Если ребёнок пришёл с нуля, ему придётся первым делом освоить средства разработки скриптов, иначе игру он не напишет. Потому данное занятие является обязательным  после выполнения прошлого урока для прохождения всем, когда бы они ни пришли заниматься.

**На это занятие стоит взять флешку с хорошим свободным объёмом памяти. Начиная с прошлого занятия все скрипты надо обязательно сохранять хоть где-то, так как на этих конструкциях потом будут строиться проекты. Если ребёнок не выполнит все задания, на следующем уроке ему предстоит доделывать. Если же он не выполнил прошлый урок хотя бы до задания 5, ему следует продолжить работу пока пятое задание не будет выполнено.**

С этого урока простейшие типичные ошибки из серии забытых скобочек, лишних запятых и так далее дети ловят сами. Помогаем только если совсем никак. Также пусть смотрят логи ошибок - ученики вполне способны скопировать и вбить в гугл-перевод текст ошибки и осознать то, что хочет машина. При сложной ошибке пусть хотя бы говорят, на какую строку ругается компилятор. Поощряем комментарии к коду.

Занятие 3.

Заявляем, что сегодня создаём некоторое окружение для своей игры. На прошлом уроке мы сделали (возможно, частично) управление для нашего объекта. Позже будет возможность его модифицировать. “Сегодня будем делать минимальное окружение чтобы было обо что биться лбом и щепотка триггеров”. Триггер - действие, происходящее при каком-то событии. Например, персонаж встал в определённой области или перешёл какую-то черту на карте. После этого срабатывает ловушка.

Первое задание даём раньше самого юнити: дети должны придумать простейшие объекты для взаимодействий. В будущем они смогут применить это в проекте, так что пусть думают наперёд. Сегодня, конечно, придётся делать по заданию, но чуть позже у ребят будет возможность разгуляться.

Открываем наш проект с управлением. Кому надо - пусть доделывает прошлое задание. Отстающим можно поставить в помощники тех, что успевают больше нормы. За peer-to-peer (система обучения “от человека к человеку”, то есть сотрудничество) стоит поощрять. Условие: не создаётся медвежья услуга - свой код каждый пишет и понимает сам, помогать можно только подправляя в трудных местах и помогая понять алгоритм работы той или иной связки.

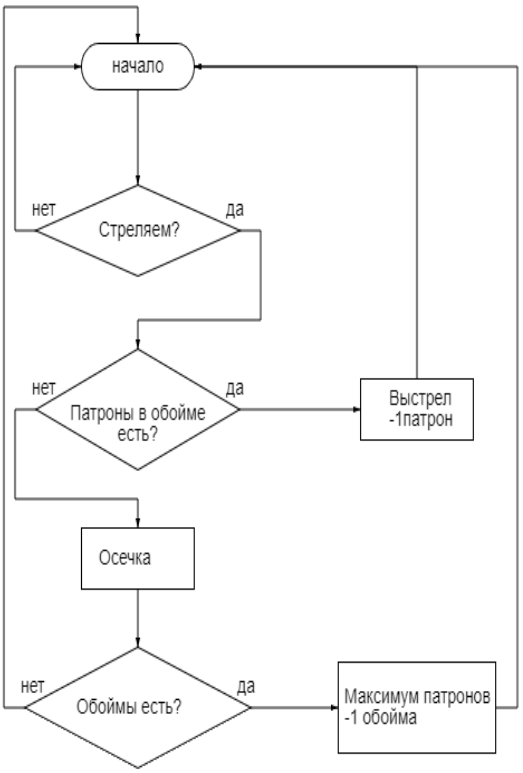
В уже созданный проект надо добавить новые картинки: button, door и wall.  Последний спрайт сразу выносим на экран. Отмечаем, что у нас многовато картинок и их стоит вывести в отдельную папку. Создаём папку Sprites и скидываем все изображения туда. (На русском так и будет- спрайт. Обозначает графический объект.)

Теперь переносим бота из иерархии прямо в ассеты. НЕ в папку спрайтов! Это полноценный объект, у которого есть трансформ, свой спрайт и даже скрипты! Объясняем разницу детям. С экрана бота можно удалить. Объясняем, что теперь бота можно не только вернуть, но и клонировать. Позже можно будет даже спавнить.

Привычным движением добавляем wall и настраиваем как нам будет удобно. Wall переводится как стена. В компоненте Sprite renderer ставим уже знакомый нам Order in layer равным нулю. Попросите детей перевести гуглом что значит этот самый Order in layer.

1) Вопросы ожидаются, причём в большом количестве. Стоит отслеживать гравитацию, помочь объявить публично и вставить rigidbody в компонент (просто перетаскиваем из иерархии объект, над которым будем выполнять действия, содержащий в себе требуемый компонент.Стоит объяснить, что transform буквально телепортирует а не двигает объект, потому и столкновений быть не может.
2) Тут всё легко: закинул спрайт стены, добавил в него box collider, отрегулировал, добавил также box collider в юнита, теперь они стукаются.\
3) Возможно, понадобится разжевать детям, как сделать так, чтобы сообщения не сыпались когда юнит стоит на кнопке а отправляют по одному за заезд. Смотри мой код. Побуждайте детей делать изменяемое в самом компоненте сообщение чтобы не приходилось менять скрипт из-за него.
4) По сути тут никакого когда и нет. От детей требуется левел-дизайн и свои наметки по игре - какие механики хотят туда добавить - жизни,каких-нибудь ботов низшего уровня развития, оружие, хилки, ускорение… Даже не знаю. Стоит объяснить, что это первая учебная сборка, задача сейчас - понять, какие механики для игры им могут пригодиться. Поощряйте вопросы из серии “хочу вот эту штуку, как её сделать?”

**Домашнее задание**:
составить ряд алгоритмов, пользуясь блок-схемами. Например, маленькая механика стрельбы из пистолета: стреляем? да - переход к следующему вопросу. Нет - к началу алгоритма. Патроны в обойме есть? да - стреляем, уменьшая количество патронов на один и к началу алгоритма. нет - звук осечки, обоймы ещё есть? да - перезарядка 5 секунд и к началу алгоритма, нет - к началу алгоритма. Таких алгоритмов хватит 2-3 для общего понимания, потом можно будет доработать. Дз сообщить детям в конце и родителям м вотсапе по требованию сбросить pdf, что в этой папке. Не перепутайте с гугл-доком!



Задания:
1) Сделать движение юнита плавным, не пошаговым.
Для этого добавляем компонент Rigidbody2d в Юните. Это физика объекта. Запусти код. Юнит упал? Обнули гравитацию в этом компоненте. Сначала попробуй найти сам, если что - потом спросишь учителя. В коде обьяви новую открытую переменную типа Rigidbody2d. Советую назвать переменную rb, но это не обязательно.
Потом закомментируй уже готовое перемещение, bool-переменную тоже закомментируй. С командой transform вся дальнейшая магия не выйдет. Придётся писать заново. Напиши плавное перемещение с только одним условием нажатия, не походовое. Используй MovePosition, как в материалах. Сделай так, чтобы Юнит мог шагать в 4 стороны.

2) Добавить стены.
Wall с английского переводится как стена. Добавляем в неё Box Collider2D. Играемся с параметрами внутри до тех пор, пока картинка не будет совпадать с зелёными краями. Не видишь зелёные края? Надо отключить background. Спроси учителя, как. Только потом обратно верни. Запусти проект. Если твой Юнит не упирается в стену, значит, мы забыли ему тоже добавить Box Collider2D. Если нет - ты сначала целиком читаешь задание а потом выполняешь его.

3) Поставить кнопку, которая при нажатии на неё Юнитом будет говорить “Смени уже эти дурацкие клетки на что получше!”
Вытащи кнопку на экран и добавь в неё Circle Collider2D. Это такой же коллайдер как и Box, просто круглый. И самое важное - поставь галочку напротив пункта Is Trigger. Без этой галочки Юнит на неё не заедет - это будет просто круглая стена. А так компонент будет только отслеживать, пересекается ли с другим коллайдером, или нет. Создай новый скрипт TriggerSays, потом запихни в кнопку и создай управление по условию. Используй булеанову переменную чтобы не получать кучу одинакового текста в консоли со скоростью пулемётной ленты. Будет также очень неплохо для следующего задания, если ты сделаешь текст публичной переменной string.

И да, стоит создать папку Scripts для скриптов. Они потихоньку накапливаются.

4)Собрать карту из полученных объектов, подумать над её доработкой.

Создай папку Prefabs (с английского - сборный, типовой. В нашем случае мы там будем хранить объекты типа ботов, стен и т. д.) Засунь туда нашего бота, которого мы перенесли из иерархии в материалы. Также засунь стену и кнопку. Сделай карту, расставляя объекты. Такую, какая бы тебя устроила. Да, если карта получится большой а тебе захочется по ней покататься полностью, спроси учителя, как закрепить камеру за твоим Юнитом. Получится изменить текст кнопки в самом компоненте? Тогда можно поставить своего рода напоминалки: вот тут кнопка для открытия двери, эта кнопка даёт энергию, тут хочу поставить ловушку. В следующий раз будет проработка карты и взаимодействия объектов.

Материалы:

**int i = 0;** - объявление целочисленной переменной с именем i и значением 0. Имя может быть произвольным, значение - любым целым числом.
**float s = 0.1f;** - переменная с плавающей запятой. Значение может быть любой десятичной дробью. Можно поставить также любое целое число. После числа надо поставить f, иначе компилятор будет ругаться.
**string msg = “Замени надпись”;** - переменная, содержащая текст.
**bool R = true;** - булеана переменная. Может быть или true, или false. Удобная переменная для вещей, на которые можно ответить да или нет. Часто исполняет роль выключателя.

**public int i** = 0; - объявление публичной целочисленной переменной i, равной нулю. Публичной потому что к ней может теперь обращаться Юнити как со стороны пользователя, так и со стороны других скриптов.

**public Rigidbody2D rb;** - объявление открытой переменной типа Rigidbody2D, по факту - ссылка на компонент, с которым предстоит работать.Имя переменной - rb

**rb.MovePosition(rb.position + new Vector2(1f, 0f));** - изменить позицию переменной по имени rb. Новая позиция - это старая позиция с прибавленным вектором (1,0). То есть по оси X больше на единицу. Или же на единицу вправо. Является физическим движением в отличие от Transform.

**void OnTriggerEnter2D(Collider2D other)**
**{**
**тело программы**
**}** - функция на уровне void start и void update.
Исполняется при условии, что коллайдер объекта пересекается с другим коллайдером

**void OnTriggerExit2D(Collider2D other)**
**{**
**тело программы**
**}** - функция на уровне void start и void update.
Исполняется при условии, что коллайдер объекта НЕ пересекается с другим коллайдером