TEST DRIVEN DEVELOPMENT WITH UNITY

Düsseldorf Unity Meetup, 2017-05-04

Jay Allan Jethwa
Team Lead Software Development
InnoGames Düsseldorf

Joined InnoGames in 2011
Before: Ubisoft Blue Byte

Currently working on
Unannounced Project

Die Siedler® online

FORGE OF EMPIRES

the Settlers My City

1. Introduction
2. Test Driven Development By Example
3. Unity Test Tools
4. Conclusion

INTRODUCTION

What is Test Driven Development?

Test-driven development (**TDD**) is a software development process that relies on the repetition of a very short development cycle:

First the developer writes an (initially failing) automated test case that defines a desired improvement or new function,

then produces the minimum amount of code to pass that test

and finally refactors the new code to acceptable standards.

Source: https://en.wikipedia.org/wiki/Test-driven_development

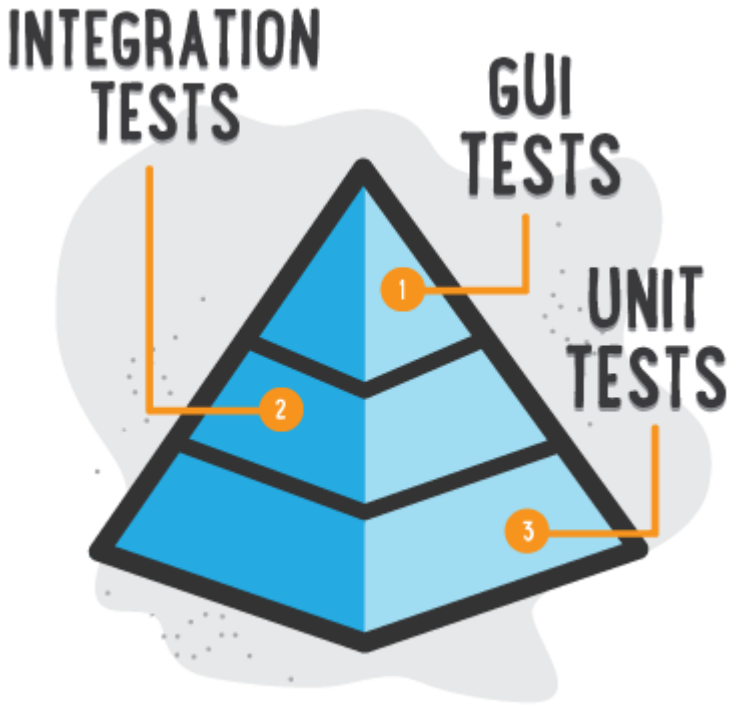


ILLUSTRATION BY SEGUE TECHNOLOGIES

Source: http://www.seguetech.com/blog/2014/10/10/benefits-unit-testing

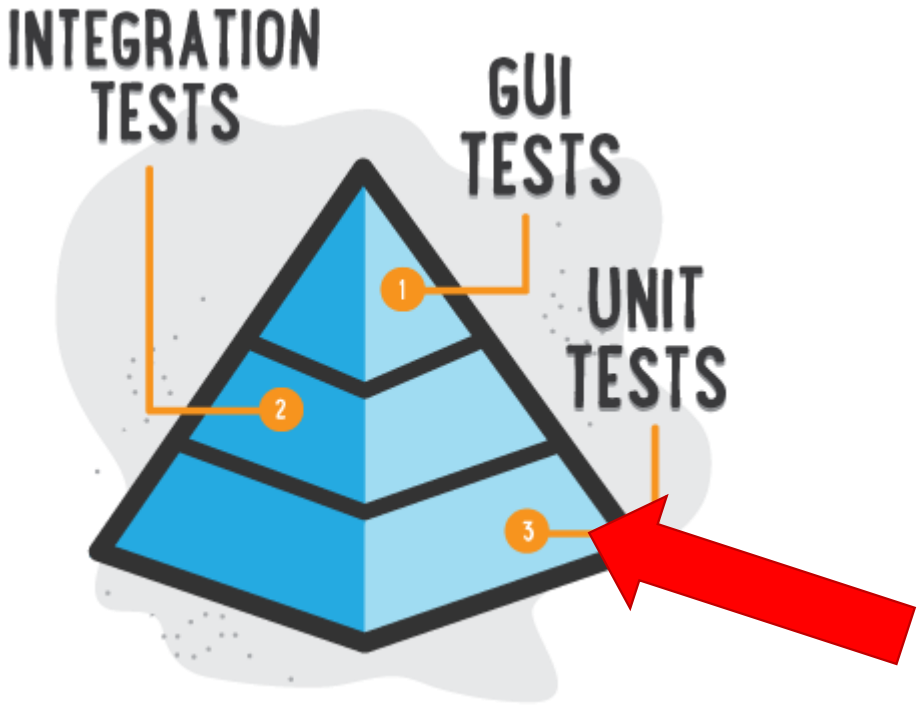


ILLUSTRATION BY SEGUE TECHNOLOGIES

Source: http://www.seguetech.com/blog/2014/10/10/benefits-unit-testing

Unit Tests verify a **single assumption** about the behavior of a system

Source: https://www.flickr.com/photos/86530412@N02/8226451812/

Health: 50

Asset by: http://kenney.nl

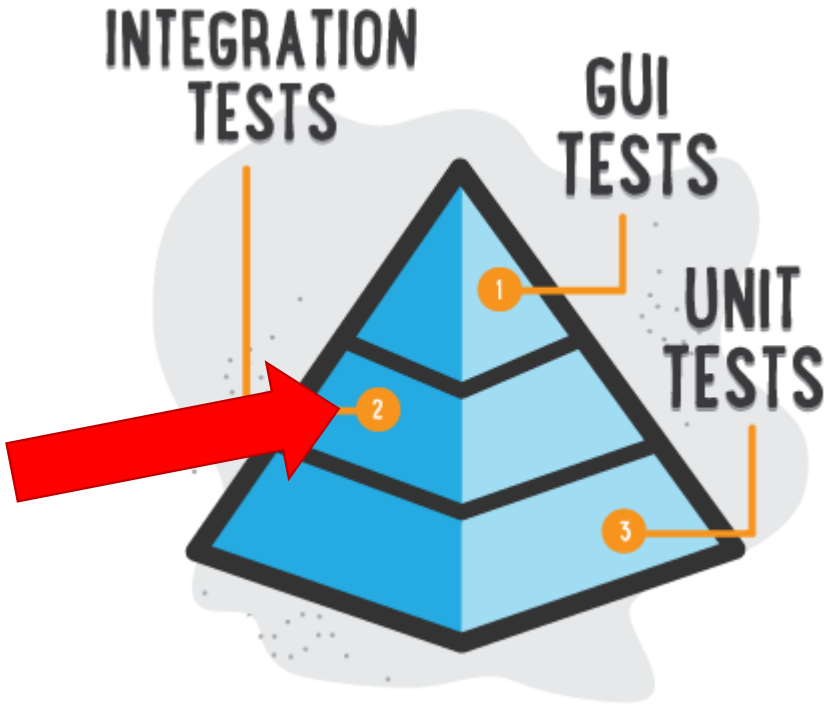
PlayerCanTakeDamage

HealthCannotBeNegative

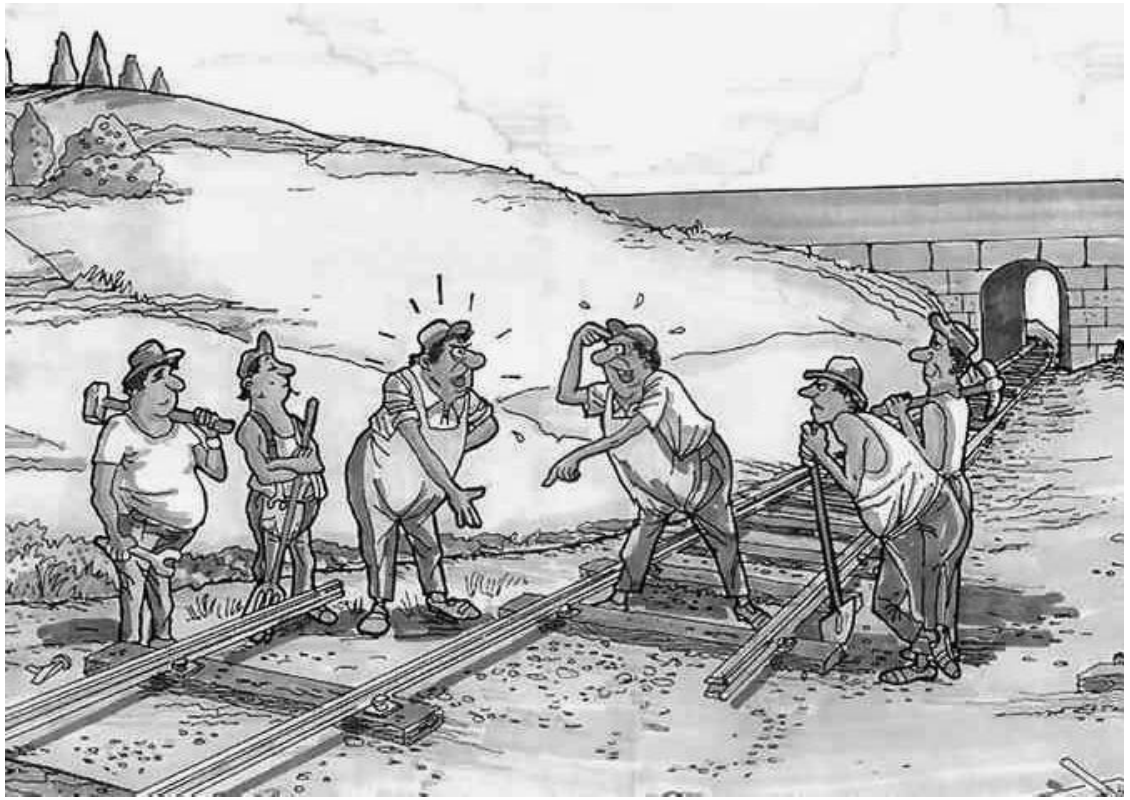
HealthCannotBeGreaterThanInitialValue

SpeedCannotBeNegative

SpeedCannotExceedMaximum



Source: http://www.seguetech.com/blog/2014/10/10/benefits-unit-testing



Integration Tests verify that **multiple systems** are connected correctly

Assets by: http://kenney.nl

PlayerStaysOnPlatform

PlayerDoesNotTakeDamageFromPlatform

PlayerMovesWithPlatform

PlayerCannotDestroyPlatform

TEST DRIVEN DEVELOPMENT BY EXAMPLE

Let‘s write some Unit Tests

STEP 1

Write a (failing) test

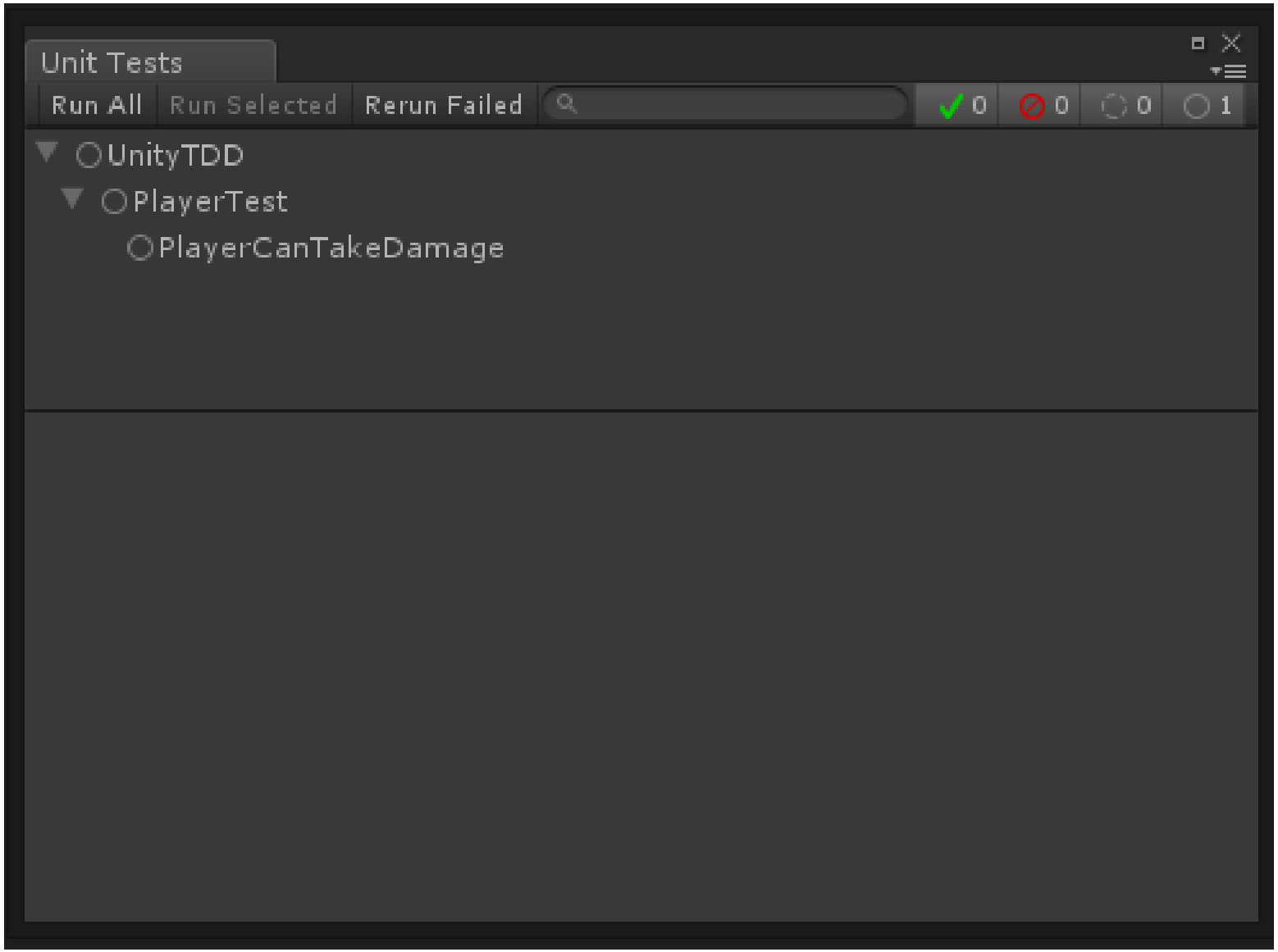
```
[TestFixture]
public class PlayerTests
{
	[Test]
	public void PlayerCanTakeDamage()
	{
		var player = new Player(50);

		player.TakeDamage(10);

		Assert.AreEqual(40, player.Health);
	}
}
```

STEP 1

Write a (failing) test

STEP 2

Check if it passes


Unit Tests
Run All
Run Selected
Rerun Failed
0
0
0
1
UnityTDD
PlayerTest
PlayerCanTakeDamage

STEP 1

Write a (failing) test

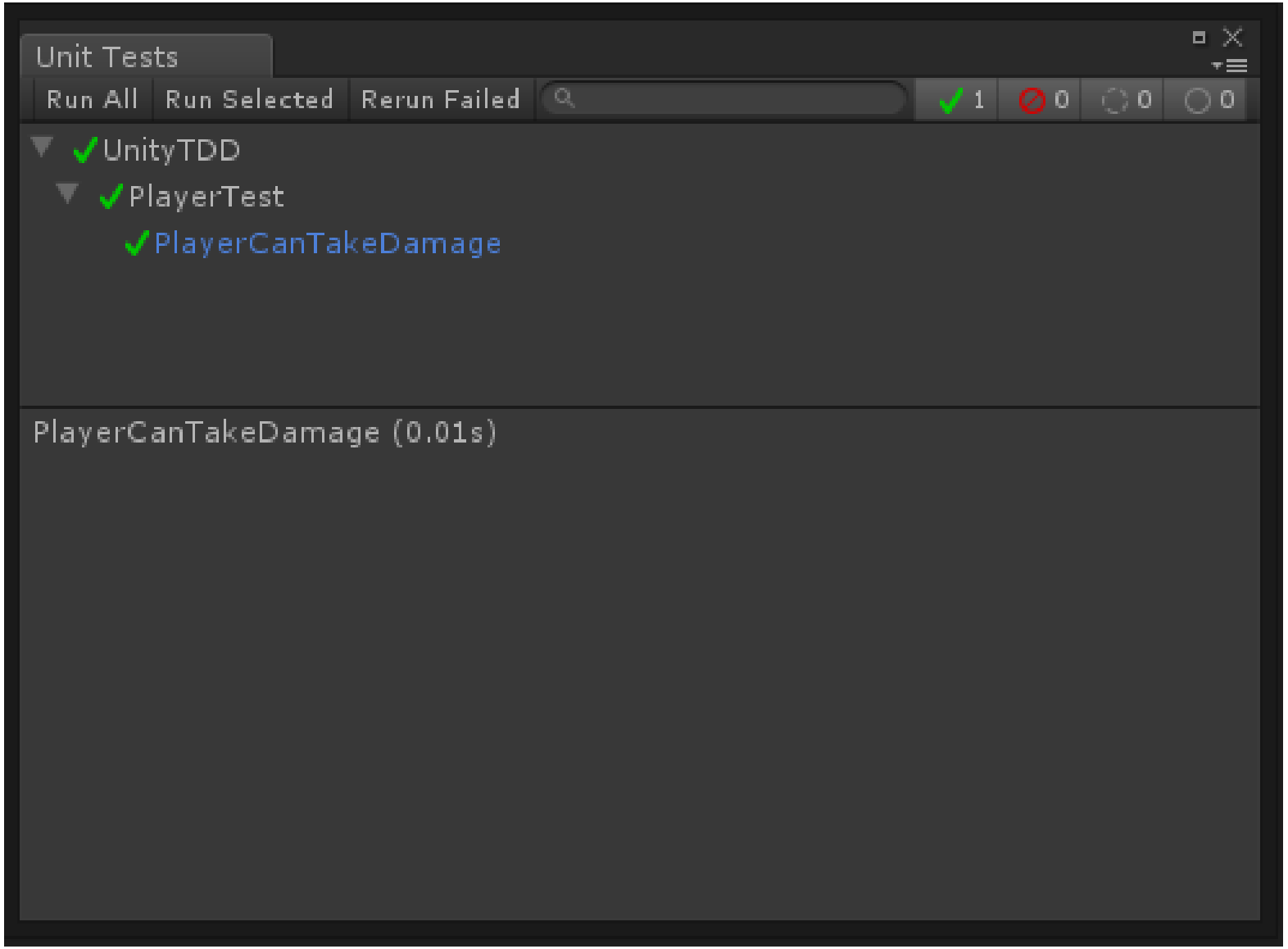
STEP 2

Check if it passes

STEP 3

Implement the actual code

```
public class Player
{
	public int Health { get; private set; }

	public Player(int initialHealth)
	{
		Health = initialHealth;
	}

	public void TakeDamage(int damage)
	{
		Health -= damage;
	}
}
```


Unit Tests
Run All
Run Selected
Rerun Failed
1
0
0
0
UnityTDD
PlayerTest
PlayerCanTakeDamage
PlayerCanTakeDamage (0.01s)

STEP 1

Write a (failing) test

STEP 2

Check if it passes

STEP 3

Implement the actual code

```
[TestFixture]
public class PlayerTests
{
        …

        [Test]
        public void PlayerHealthCannotBeNegative()
        {
                var player = new Player(50);

                player.TakeDamage(60);

                Assert.AreEqual(0, player.Health);
        }
}
```

STEP 1

Write a (failing) test

STEP 2

Check if it passes

STEP 3

Implement the actual code


Unit Tests
Run All
Run Selected
Rerun Failed
1
1
0
0
UnityTDD
PlayerTest
PlayerCanTakeDamage
PlayerHealthCannotBeNegative
PlayerHealthCannotBeNegative (0.004s)

Expected: 0
But was: -10
---
at PlayerTest.PlayerHealthCannotBeNegative () [0x00010] in D:\Dev\UnityTDD\Assets\UnitTest\Editor\PlayerTest.cs:23

STEP 1

Write a (failing) test

STEP 2

Check if it passes

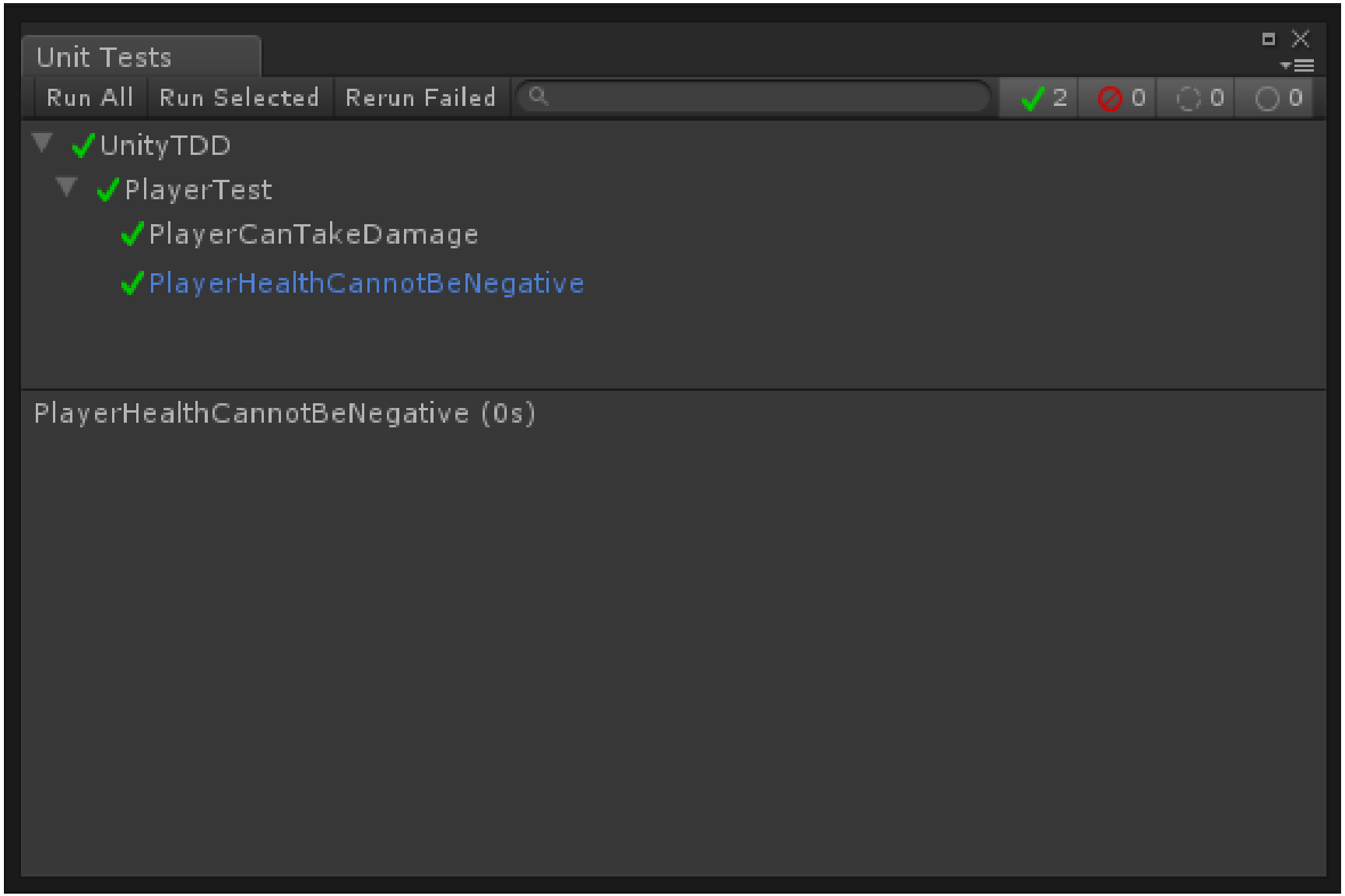
STEP 3

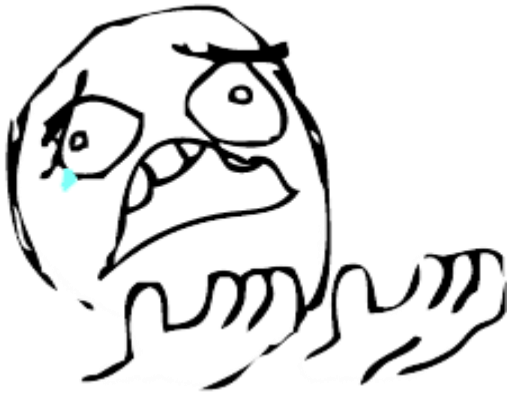
Implement the actual code

```
public class Player
{
	public int Health { get; private set; }

	public Player(int initialHealth)
	{
		Health = initialHealth;
	}

	public void TakeDamage(int damage)
	{
		Health -= damage;
	}
}
```

```
public class Player
{
        public int Health { get; private set; }

        public Player(int initialHealth)
        {
                Health = initialHealth;
        }

        public void TakeDamage(int damage)
        {
                Health -= damage;
        }
}
```

```
public class Player
{
        public int Health { get; private set; }

        public Player(int initialHealth)
        {
                Health = initialHealth;
        }

        public void TakeDamage(int damage)
        {
                Health = Math.Max(0, Health – damage);
        }
}
```


Unit Tests
Run All
Run Selected
Rerun Failed
2
0
0
0
UnityTDD
PlayerTest
PlayerCanTakeDamage
PlayerHealthCannotBeNegative
PlayerHealthCannotBeNegative (0s)



WHY?

up to

90%

decrease in defect density compared to projects which do not use TDD practises

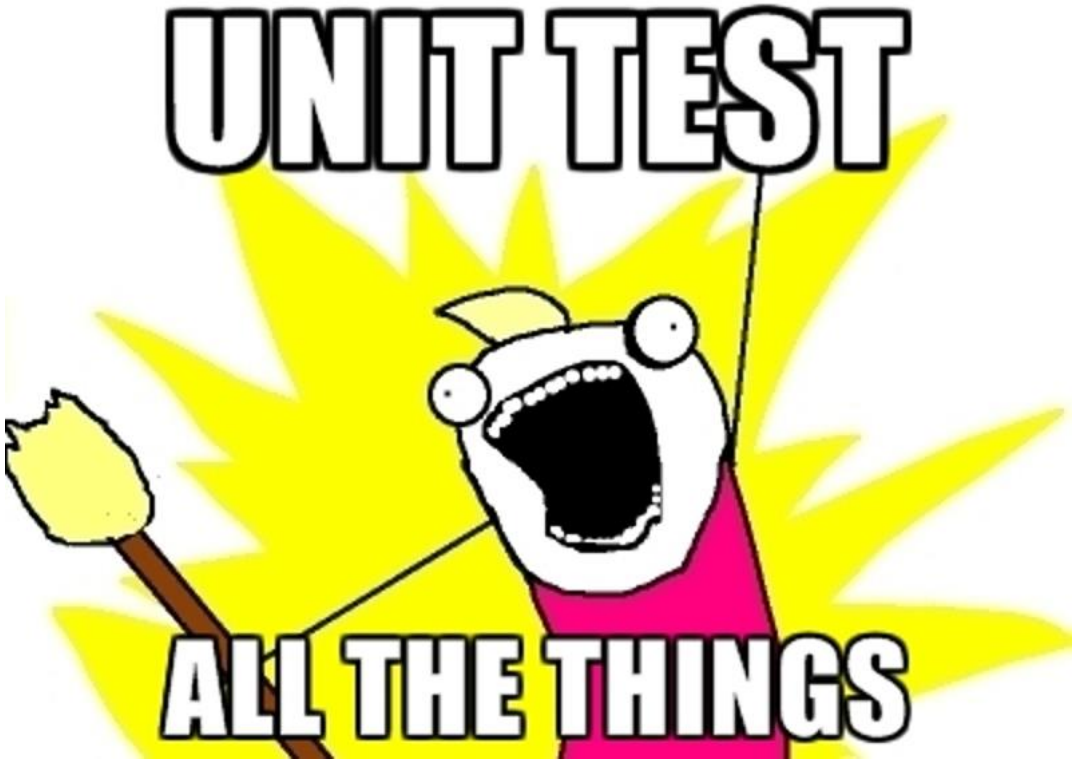
Source: *Realizing quality improvement through test driven development: results and experiences of four industrial teams*
http://research.microsoft.com/en-us/groups/ese/nagappan_tdd.pdf

up to

31%

increase in overall source code quality

Source: *Quantitatively Evaluating Test-Driven Development by Applying Object-Oriented Quality Metrics to Open Source Projects*
http://www.nomachetejuggling.com/files/tdd_thesis.pdf


UNIT TEST
ALL THE THINGS

up to

35% increase in development time

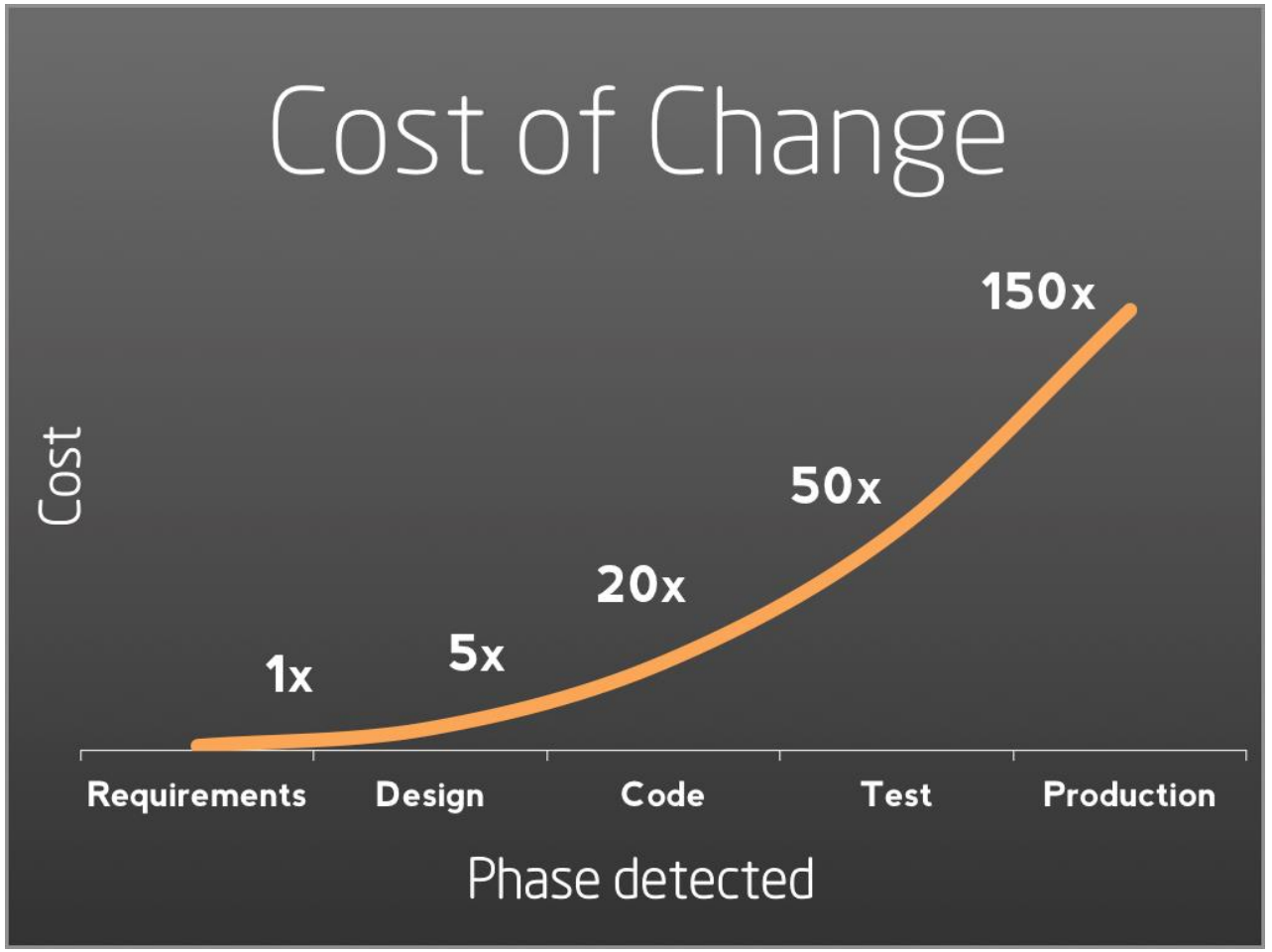
Source: *Realizing quality improvement through test driven development: results and experiences of four industrial teams*
http://research.microsoft.com/en-us/groups/ese/nagappan_tdd.pdf

Time

Quality

Cost

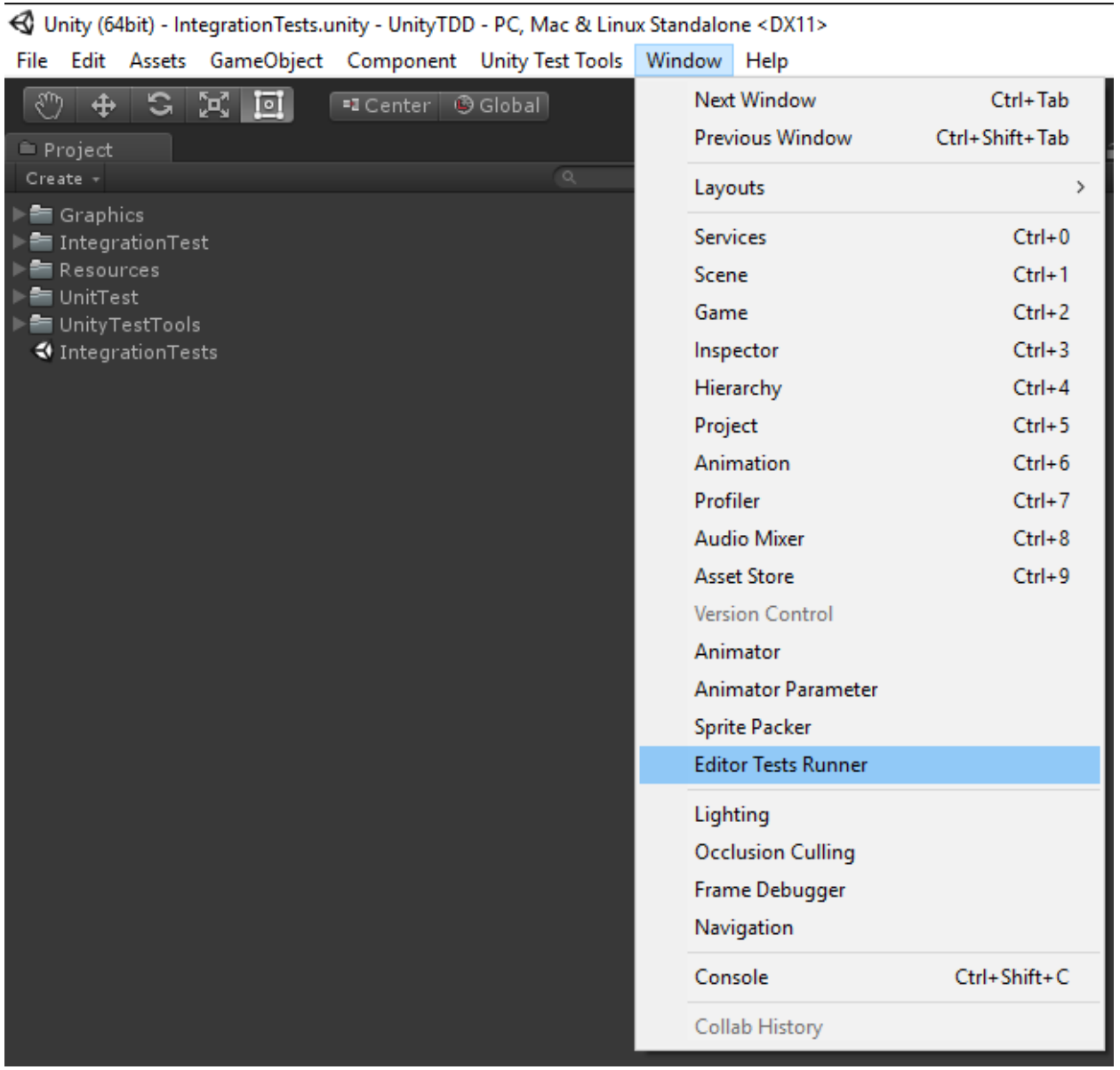
Source: https://www.flickr.com/photos/fromthevalleys-/15570115469/



Source: http://tocodeishuman.com/blog/2013/08/23/revisiting-the-cost-of-change-curve/

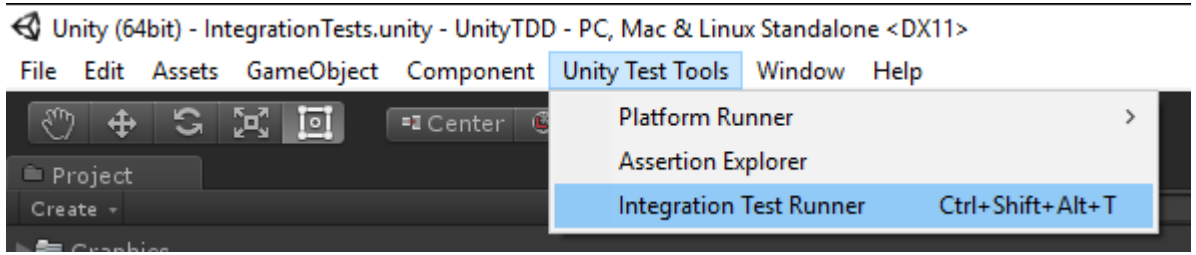
3

UNITY TEST TOOLS


Unity (64bit) - IntegrationTests.unity - UnityTDD - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Unity Test Tools Window Help
Center Global
Project
Create
Graphics
IntegrationTest
Resources
UnitTest
UnityTestTools
IntegrationTests
Next Window Ctrl+Tab
Previous Window Ctrl+Shift+Tab
Layouts
Services Ctrl+0
Scene Ctrl+1
Game Ctrl+2
Inspector Ctrl+3
Hierarchy Ctrl+4
Project Ctrl+5
Animation Ctrl+6
Profiler Ctrl+7
Audio Mixer Ctrl+8
Asset Store Ctrl+9
Version Control
Animator
Animator Parameter
Sprite Packer
Editor Tests Runner
Lighting
Occlusion Culling
Frame Debugger
Navigation
Console Ctrl+Shift+C
Collab History

Available in the Unity Asset Store: http://InnoGam.es/UnityTestTools

Integration Test Runner Menu



DEMO TIME

Let‘s create an Integration Test in Unity together

4

CONCLUSION

Test Driven Development helps you to
write better code with less errors

…but comes at the cost of
longer development cycles

Unity supports both **Unit Tests** and
Integration Tests via Unity Test Tools

Find the right balance, at least test the most critical parts of your game

INTERESTED IN MORE?

Dependency Injection

Code Coverage

Assertion

Moq

Mocking

Automated Tests

GUI Tests

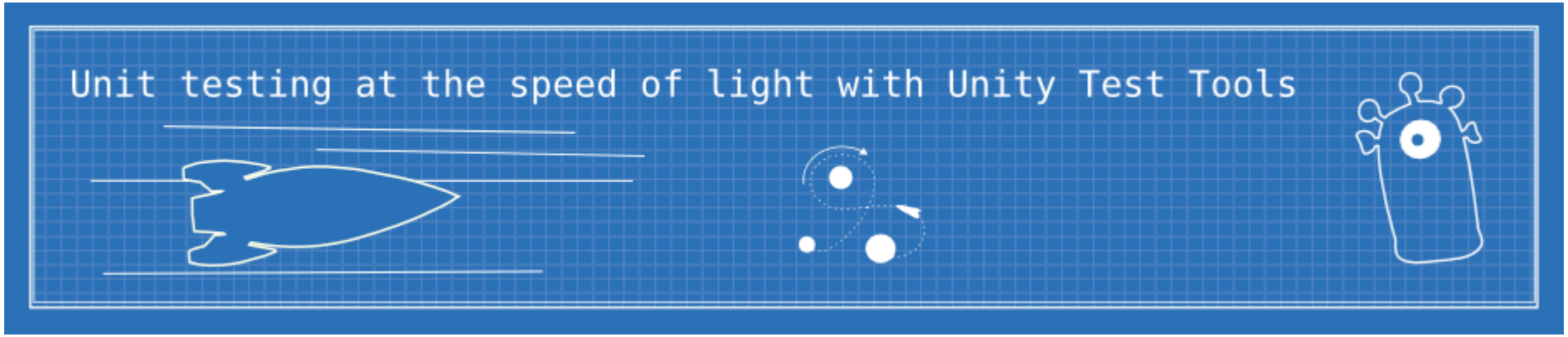
Clean Code

NUnit

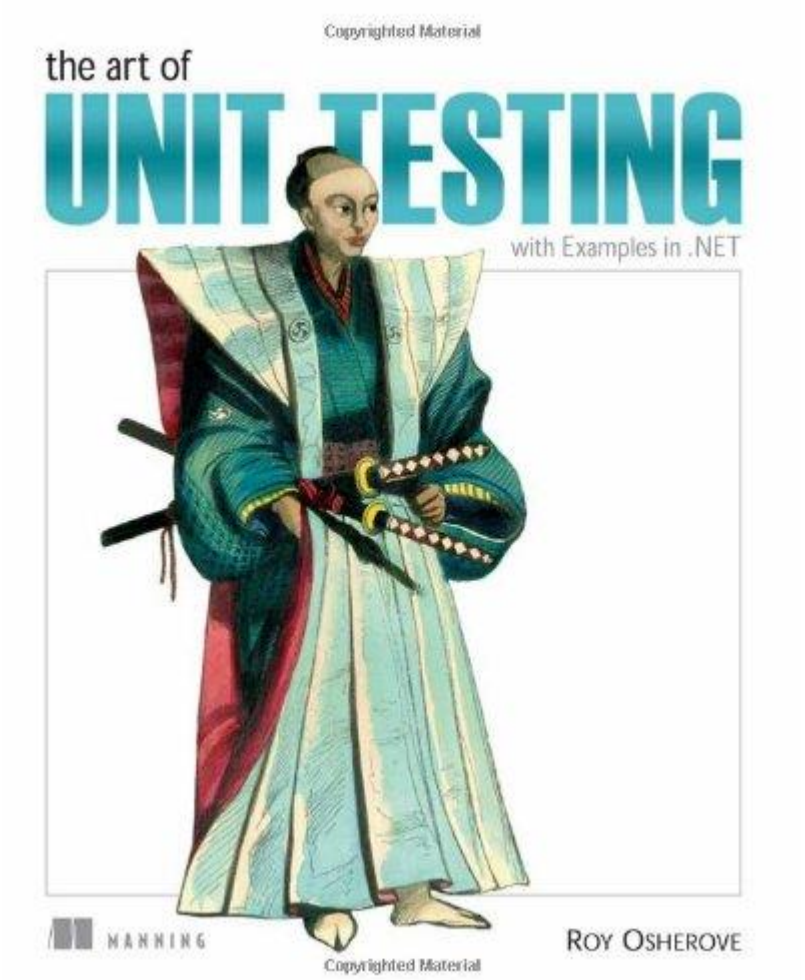
Platform Runner

Continuous Integration

Dynamic Integration Tests



http://InnoGam.es/UnityTesting



The Art of Unit Testing

by Roy Osherove

THANK YOU FOR YOUR ATTENTION!

 www.jethwa.de

 jay.allan-jethwa@innogames.com