text\_level;

public GameObject Text\_sword;

public GameObject Text\_mana;

public int HP = 100;

public int HP\_M = 100;

public int Level = 1;

public int damg = 10;

public int scoreS = 1;

public int scoreH = 1;

public int mana = 50;

public int Monstr = 100;

bool a = false;

[Test]

public void SimplePasses()

{

}

[UnityTest]

public IEnumerator WithEnumeratorPasses()

{

void ww()

{

Debug.Log("Defense");

HP\_M -= 0;

HP -= 0;

}

ww();

if (HP == 100)

{

a = true;

}

else

{

a = false;

}

Assert.IsTrue(a);

yield return null;

}

}

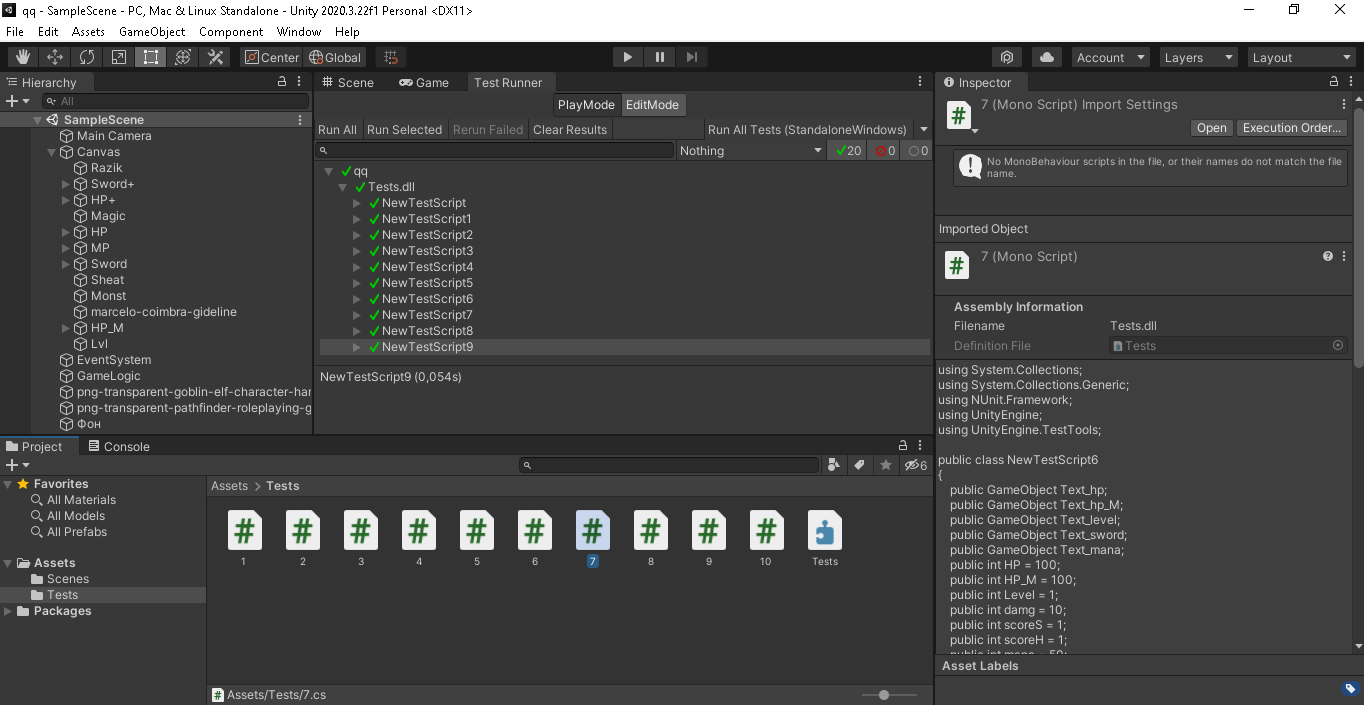


Рисунок 2 – результат работы тест-кейсов