Обмен опытом и передовые идеи по использованию свободного ПО в образовании

Робот на экране. Движение и управление

Робототехника в школе – дорогое и очень сложное занятие? А вот и нет, если подойти к нему творчески, как это сделала **Татьяна Казанцева**.

Наш эксперт

Во время, свободное от корпения над написанием методики скрещивания Scratch и Arduino, **Татьяна Казанцева** оттачивает навыки работы со свободным ПО для использования в школе и дома.

Многие думают, что управление роботом требует наличия огромной «железяки», которая будет, подчиняясь командам пользователя по типу «Робот, иди туда», грохотать по помещению. Но если разобраться с том, что фактически представляет собой робот, то это просто Исполнитель, который выполняет определенные команды Пользователя или работает по заданному Алгоритму. Вы уже наверняка знакомы с этими понятиями и изучали их ранее на уроках информатики.

Поэтому прежде чем переходить к реальному роботу, надо потренироваться на его компьютерной имитации (симуляции), а также освоить работу с основными микроэлектронными компонентами – то есть радиодеталями и собранными на их основе датчиками и управляющими блоками (мы же не забыли, что Робот состоит из радиодеталей?). В рамках рубрики Школа **LXF** мы начинаем цикл статей, где все последующие задания основаны на управлении виртуальным роботом-исполнителем. Изучив и выполнив их, мы позже сможем управлять реальным механизмом, благо такой робот предоставлен нам проектом Тырнет и носит имя Скратчдуино. Мы будем использовать программную среду Scratch, поэтому прежде чем приступать к занятиям, советую вам пройти (вместе с учениками) курс по данной среде, который (как и учебные пособия) можно найти на сайте http://letopisi.ru/index.php/Школа_Scratch.

Терминология

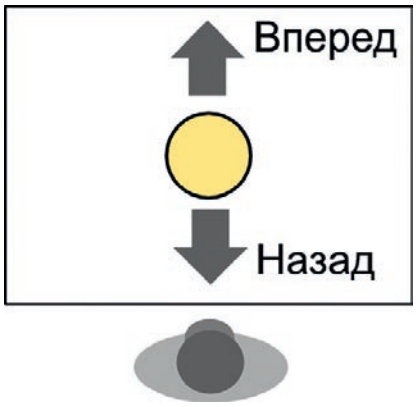
» **Исполнитель** Компьютерная программа или механизм, управляемый ею, выполняющий непосредственные приказы пользователя или работающий по заданному алгоритму. Исполнитель имеет определенный набор команд – систему команд исполнителя и ограничения работы.

» **Алгоритм** Последовательность команд, заданная пользователем, которую должен выполнить Исполнитель.

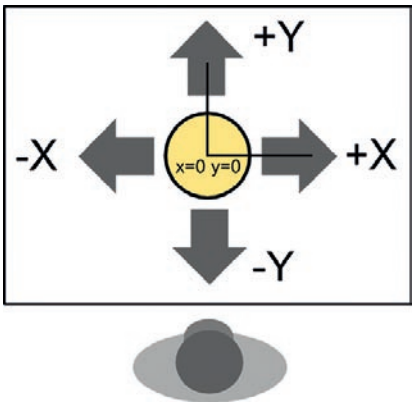
Движение робота

Говоря кому-то, куда ему двигаться, вы отдаете команду. Эта команда может быть привязана к некому ориентиру, чтобы Исполнитель команды мог понять, что от него хотят. В частности, ориентиром (или его еще называют маяком) можете быть вы – например, вы даете команду вашему приятелю: «иди ко мне». Также маяком может быть какой-то предмет: «Иди до того стола». Или команда может быть привязана к какой-либо системе координат.

Встав лицом к Исполнителю, вы сами будете для него маяком, и он сможет выполнять команды относительно вас – вперед (от вас), назад (к вам), влево и вправо. Или, если это экран, Исполнитель сможет двигаться вверх, вниз, влево или вправо, счи-



› Сориентируем нашего робота по направлениям.



› Наш робот двигается в абсолютных координатах и может быть установлен или перемещен в любую точку. В реальной жизни так работают, скажем, роботы-манипуляторы сборочных производств.

тая маяками стороны экрана. Что же выбрать? Так как мы имеем дело с роботом, то на местности мы имеем дело с привязкой к поверхности и сторонам света, поэтому считаем, что на экране перед нами вид на Робота сверху и он должен передвигаться относительно вас. Вы находитесь за пределами экрана, внизу. То есть, по команде «вперед» (от вас), робот на экране будет двигаться вверх; по команде «вниз» – к вам.

» **ЗАДАНИЕ** Нарисуйте, куда будет двигаться робот по командам «влево» и «вправо».

Движение в абсолютных координатах

Откройте Scratch, удалите текущий персонаж (щелчок правой кнопкой мыши на нем – Удалить) и выберите Рисовать новый объект  . Наш робот будет круглый, поэтому нарисуйте что-то похожее на закрашенный круг.

Теперь составим программу, управляющую роботом. Заглянув в раздел Движение, вы не обнаружите параметров «вперед, назад, влево, вправо», так как управление программным роботом (как вы уже, наверное, знаете) идет по системе координат. В нашем случае движение вперед будет означать увеличение координаты Y, назад – ее уменьшение. Вправо – увеличивает координату X, а влево – уменьшает ее. Центр экрана будет иметь координаты X=0, Y=0.

Задайте следующий код:

При старте программы робот устанавливается в исходную точку (координату 0,0). При нажатии стрелки вверх робот переместится вперед (то есть увеличит координату Y). Наш робот теперь имеет собственный набор команд – «вперед», «назад», «влево», «вправо»

» **ЗАДАНИЕ** Доработайте программу, чтобы робот перемещался по нажатию стрелок вниз, вправо или влево.

Ориентация (относительные координаты)

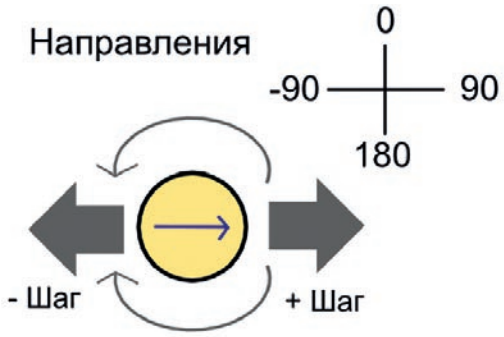
Попросите своего друга временно побыть Роботом. Пускай он встанет спиной к вам, а затем выполнит ваши команды «вперед, назад, влево, вправо». Вы заметили, что его ориентация относительно вас не изменилась?

Скомандуйте другу «повернись направо». Вы видите, что он стал к вам боком. Теперь дайте команду «вперед». Ваш друг, изображая робота, только что переместился туда, куда ранее вы могли переместить робота командой «влево». То есть робот сменил свое расположение в пространстве (ориентацию), и теперь старые команды вызывают его перемещение в другом направлении.

В реальной жизни роботы также обычно ориентированы – они не в состоянии просто переместиться влево или вправо, не поменяв свою ориентацию в пространстве.

Поправьте спрайт робота в меню спрайта, чтобы он стал выглядеть так (**Спрайт > Костюмы > Редактировать**): 

» **ВОПРОС** Каким образом теперь ориентирован робот относительно вас?



› **Наш робот ориентирован в пространстве.**

Вы можете заметить, что робот стоит, повернувшись вправо. Все спрайты в Scratch изначально имеют такое направление (направление 90 или «вправо»), которое затем вы можете изменить «кирпичиком»:

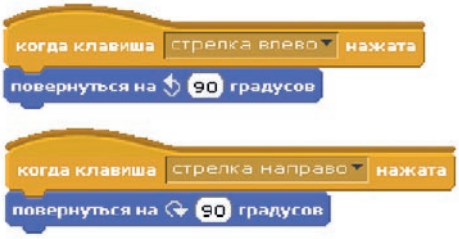
Текущую ориентацию можно увидеть в меню спрайта или вывести на экран, отметив в разделе **Движение** кирпичик **[] Направление**. Направления обозначены цифрами как: 90 – «вправо», 0 – «вверх, -90 – «влево», 180 – «вниз».

Видоизменим предыдущую программу, задав по нажатию клавиш влево и вправо поворот робота. Для этого используем блок

При повороте робота обычно не используют понятия «влево» и «вправо», а говорят, что он должен повернуться «по часовой стрелке» или «против часовой стрелки».

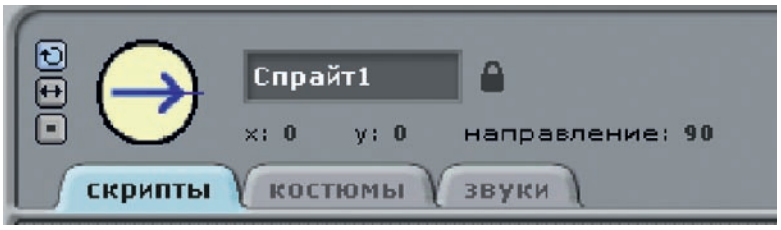
» **ВОПРОС** Куда повернется робот при повороте «по часовой стрелке»? Влево или вправо?

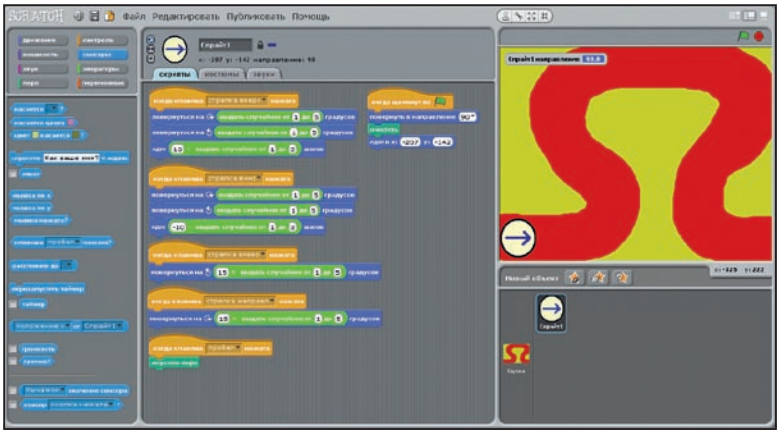
Измените в предыдущей программе блоки управления стрелками следующим образом:



Теперь запустите программу. Вы увидите, что при нажатии на стрелки влево или вправо робот поворачивается на 90 градусов. Но если нажать вверх или вниз, то он начинает смещаться, а не двигаться по направлению стрелки. Это происходит из-за того, что мы смешали системы координат. У нас теперь нет привязки к конкретным значениям и нет точки отсчета, поэтому необходимо использовать команды вида «вперед» или «назад», указав значение для перемещения.

› **Меню спрайта.**





Сцена для дороги робота.

Измените блоки управления стрелками «вверх» и «вниз», используя конструкцию

Это заставит робота переместиться в заданном направлении на указанное число шагов (в случае Scratch это число пикселей на экране). «Вперед» будет задаваться положительным числом (+), а «назад» – отрицательным.

У вас должна в итоге получиться следующая программа:

Робот может обойтись без заднего хода. Если его конструкция позволяет ему поворачиваться на месте, как у нашего идеального компьютерного Робота, то достаточно выполнить поворот на 180 градусов, и робот поедет в обратную сторону. Также робот может поворачиваться не только на 90 градусов.

» **ЗАДАНИЕ 1:** Поэкспериментируйте с углами поворота. Замените 90 градусов на 45, 30, 15, 5 и посмотрите, что при этом получится.

» **ЗАДАНИЕ 2:** Поменяйте начальное направление. Поставьте робота в направление «вверх» и посмотрите, как изменяется управление им.

» **ЗАДАНИЕ 3:** Измените величину шага. Выясните, как влияет данная величина на скорость движения робота.

Управляемость (точность отработки алгоритма)

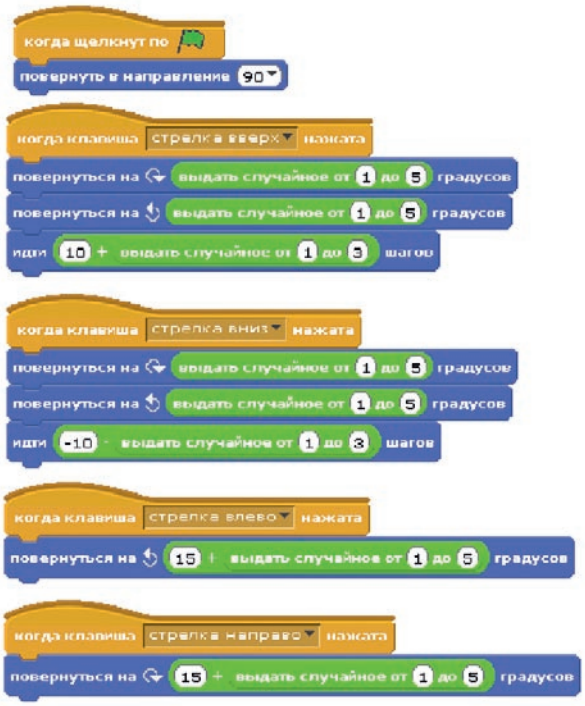
Наш робот уже может управляться с клавиатуры, и вы можете «гонять» его по экрану. Если вы выполнили предыдущие задания, то знаете, как сделать робота более управляемым – то есть заставить его поворачиваться на больший или меньший угол, изменять скорость его движения, а также задавать начальную ориентацию.

Но реальный робот имеет, увы, худшую управляемость (по крайней мере наш учебный робот, которого мы сможем запрограммировать в дальнейшем). Что это означает?

Реальный робот не может повернуться на точный угол (плюс-минус определенное количество градусов), не может переместиться на точное количество шагов, и возможны задержки и пропуски наших команд из-за проблем при передаче данных к роботу.

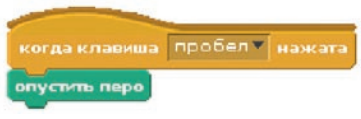
Даже установив так называемую «обратную связь» – то есть специальные устройства, которые смогут сказать, что робот выполнил действие с некоторой точностью – мы все равно будем иметь определенную погрешность, да и стоимость самого робота возрастет.

Во многих случаях точность робота по отработке команд алгоритма не так важна, как скорость реакции на события и отработки команд извне. Давайте посмотрим, как влияет неточность отработки команд алгоритма на нашего компьютерного робота. Для этого добавим в каждую команду случайную составляющую. Наша программа примет следующий вид:



Блоки поворота на случайную величину при движении вперед и назад нужны, потому что реальный робот при движении вперед обычно отклоняется в ту или иную сторону из-за погрешности работы двигателей.

Вы можете отследить текущее перемещение робота с помощью «пера» в Scratch. Для этого добавьте следующий блок:



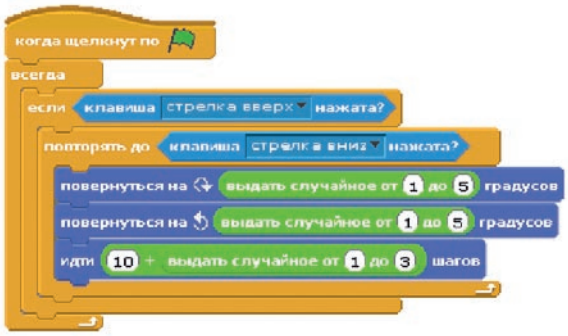
Теперь по нажатию клавиши «Пробел» робот будет оставлять за собой «след».

В код начальной инициализации после задания направления не забудьте добавить «кирпичик» **Очистить**. Также вы можете установить начальные координаты робота, отличные от (0,0).

» **ЗАДАНИЕ 1:** Нарисуйте роботу дорогу, похожую на изображенной на рисунке, и загрузите ее в качестве фона Сцены. Попросите ваших друзей или знакомых проехать по данной дороге, а также потренируйтесь сами управлять «почти» реальным роботом.

» **ЗАДАНИЕ 2:** Поэкспериментируйте с величинами случайных чисел. Чем больше данное число, тем хуже работает робот. Что произойдет, если величина случайного числа будет больше самой величины действия?

А теперь сделаем так, чтобы при нажатии стрелки «вверх» робот начинал двигаться вперед, а при нажатии «вниз» останавливался. Для этого нужно заменить код алгоритма по нажатию этих стрелок на такой единый блок:



Что он делает? При нажатии на флажок запуска запускается работа бесконечного цикла и идет проверка условия **Если нажата стрелка вверх**. При его выполнении начинает работу **Повторять до** – это будет продолжаться в цикле, пока не нажмут стрелку «вниз». Внутри цикла – код движения вперед с погрешностью.

» **ВОПРОС** Когда управляемость роботом лучше – при использовании прерывистого или непрерывного движения? Что менее утомительно для управляющего роботом? Какой способ управления вы бы предпочли?

Если робот едет слишком быстро, уменьшите значение шага, к примеру, до 5 (а случайную величину сделайте от 1 до 2) или добавьте в цикл после «кирпичика» Идти задержку («кирпичик») в 0,2–0,5 секунд.

» **ВОПРОС** Как влияет изменение величины шага и задержки на управляемость Роботом?

Галерея с примерами расположена на сайте http://scratch.mit.edu/galleries/view/157855.

Надеемся, что вы легко освоите начальные премудрости робототехники и написания программ в Скратч. В дальнейшем нас ждут вещи посложнее – определение положения и движение в заданные места, датчики и, если у вас будет интерес к данной теме, использование реальных роботов и контроллеров. Напишите нам на info@linuxformat.ru, если эта тема для вас актуальна. LXF