ПРИМЕРЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ИГР В UNITY

crazy panda

DevGAMM MOSCOW 2018

ЕЛИСЕЕВ ЕВГЕНИЙ

crazy panda

Unity Developer

e.eliseev@crazypanda.ru
camohabodka@gmail.com

www.crazypanda.ru

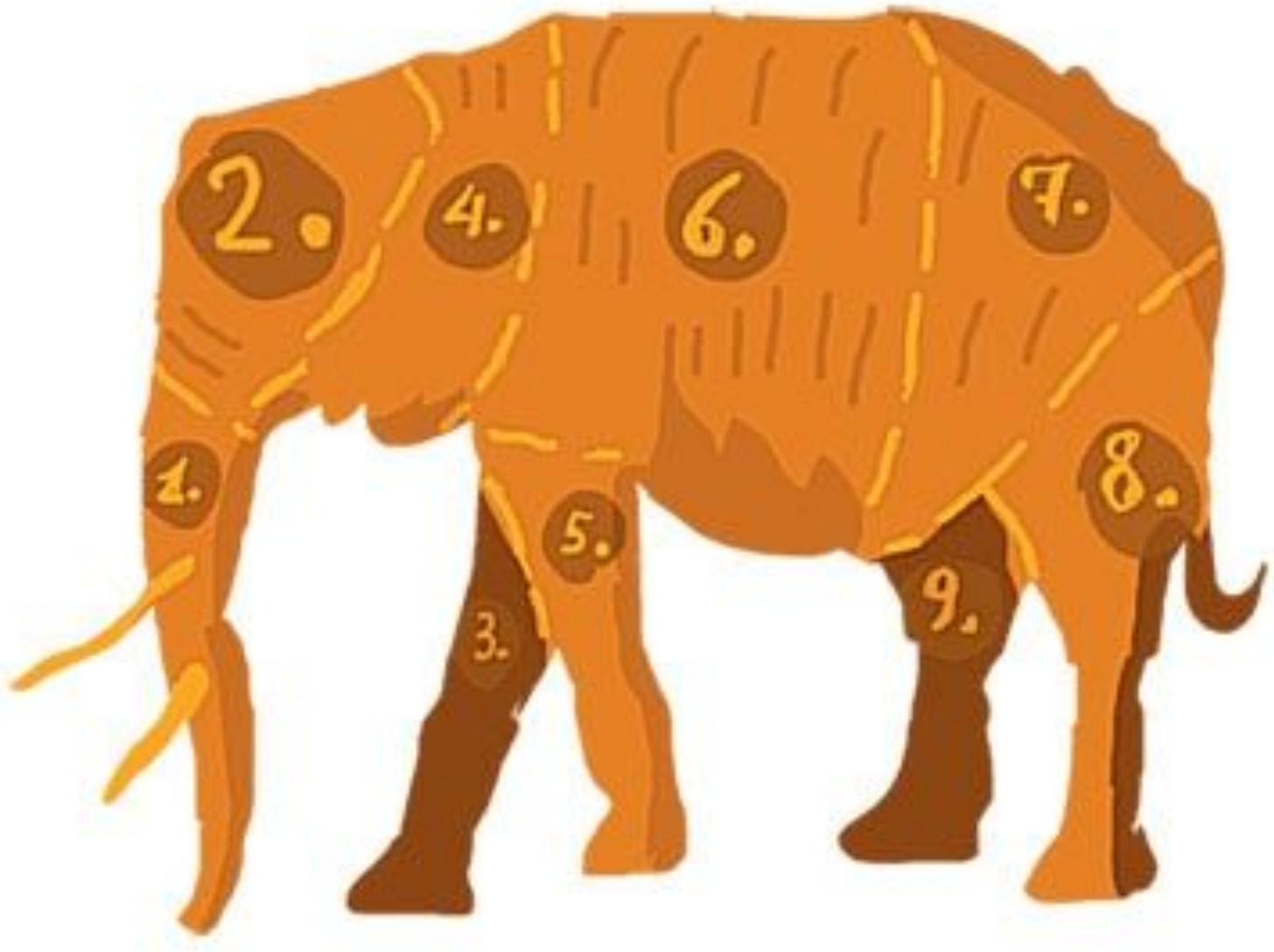
Ситуации

- Новый функционал ломает старый
  - или старый функционал бесследно исчезает
- QA перегружен
  - или QA отсутствует
- “Всё сломалось! *** ***!”
  - в мягкой форме: “Я уже заказал вам пиццу”
- "Это всегда так работало!"
  - если разобраться: "Это сломалось два месяца назад. Я написал в чатик, но мне никто не ответил"
- Команда встала, потому что основная ветка разработки разломана
  - а виновник уехал на Бали
- Неожиданно перед релизом все узнали что билд вылез за 100 мб
  - за 100 мб он вылез три месяца назад и сейчас весит 150 мб
  - маркетинг уже закупил трафик

Ситуации

- Новый функционал ломает старый
  - или старый функционал бесследно исчезает
- QA перегружен
  - или QA отсутствует
- “Всё сломалось! *** ***!”
  - в мягкой форме: “Я уже заказал вам пиццу”
- "Это всегда так работало!"
  - если разобраться: "Это сломалось два месяца назад. Я написал в чатик, но мне никто не ответил"
- Команда встала, потому что основная ветка разработки разломана
  - а виновник уехал на Бали
- Неожиданно перед релизом все узнали что билд вылез за 100 мб
  - за 100 мб он вылез три месяца назад и сейчас весит 150 мб

КАК?



ПО КУСОЧКАМ!

Четыре кусочка

- Этап 1. Непрерывная интеграция
- Этап 2. Интеграционное тестирование
- Этап 3. Тестирование скриншотами
- Этап 4. Performance-тестирование

ЭТАП 1

Непрерывная интеграция

Непрерывная интеграция. Continuous Integration

- Сделать что-то по расписанию или по какому-либо событию
  - Собрать ночную сборку и залить в HockeyApp
  - Прогнать тесты на каждый коммит в репозиторий
- Хранит билды, результаты тестов и метрики
- Множество готовых решений

Процесс

1. Коммит в репозиторий
2. Continuous Integration
   a. Unity Test Runner Edit Mode тесты (Unit-тесты)
   b. Unity Test Runner Play Mode тесты (Интеграционные тесты в редакторе)
   c. Сборка билда для каждой платформы (WebGL, Android, iOS)
   d. Unity Test Runner Play Mode тесты (Интеграционные на устройстве)
   e. Деплой (по требованию)

Результаты первого этапа. Непрерывная интеграция

- Основная ветка всегда в рабочем состоянии
- Нет ошибок в редакторе
- Билд для каждой платформы
- Знаем когда билд вышел за пределы необходимых размеров
- История из собранных билдов позволяет быстро сравнить различные билды между собой

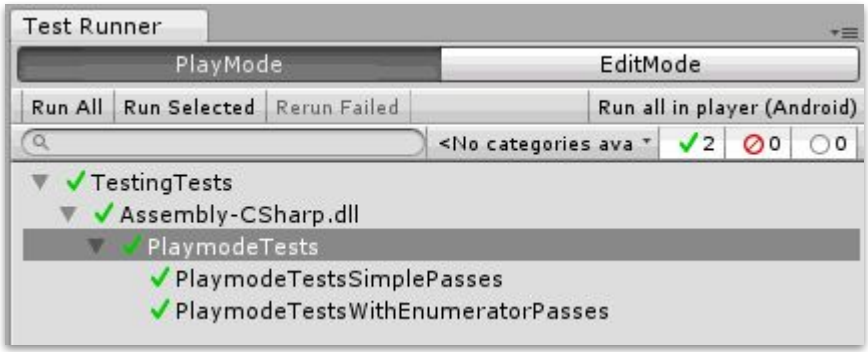
ЭТАП 2

Интеграционное тестирование

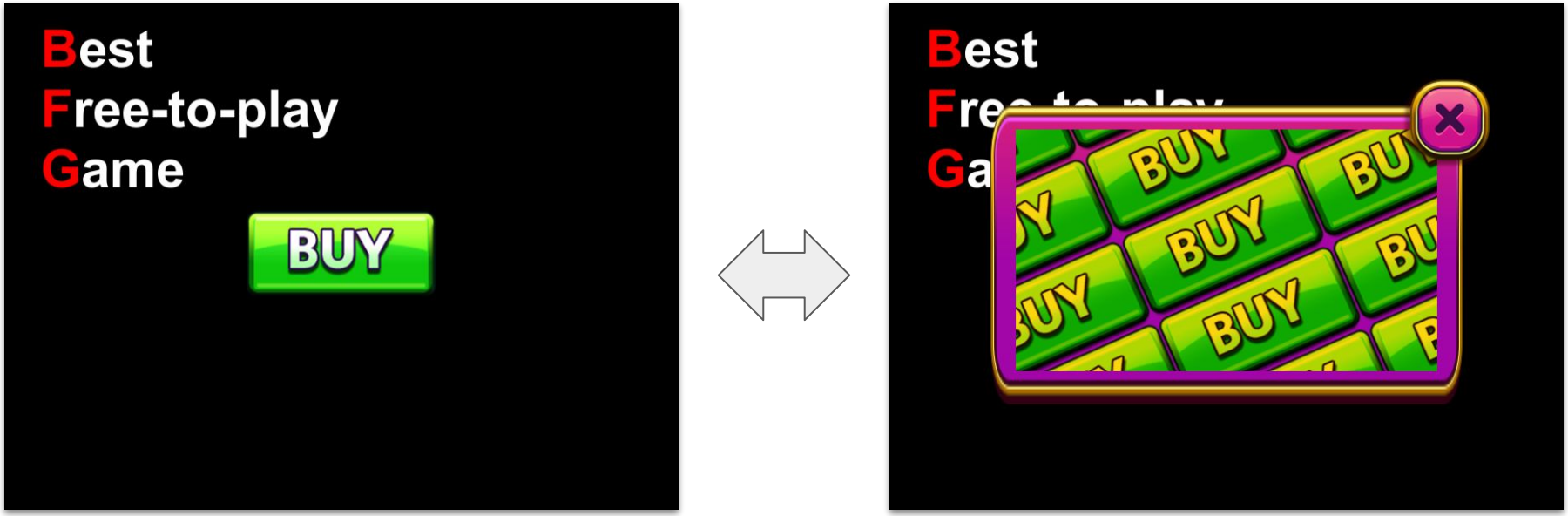
Интеграционное тестирование

- Взаимодействие модулей
- Бизнес-логика
- Можно проводить на реальных устройствах
- Дополнение к ручному тестированию и unit-тестам

Unity Test Runner



Тест-кейс. Игра BFG.



Пример теста

```
[ UnityTest ]
public IEnumerator ShopWindowTest()
{
        // -> .. создание gameProvider, viewProvider

        yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
        yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
        yield return viewProvider.WaitView< ViewShopExample >();
        yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
        yield return gameProvider.CleanUpGame();
}
```

Запуск сцены

```
[ UnityTest ]
public IEnumerator ShopWindowTest()
{
        // -> .. создание gameProvider, viewProvider

        yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
        yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
        yield return viewProvider.WaitView< ViewShopExample >();
        yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
        yield return gameProvider.CleanUpGame();
}
```

TestElement - помечаем поля

```
public class ViewShopExample : MonoBehaviour
{
        [ SerializeField ]
        [ TestElement( "Close" ) ]
        private Button _closeButton;

        [ TestElement( "OffersCount" ) ]
        private int _offersCount;
}

// использование
viewProvider.ClickButton< ViewShopExample >( "Close" )
viewProvider.GetElementFromView< ViewShopExample, int >( "OffersCount" )
viewProvider.GetElementByName< ViewShopExample, int >( "_offersCount" )
```

Нажимаем на кнопку

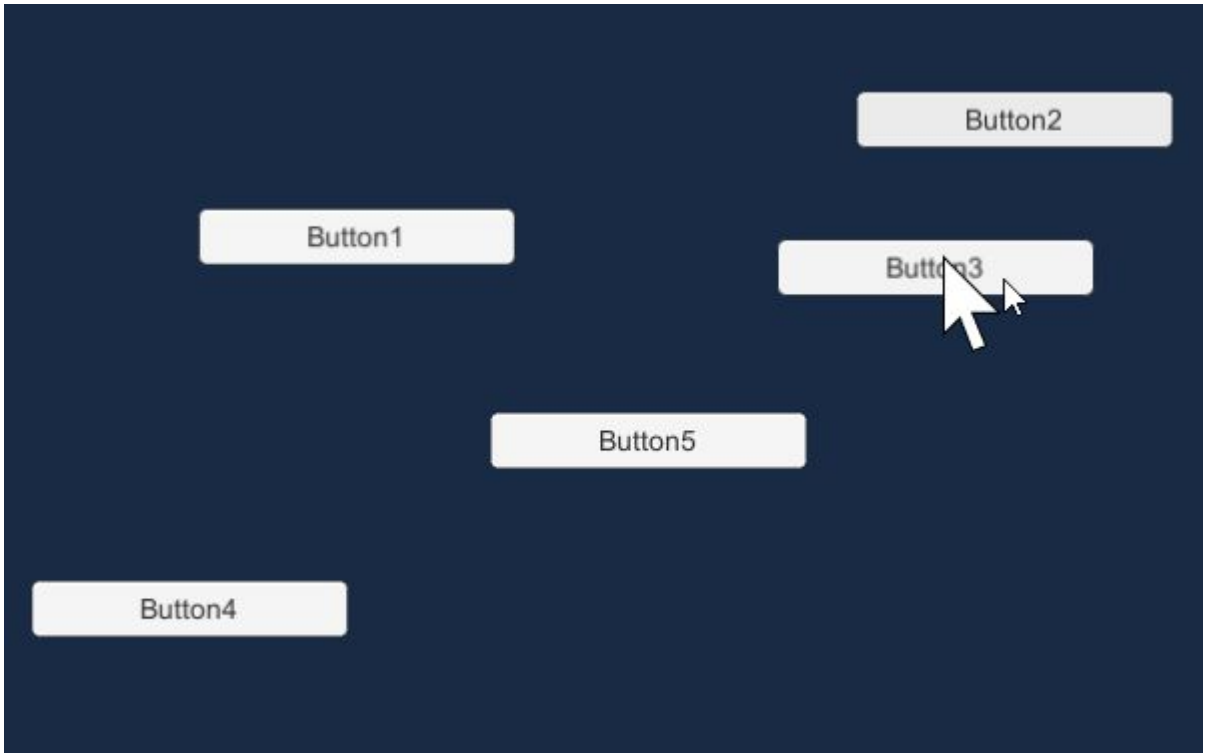
```
[ UnityTest ]
public IEnumerator ShopWindowTest()
{
        // -> .. создание gameProvider, viewProvider

        yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
        yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
        yield return viewProvider.WaitView< ViewShopExample >();
        yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
        yield return gameProvider.CleanUpGame();
}
```

Способы нажать на кнопку в Unity UI

- Внешний - API операционной системы
  - Работает не только на кнопки, но и на все приложение
  - Для каждой платформы нужно писать реализацию
- Внутренний - вызываем событие onClick
  - Работает только для кнопок Unity UI
  - Работает на всех платформах
  - Прокликивание сквозь другие UI элементы
- Внутренний - переопределение BaseInput
  - Работает для всех элементов Unity UI
  - Работает для любой платформы
  - Эмулируем курсор и касания из кода

Эмуляция курсора в Unity UI



Assert

```
yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
yield return viewProvider.WaitView< ViewShopExample >();
Assert.AreEqual( 100500, viewProvider.GetElementFromView< ViewShopExample, int >( "OffersCount" ) );
yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
yield return gameProvider.CleanUpGame();
```

Делаем скриншоты

```
GameViewUtils.SetResolution( 800, 600 );
yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
yield return viewProvider.WaitView< ViewShopExample >();
Assert.AreEqual( 100500, viewProvider.GetElementFromView< ViewShopExample, int >( "OffersCount" ) );
yield return screenshotHelper.CreateScreenshot( "Shop" );
yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
yield return gameProvider.CleanUpGame();
```

Результаты этапа 2. Интеграционное тестирование

- Запускаемость игры
- Основной игровой луп
- Дополнительная логика
- Можно определить состояние билда просто посмотрев на скриншоты

ЭТАП 3

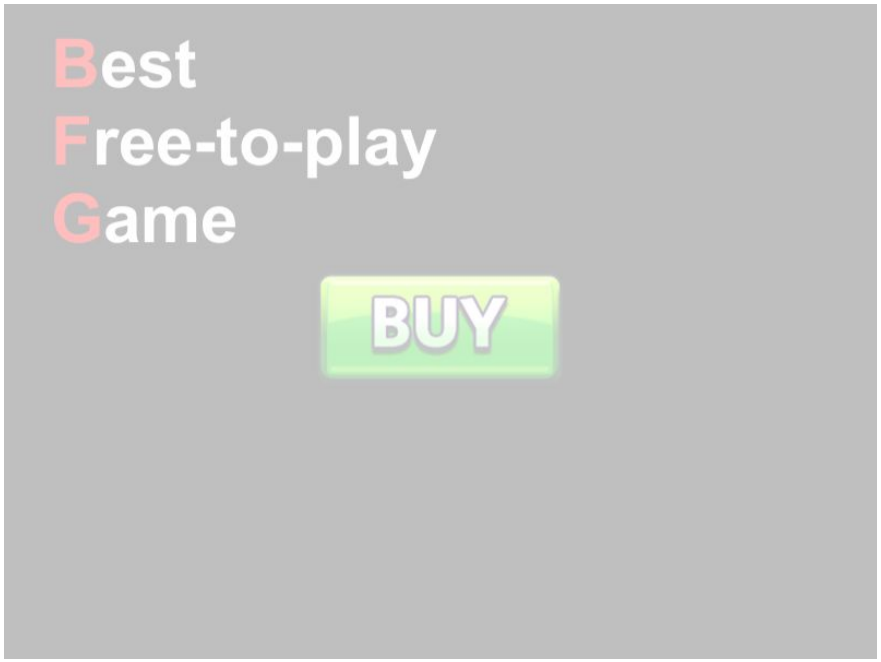
Тестирование скриншотами

Сравнение скриншотов

```
GameViewUtils.SetResolution( 800, 600 );
yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
var beforeShop = screenshotHelper.CreateScreenshot( "MainMenu" );
yield return beforeShop;
yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
yield return viewProvider.WaitView< ViewShopExample >();
Assert.AreEqual( 100500, viewProvider.GetElementFromView< ViewShopExample, int >( "OffersCount" ) );
yield return screenshotHelper.CreateScreenshot( "Shop" );
yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
var afterShop = screenshotHelper.CreateScreenshot( "MainMenu" );
yield return afterShop;
var diffImagePath = screenshotHelper.GetPath( "MainMenuDiffAfterShopOpened" );
var pixelsChanged = ImageComparer.Compare( beforeShop.Path, afterShop.Path, diffImagePath );
Assert.AreEqual( 0, pixelsChanged );
yield return gameProvider.CleanUpGame();
```

Сравнение скриншотов. Результат.

PASSED

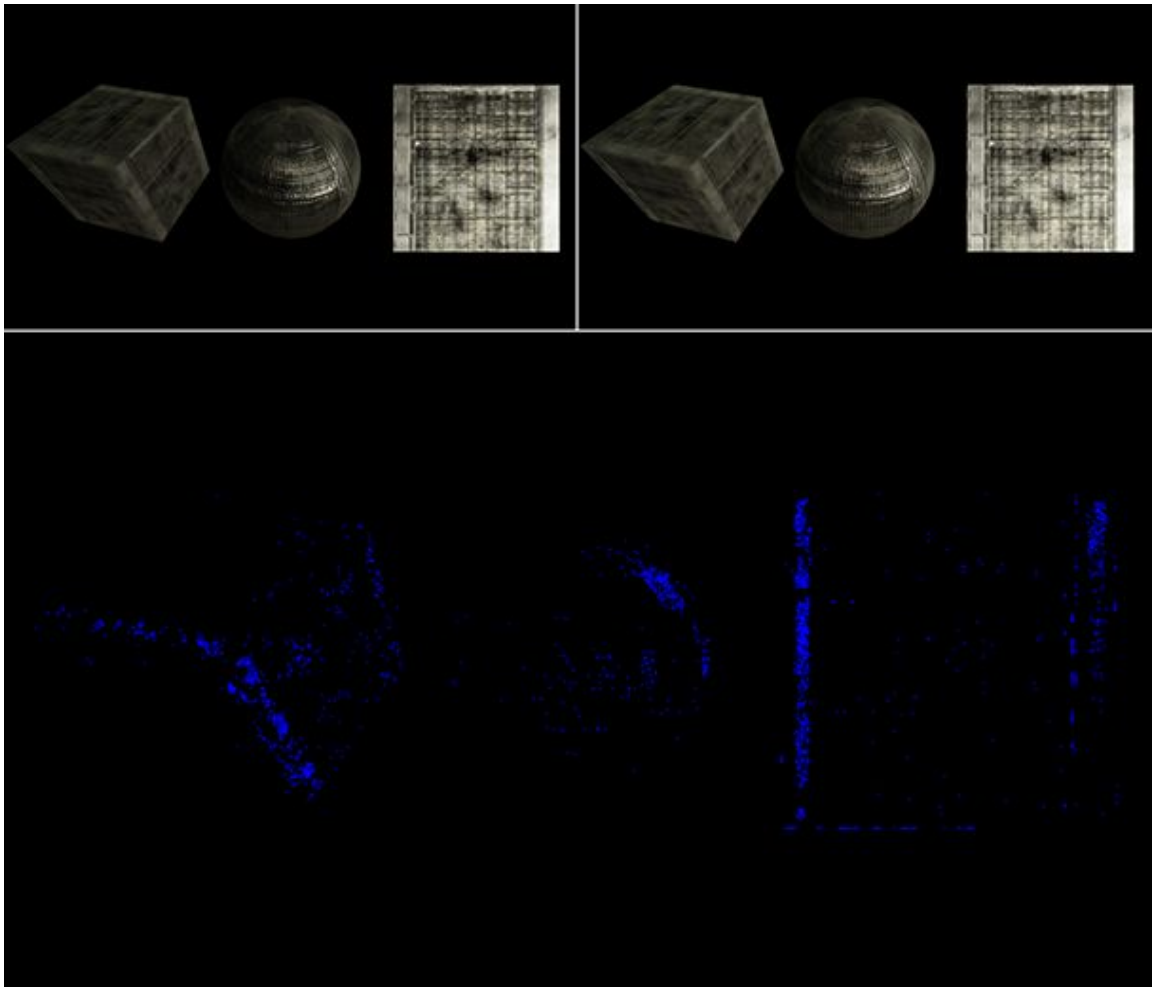


FAILED

Итоговый результат

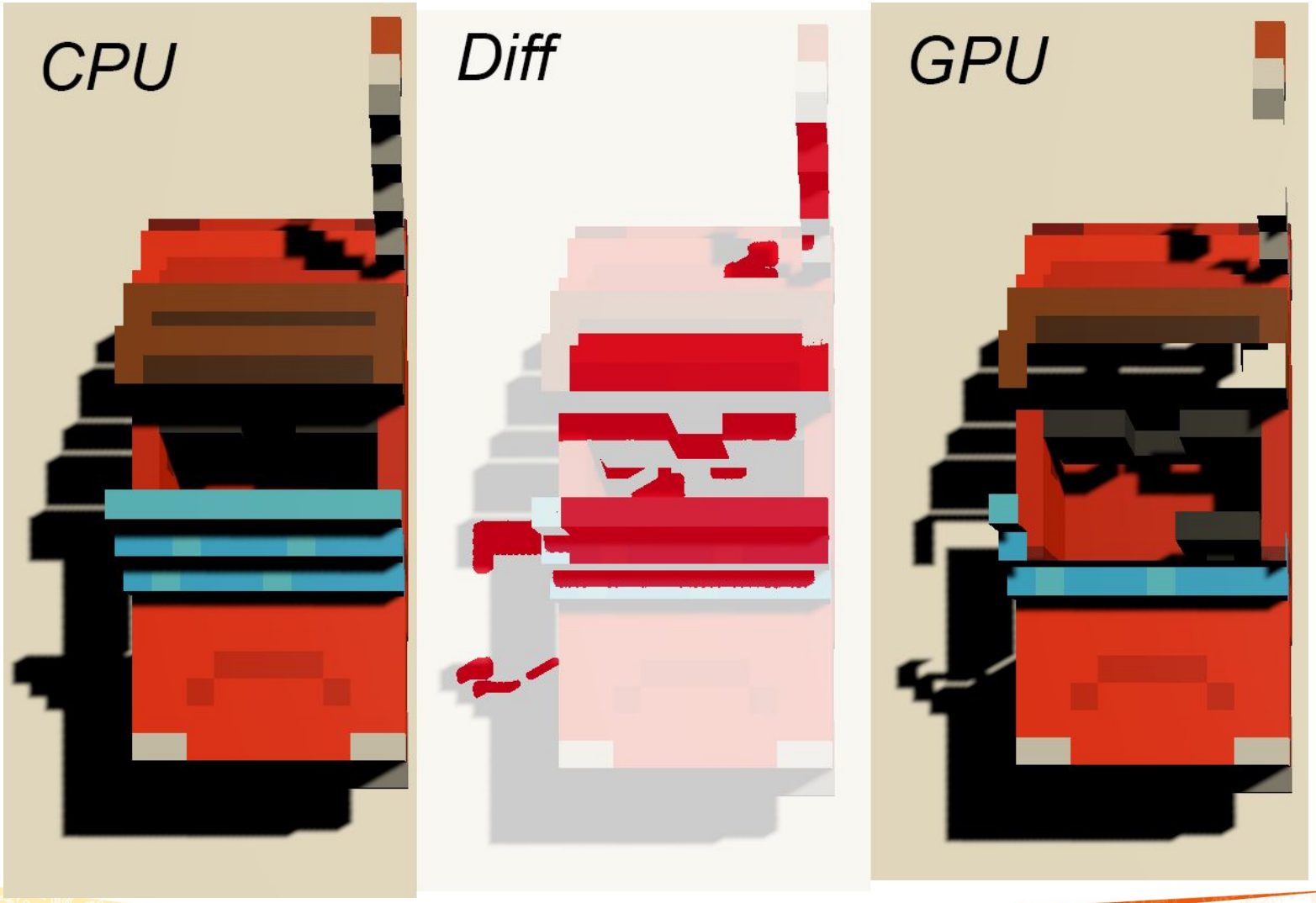
```
GameViewUtils.SetResolution( 800, 600 );
yield return gameProvider.StartWithCustomScene( "TestFrameworkExample" );
var beforeShop = screenshotHelper.CreateScreenshot( "MainMenu" );
yield return beforeShop;
yield return viewProvider.ClickButton< ViewMainMenuExample >( "Shop" );
yield return viewProvider.WaitView< ViewShopExample >();
yield return screenshotHelper.CreateScreenshot( "Shop" );
yield return viewProvider.ClickButton< ViewShopExample >( "Close" );
var afterShop = screenshotHelper.CreateScreenshot( "MainMenu" );
yield return afterShop;
Assert.AreEqual( 100500, viewProvider.GetElementFromView< ViewShopExample, int >( "OffersCount" ) );
var diffImagePath = screenshotHelper.GetPath( "MainMenuDiffAfterShopOpened" );
var pixelsChanged = ImageComparer.Compare( beforeShop.Path, afterShop.Path, diffImagePath );
Assert.AreEqual( 0, pixelsChanged );
yield return gameProvider.CleanUpGame();
```

Тестирование скриншотами. Шейдеры



Источник: https://simonschreibt.de/wft/watchdog-compare/

Тестирование скриншотами. Алгоритмы



Тестирование скриншотами. Сравнение с оригиналом

- Храним оригинал скриншота в репозитории
- Сравниваем новые скриншоты против оригинала
- Если есть изменения между новыми скриншотами и старыми
  - Решение 1 - новые скриншоты это и есть новый оригинал
  - Решение 2 - открываем баг

Результаты этапа 3. Тестирование скриншотами

- Из кода сложно понять, что игра выглядит неправильно
- Человек справляется лучше со сравниванием скриншотов, если он знает куда смотреть
- Компьютер может сравнить скриншоты и принять по этому поводу решение, либо просто показать разницу человеку, который будет принимать решение

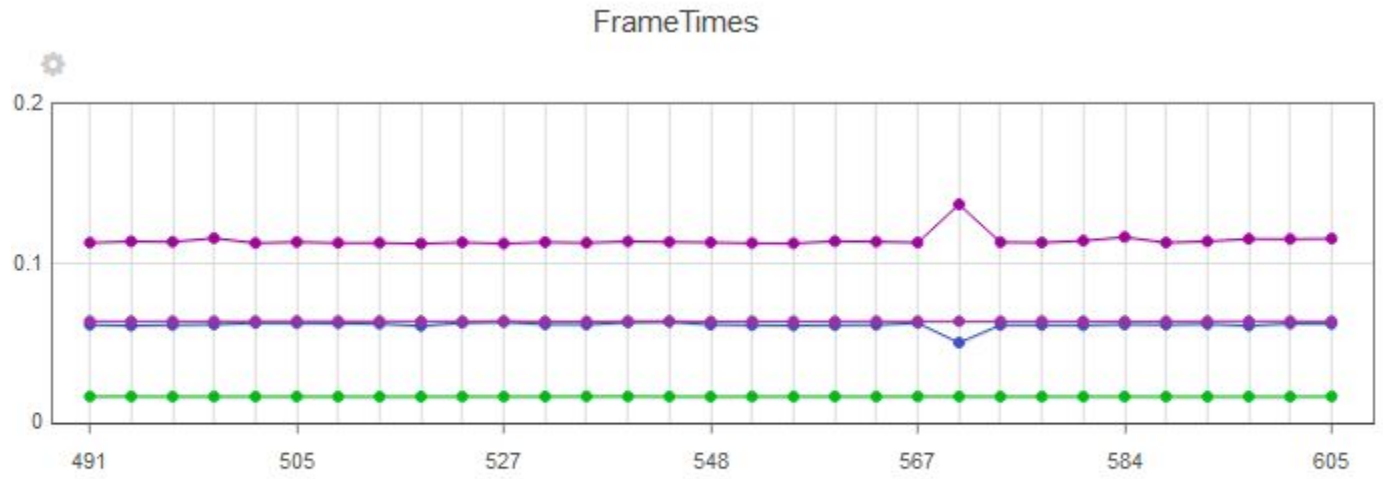
ЭТАП 4

Performance-тестирование

Performance-тестирование

- Собираем метрики во время тестов
  - Память
  - Производительность
  - Скорость загрузки
  - Миллион других параметров
- Строим график по этим метрикам

Скриншот Performance-тестирования



Результаты этапа 4. Performance-тестирование

- Известные метрики об игре
- Известен момент когда что-то пошло не так
- Проще принять решение готов билд к релизу или нет

Выводы

- Поэтапно вводим интеграционное тестирование
  - Этап 1. Непрерывная интеграция
  - Этап 2. Интеграционное тестирование
  - Этап 3. Тестирование скриншотами
  - Этап 4. Performance-тестирование
- Автоматизируем процесс тестирования
- Быстрое время реагирования, если что-то идет не так
- Экономим деньги (баг найденный раньше стоит намного дешевле)
- Увеличиваем количество Smoke-тестов (кардинально)
- Меньше неожиданностей
- Команда знает о состоянии билда
- У программиста есть страховочная сеть
- Можно осуществить на любой стадии проекта

Полезные ссылки

- https://www.gdcvault.com/play/1025013/Tools-Tutorial-Day-Tools-to (GDC, Amy Phillips, Tools to Reduce Open Bug Count at Media Molecule)
- http://gdcvault.com/play/1022784/Fast-Iteration-Tools-in-the (GDC, Alen Ladavac, Fast Iteration Tools in the Production of the Talos Principle)
- https://www.youtube.com/watch?v=ff5LNHGBGoM (DataArt, Валентин Анопренко, Интеграционные автотесты бизнес-логики)
- https://simonschreibt.de/wft/watchdog/ (Simon Schreibt, Using screenshot comparing techniques)
- https://www.youtube.com/watch?v=ULwdj_Vr_WA (HolyJS, Роман Дворнов, Unit-тестирование скриншотами: преодолеваем звуковой барьер)
- https://www.youtube.com/watch?v=LEy3_2ZzWpk (DotNext, Андрей Акиньшин, Поговорим про Performance-тестирование)

Бонус 1. Улучшаем плей-тесты

- Увеличиваем Time.timeScale (не применимо для всех тестов/приложений)
- Не должны зависеть от рандома
- Узкая функциональность
- Нам не важна производительность кода теста (поэтому мы активно используем рефлексию в тестах)
- Monkey Runner
- Использовать функционал ботов и/или реплеев

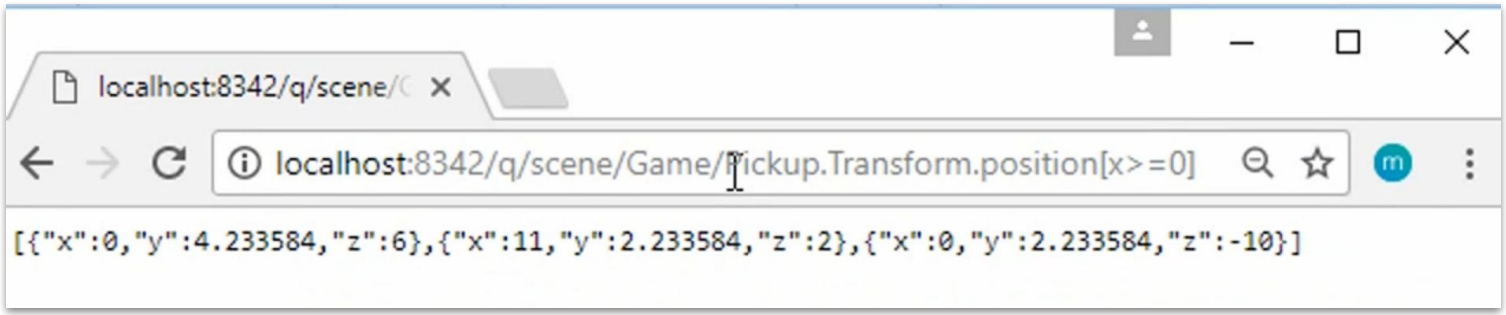
Бонус 2. Что хотелось бы видеть в Unity

- Возможность экспортировать результаты тестов с устройства
- Возможность фильтровать тесты при запуске из консоли
- Возможность фильтровать тесты при запуске на устройстве
- Play-тесты в Unity Cloud Build
- Code Coverage

Бонус 3. Альтернативы. SikuliX

Источник: http://sikulix.com/

Бонус 4. Альтернативы. Unium




```
var url = "ws://localhost:8342/ws";

Log( "Connecting to " + url );
yield* Connect( url );

Log( "Connected to game, checking we are in the tutorial scene" );
var about = (yield* Get( "about", "/about" ));
if( !about || !about.data || about.data.Scene != 'Tutorial' ) {
  throw new Error( "Not running the tutorial scene" );
}

Log( "Reload scene" );
yield* Get( "sceneLoaded", "/bind/events.sceneLoaded" );
yield* Get( "reload", "/utils/scene/Tutorial" );
```

Источник: https://assetstore.unity.com/packages/tools/unium-automated-test-tools-95998

Вопросы

ЕЛИСЕЕВ ЕВГЕНИЙ

crazy panda

Unity Developer

e.eliseev@crazypanda.ru
camohabodka@gmail.com

www.crazypanda.ru