TSR
Test Summary Report
Dashing Cube

Riferimento	Nc11_dashingcube-tp.docx
Versione	0.3
Data	08/12/2024
Destinatario	Prof. Carmine Gravino
Presentato da	Vincenzo Beniamino Fresa, Francesco Botta
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
22/12/2024	0.1	Stesura dell'introduzione	Francesco Botta Vincenzo Beniamino Fresa
26/12/2024	0.2	Stesura dei Testing	Francesco Botta
02/01/2025	0.3	Prima Revisione del Documento	Francesco Botta

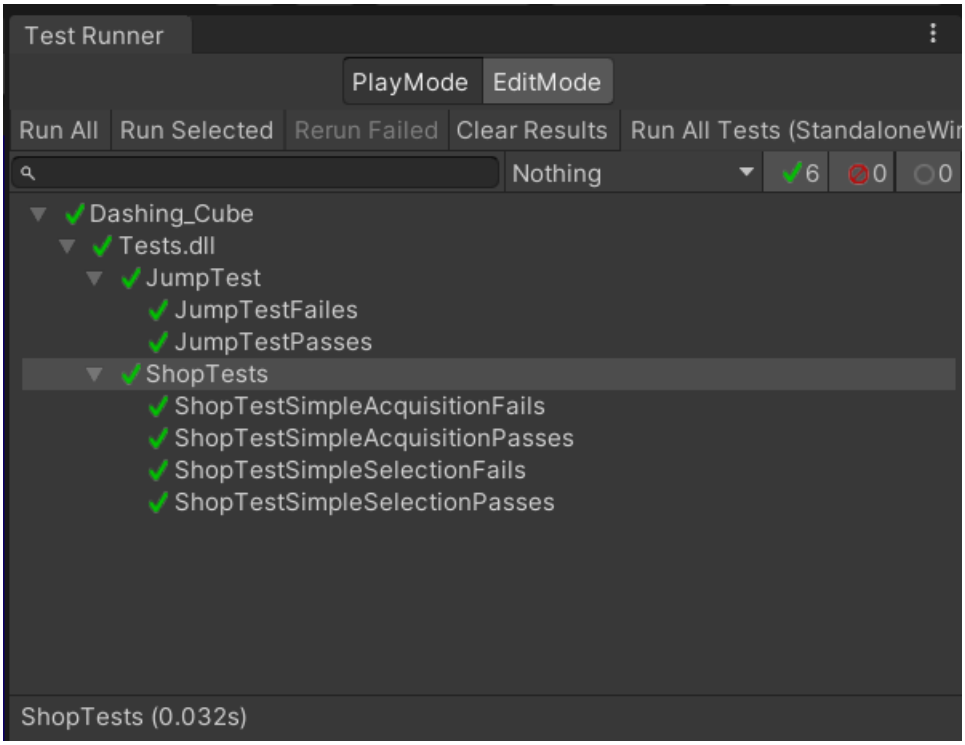
Obiettivo

Il seguente documento si pone l'obiettivo di riportare e descrivere i Testing effettuati per una corretta funzione dell'intero sistema.
Le funzionalità testate sono quelle citate dal Test Plan:

- Salto del Giocatore
- Selezione di un Cosmetico
- Acquisto di un Cosmetico

Unit Testing

I Test di Unità sono stati svolti gestendo alternativamente i vari compiti, sfruttando ciò che l'engine Unity offre per il lavoro di suddetto testing. Nel momento in cui si notavano degli errori, i programmatori hanno provveduto a risolverli.



Salto del Giocatore (1.1 e 1.2)

```
using System.Collections;
using System.Collections.Generic;
using NUnit.Framework;
using UnityEngine;
using UnityEngine.TestTools;

0 riferimenti
public class JumpTest
{

    [UnityTest]
    0 riferimenti
    public IEnumerator JumpTestPasses()
    {
        // Use the Assert class to test conditions.
        // Use yield to skip a frame.
        var player = new GameObject();
        var playerController = player.AddComponent<PlayerController>();
        var rb = player.AddComponent<Rigidbody2D>();

        rb.sleepMode = RigidbodySleepMode2D.NeverSleep;
        playerController.onTheField = true;
        playerController.yJump = 8f;


        playerController.Jump();



        yield return new WaitForSeconds(0);

        Assert.AreEqual(playerController.onTheField, false);
    }

    [UnityTest]
    0 riferimenti
    public IEnumerator JumpTestFailes()
    {
        // Use the Assert class to test conditions.
        // Use yield to skip a frame.
        var player = new GameObject();
        var playerController = player.AddComponent<PlayerController>();
        var rb = player.AddComponent<Rigidbody2D>();

        rb.sleepMode = RigidbodySleepMode2D.NeverSleep;
        playerController.onTheField = false;
        playerController.yJump = 8f;


        playerController.Jump();



        yield return new WaitForSeconds(playerController.maximumHeightSec);

        Assert.AreEqual(playerController.onTheField, false);  //La logica è che il salto non viene eseguito proprio e per questo motivo non rientra nel metodo
    }
}
```

Selezione di un Cosmetico (2.1 e 2.2)

```
using NUnit.Framework;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.TestTools;

Script Unity | 0 riferimenti
public class ShopTests : MonoBehaviour
{
    private GameObject shopObject;
    private ShopController shopController;
    private MenuController menuController;
    private FileManager fileManager;

    [SetUp]
    0 riferimenti
    public void SetUp()
    {
        // Crea un GameObject con tutti i componenti necessari
        shopObject = new GameObject();
        shopController = shopObject.AddComponent<ShopController>();
        fileManager = shopObject.AddComponent<FileManager>();


        // Inizializza i campi privati con valori di esempio
        shopController.isSkinAcquired = new bool[6] { false, false, false, false, false, false };
        shopController.isAlreadySelected = new bool[6] { false, false, false, false, false, false };
        shopController.coinsRequired = new int[] { 100, 200, 300, 400, 500, 600 };

        shopController.coins = 500;

    }

    // A Test behaves as an ordinary method
    [Test]
    0 riferimenti
    public void ShopTestSimpleSelectionPasses()
    {
        shopController.isSkinAcquired[0] = true;
        shopController.IsSkinSelected0();

        Assert.AreEqual(shopController.isAlreadySelected[0], true);

    }

    [Test]
    0 riferimenti
    public void ShopTestSimpleSelectionFails()
    {
        shopController.isSkinAcquired[0] = false;
        shopController.IsSkinSelected0();

        Assert.AreEqual(shopController.isAlreadySelected[0], false);

    }
```

Acquisto di un Cosmetico (3.1 e 3.4)

```
[Test]
0 riferimenti
public void ShopTestSimpleAcquisitionPasses()
{
    shopController.isSkinAcquired[0] = false;
    shopController.coins = 500;
    shopController.IsSkinSelected0();

    Assert.AreEqual(shopController.isSkinAcquired[0], true);

}

[Test]
0 riferimenti
public void ShopTestSimpleAcquisitionFails()
{
    shopController.isSkinAcquired[0] = false;
    shopController.coins = 5;
    shopController.IsSkinSelected0();

    Assert.AreEqual(shopController.isSkinAcquired[0], false);

}
}
```

System Testing

Per il testing di sistema si sono manualmente eseguite tutte le funzionalità a mano a mano implementate ed ogni volta che si presentava un errore, si cercava di risolvere il suddetto prima dell'aggiunta di nuove meccaniche nel sistema.