Описание: данное тестовое задание направлено на определение и оценку уровня знаний претендента на должность клиентского программиста мобильных приложений. Оцениваются навыки по работе на языке программирования C#, .NET и навыки программирования под Unity3D. На выбор дается 3 варианта сложности задания. Каждый вариант является усложнением основного (вариант **I**) варианта и предполагает демонстрацию дополнительных навыков и знаний, что отражается на уровне дальнейшей заработной платы и увеличивает шансы на положительное решение. Претендентам на должность лида, а так же команде разработчиков необходимо выполнять вариант сложности **III (IV для команды),** а так же в письменной форме (либо устно подготовить аргументированную точку зрения) ответить на следующий вопрос: какую систему контроля версий и почему вы бы выбрали, если бы велась разработка клиент-серверного приложения в команде порядка 10-и человек, с использованием более чем одного языка программирования или скриптовой подсистемой?

Оценке подлежит:

1) степень соответствия тестового задания поставленным условиям;
2) качество кода (простота и прозрачность, масштабируемость, надежность, читаемость, уровень ООП);
3) соответствие дате выполнения поставленным срокам;

Суть задания:

Вариант сложности I:

Реализовать простую игру «Кружочки» с использованием Unity3D (версия 4+). Целью игрока является набор очков за счет «лопанья» кружочков двигающихся сверху вниз экрана. «Лопанье» происходит простым нажатием на экран в зону круга, после чего он пропадает. Кружки двигаются без ускорения, с постоянной скоростью, которая зависит от размера кружка (чем меньше он - тем быстрее падает). За «лопанье» каждого кружочка дается определенное кол-во очков, которое так же зависит от его размера (чем меньше он – тем больше очков). Таким образом, маленькие кружочки игроку «лопать» тяжелее, но очков за них дается больше. Скорость и размер кружочка генерируется при его создании. Размер генерируется случайным образом в каком-то промежутке. На основе размера уже формируется скорость и кол-во даваемых за «лопанье» кружочка очков.

Игровой экран представлен фоном на усмотрение претендента, в левом нижнем углу счетчик очков и таймер. Кружки начинают падать сверху-вниз сразу после старта приложения (т.е. интерфейс старта или окончания игры не обязателен, но может быть сделан по желанию), при этом падают они не с какой-то конкретной точки, а появляются случайным образом вдоль всей верхней границы экрана, но с условием, что круг полностью вмещается в экран. При достижении нижней границы экрана круги сами пропадают, не давая очков. Так же необходимо, чтоб круги генерировались

случайного цвета и со временем их появлялось все больше, и базовые скорости увеличивались (как в тетрисе, т.е. имеет место повышение уровня сложности).

Важным условием является необходимость получения игрового контента из бандлов. Сами шарики, эффекты, возможные звуки/музыка для игры должны выкачиваться из сети в бандлах (Unity bundles) с использованием класса WWW. Допустимо выкачивание из папки в директории игры, т.к. сути это не меняет.

Логического завершения игра не имеет, т.е. условия проигрыша и выигрыша не установлены. Скорость генерации, скорость падения, размеры и спектр цветов кругов остаются на усмотрение претендента.

Вариант сложности II:

Суть задания остается той же. Разница состоит в том, что кружки должны содержать генерируемые в режиме реального времени текстуры. Генерации подлежат размер и цвет текстуры (необходимый минимум - заливка. Градиент или рисунок - по желанию, будет зачтен как плюс к заданию). Допустимые разрешения 32x32, 64x64, 128x128 и 256x256. Текстуры большего разрешения должны присваиваться кружкам большего размера и наоборот. Основная сложность заключается в том, что текстуры должны не только динамически создаваться, но и корректно отгружаться, освобождая память, когда становятся ненужными т.е. необходимо реализовать менеджер ресурсов, который будет последовательно генерировать текстуры, использовать в кружках определенные цвета и размеры, после чего удалять их и создавать новые сеты текстур взамен старых. Создавать текстуры стоит постепенно прямо во время игрового уровня, но в момент смены уровня сложности на следующий реализовывать подмену одного сета на другой, с отгрузкой предыдущего. Таким образом будут минимизированы возможные тормоза.

Вариант сложности III:

Суть задания остается той же что и вариант сложности **II**. Разница состоит в том, что приложение необходимо организовать таким образом, чтобы была возможность из одного клиента не только начинать новую игру, но и подключаться по определенному айпи и порту к уже существующей игре для просмотра того, как играет другой игрок. Важным нюансом является запрет на использование существующей в юнити сетевой подсистемы (класс Network). Т.е. необходимо продемонстрировать знания организации сокетного соединения с использованием только .NET библиотеки.

Вариант сложности IV (групповой):

Групповой вариант подразумевает работу над тестовым заданием группы претендентов и является максимальным по своей сложности. Включает в себя вариант сложности **III**, с усложнением сетевого взаимодействия и геймплея. Игра должна начинаться только после того, как двое игроков подключатся друг к другу и подтвердят нажатием на кнопку свою готовность к игре. Так же в данном варианте

подразумевается возможность ввести свой ник и вводится понятие победы и поражения в игре, т.е. игроку дается определенное кол-во кружков, которые можно пропустить не кликнув по ним. Игровой экран должен быть разделен на две равные половины. В одной части игрок видит свою игру, свой ник, свои очки и кол-во оставшихся пропусков кружков, во второй - аналогичные данные соперника. Когда один из игроков пропускает заданное кол-во кружков, игра останавливается, каждому игроку выводится сообщение о победе, либо поражении и меню с возможностью начать заново игру с этим соперником, либо подключиться к другому игроку.

Результат тестового задания:

В качестве результата необходимо предоставить весь исходный код приложения с проектом Unity3D, компилируемый без установки дополнительных библиотек и сторонних решений, в том числе написанных заранее математических библиотек и т.п. Самым предпочтительным является код написанный “с нуля”, т.к. максимально точно показывает именно текущий уровень. Большим плюсом будет демонстрация навыков владения одной из систем контроля версий, т.е. организация хранения кода в одном из публичных git или svn репозиториев с предоставлением возможности просмотра истории коммитов при передаче тестового задания.

Дополнительно:

Наличие комментариев в коде обязательно. Большим плюсом будет внесение разнообразия в игру, за счет внедрения эффектов и различных визуальных примочек. Приложение должно быть организовано максимально оптимально с точки зрения потребления ресурсов, оперативная память не должна утекать в процессе игры.