

# ТЕСТОВОЕ ЗАДАНИЕ

Unity Developer

## ПРИВЕТ!

Спасибо за интерес к нашей компании.

Мы создаем мобильные приложения с использованием технологии Augmented Reality, в которых волшебный детский мир с персонажами - оживает.

В этом тестовом задании вы сможете продемонстрировать свои технические навыки и изобретательность.



LIVE  
animations

## Основная технология разработки

Для реализации программы выбрана технология Unity3D. Unity3D - это инструмент для разработки двух и трёхмерных приложений, работающий под операционными системами Windows, Linux и OS X. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One и MotionParallax3D.

C# используется как основной язык разработки. Компилятор, который использует Unity3D, поддерживает практически все возможности языка C#, за исключением отдельных особенностей .Net, которые еще не отражены в текущей версии интегрированной среды разработки MonoDevelop.

### Используемые принципы программирования и Архитектура

SOLID (сокр. от англ. single responsibility, open-closed, Liskov substitution, interface segregation и dependency inversion) 5 важных принципов: Принцип единственной ответственности (Single responsibility), Принцип открытости/ закрытости (Open-closed), Принцип подстановки Барбары Лисков (Liskov substitution), Принцип разделения интерфейса (Interface segregation), Принцип инверсии зависимостей (Dependency Inversion)

IoC (окр. от англ. Inversion of Control, ) — важный принцип, используемый для уменьшения зацепления в компьютерных программах. Также архитектурное решение интеграции, упрощающее расширение возможностей системы, при котором контроль над потоком управления программы остаётся за каркасом.

KISS — (окр. от англ. Keep It Simple, Stupid / Keep it short and simple / keep it small and simple) — не усложняй! Смысл этого принципа программирования заключается в том, что стоит делать максимально простую и понятную архитектуру, применять шаблоны проектирования и не изобретать велосипед. Поэтому в области проектирования простота должна быть одной из ключевых целей, и следует избегать ненужной сложности.

Для архитектуры приложения необходимо использовать MVC - модель. Данная модель архитектуры одна из наиболее подходящих для приложений такого рода. Она позволяет отделить логику от визуализации и контроля. Архитектура позволяет довольно удобно распределить процесс создания продукта между разработчиками, осуществлять поддержку и разработку новых версий и имеет потенциал для создания кода с хорошей читаемостью.

Для реализации всех этих принципов и архитектуры мы будем использовать StrangeIoC.

StrangeIoC - это MVC-фреймворк для Unity на языке C#. По сути, является адаптацией фреймворка Robotlegs (используемого при Flash-разработке) для Unity на C#. Что является дополнительным преимуществом. Также он имеет удобный механизм инъекций и обмена событиями через общий контекст. Фреймворк реализует принцип инверсии контроля. Этот принцип позволяет разрабатывать архитектуру с большой автономностью отдельных компонентов, и минимизировать зависимости между ними.

# Задание

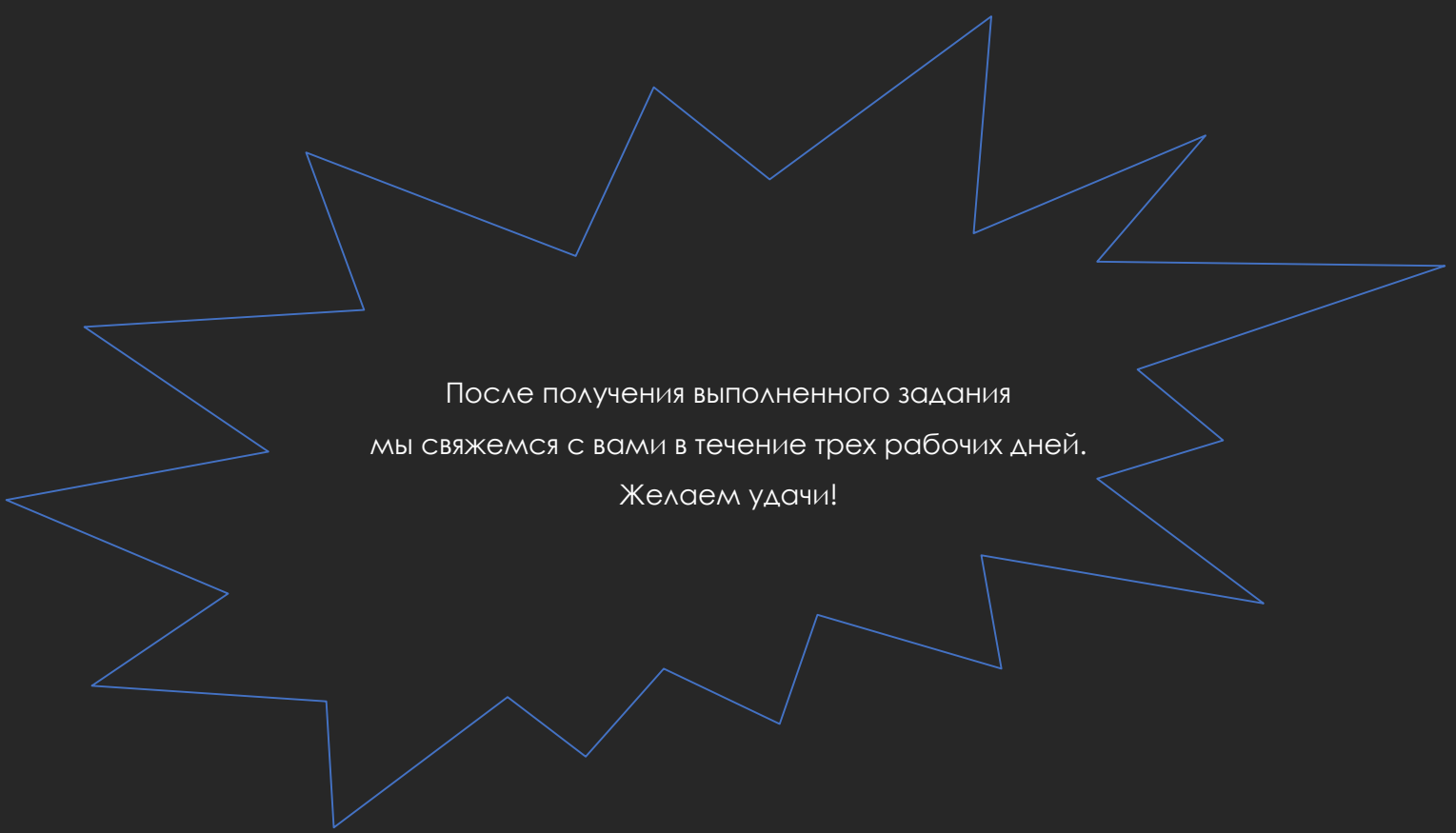
1. Скачать StrangeIoC: <https://github.com/AndreySkyFoxSidorov/strangeioc.git>
2. Изучить работу StrangeIoC, в этом поможет Youtube и Офф сайт <http://strangeioc.github.io/strangeioc/>  
(Нам важно чтобы вы могли самостоятельно изучать и разбираться в новых технологиях и фреймворках)
3. Реализовать игру Тетрис используя StrangeIoC и Unity "input.touches" интерфейс. Сторонние плагины кроме StrangeIoC использовать запрещено. Ресурсы для игры можно скачать по ссылке:

<https://goo.gl/gGwDx1>

в ресурсах есть видео ролик примера работы тетриса "Assets\Art\EXAMPLE.mp4"

(Игра тетрис взята потому что есть много примеров реализации ее функционала, нам важно чтобы вы могли качественно написать эту простенькую игру и она работала на IOS или Android )

4. Оценивать качество написания и ваш скилл, мы будем по правильности написанного кода, умения использования паттернов проектирования, по пониманию Model-View-Controller подхода. Для этого нужно выполненное задание залить на Github или Bitbucket. Важно чтобы мы видели историю коммитов и понимали поэтапность написания тестового задания.



После получения выполненного задания  
мы свяжемся с вами в течение трех рабочих дней.  
Желаем удачи!