Роботы-спортсмены? Да!

Возможно ли такое? **Татьяна Казанцева** утверждает, что возможно – для них даже существует своя Олимпиада.

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.

Как узнать, чей робот лучше — устроить конкурс красоты? Поверить на слово его создателям? Верный ответ нам может дать только честная борьба. Робоолимпиады (или Robocup) — уже признанный способ выявить, чей робот имеет лучшую конструкцию и алгоритмы работы и кому благоволит фортуна.

В прошлых выпусках **LXF**, изучая датчики, мы уже фактически сделали первые шаги в сторону соревнований – езда по линии и выход из лабиринта являются фундаментальными дисциплинами, которые робот должен обязательно пройти.

Участие в таких соревнованиях является не только желательным, но даже обязательным для признания конструкции робота успешной, и позволяет выявить дефекты конструкции.

Давайте поднимем планку и создадим симуляцию более сложных дисциплин, на которых обкатаем алгоритмы, впоследствии применимые к реальному «железу».

Японская борьба

Начнем с симулятора японской борьбы сумо. Она как-никак подходит к роботам – не правда ли, вид толстячков-сумотори (борцов) сверху напоминает круглый корпус робота? Правила борьбы очень просты: вытолкни другого из круга. Но задача данного плана гораздо сложнее, чем просто езда по линии – нам надо смоделировать некоторые физические явления, а именно столкновение роботов. То есть роботы должны двигаться, сталкиваться и отпихивать друг друга.

Так как полноценную физику взаимодействия описать достаточно сложно (особенно в Scratch, и особенно для тех, кто не знаком с понятиями момента и упругого соударения и тригонометрическими функциями), мы попытаемся сделать упрощенную модель, которая с некоторым приближением будет имитировать реальную, но обходиться без сложных функций.

Представим, что у робота два колеса, и если его специально не повернуть, он может двигаться при столкновении только вперед и назад. Для этого мы используем понятие направления Scratch (то есть угол поворота робота) и движения, происходящего вперед и назад по прямой линии направления. Борьба будет между красным и синим роботом, поэтому создайте два персонажа, как показано на рис. 1.

Рис. 1. Портреты роботов-соперников: горячий против холодного?

Понятие направления мы разбирали в статье **LXF156** «Робот на экране. Движение и управление» в рамках понятия Ориентация.

Так как наш робот круглый, нужно будет только определить, куда двигался робот соперника, а куда – наш робот, и подвинуть робота в нужном направлении. Код приблизительно будет выглядеть следующим образом:

Если наш робот ехал в положительном направлении (больше нуля – направо), а робот соперника двигался в отрицательном (меньше нуля – налево, навстречу нам), то нас отбросит назад. То же произойдет, если мы двигались в отрицательном направлении (налево), а робот соперника, наоборот, направо: нас также отбросит назад (то есть нужно будет двигаться на минус число шагов).

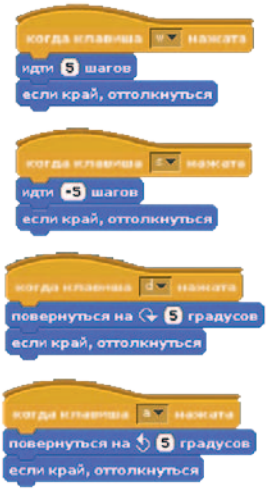
Для отслеживания направления мы вводим две переменные – **napr1** и **napr2**, для красного и синего робота соответственно, а изменение направления определяем событием **kasanie**.

Коды для красного и синего робота абсолютно одинаковы.

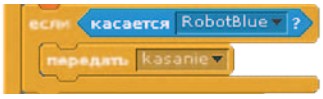
Добавив к роботам код управления, можно перемещать роботов по полю соответственно стрелками курсора и клавишами WASD.

Код для красного робота –

Код для синего робота –



Далее нужно реализовать код реагирования на касание. Передадим сигнал **kasanie**, если будет касание противоположного персонажа. Например, для красного робота этот участок кода будет выглядеть так:



Теперь уже можно погонять роботов по полю; но наша задача – реализовать настоящую борьбу. Для этого запрограммируем арену.

Нарисуйте фон для сцены, как показано на рис. 2.

Как видно, у нас имеется арена желтого цвета и зеленое поле реагирования для определения выталкивания робота. Вы можете оформить арену как вам нравится – главное, не используйте цвет, выбранный для пространства вне арены, чтобы они зрительно не сливались.

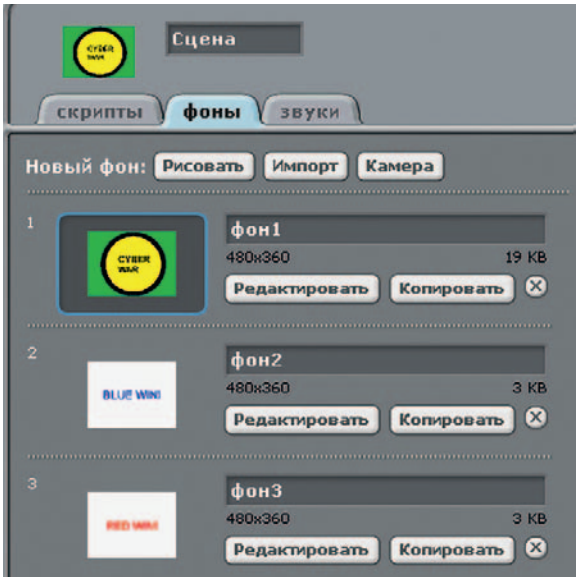
Также создадим еще два фона: один с сообщением о выигрыше красного робота, второй – синего (рис. 3).

Для отображения нужного фона введем два события – **redwin** и **bluewin**.

Код для сцены будет таким:

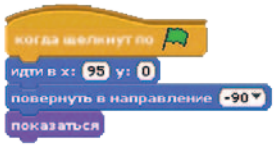
Теперь нам остается сделать две последние вещи: дописать к коду касания выдачу сигнала, обозначающего выигрыш робота противоположного цвета при касании зеленого поля, а также поставить роботов в начальное положение. Допишем эти два участка кода.

› Рис. 2. Арена, как в цирке, посыпана песком, а проигравшего выпрут на травку.

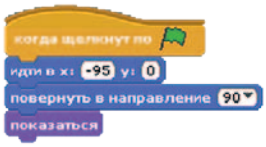


› Рис. 3. Нельзя же стадиону не иметь табло результатов.

Для красного робота –



а для синего –



Вы можете скачать полный код проекта с сайта http://scratch.mit.edu/projects/akdengi/2618677.

В следующей части мы снабдим наших роботов «разумом» и разберем различные алгоритмы их управления. Если вы сумеете сделать это раньше нас, напишите нам на info@linuxformat.ru, и мы обязательно опубликуем и ваше решение. LXF