Дополнительные материалы к курсу информатики 5-6, 7-9 классы на основе завершенной предметной линии учебников «Информатика» для 5–9 классов общеобразовательных учреждений Л.Л.Босовой, А.Ю.Босовой

Часть первая (5-6 классы)

Время реализации модуля по робототехнике (включая программирование) – 18 час.

Авторы:

Воронина Вероника Вадимовна
учитель информатики МБОУ СОШ №7 г.Павлово

Воронин Игорь Вадимович
начальник отдела информационных технологий Института Проблем Лазерных и Информационных технологий Российской Академии Наук

Павлово, Москва

2016

Содержание

В современном мире цивилизованного человека на каждом шагу может ожидать встреча с роботами.

А вообще-то, что такое робот? Какие роботы бывают? Как ими управлять и что нужно знать, чтобы научиться и самому сделать робота?

Давайте подумаем вместе...

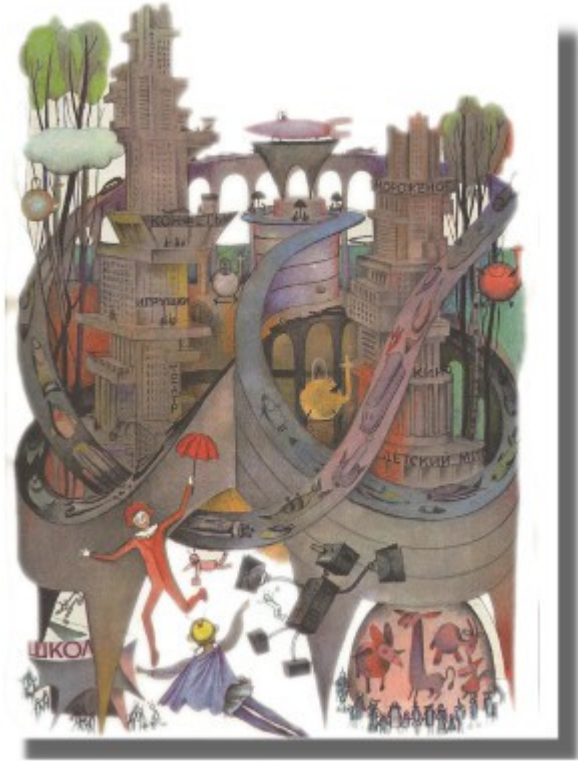
§ 1 Эпизод первый. Как роботы стали роботами. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=2

Тема по информатике: Компьютер. Компьютер — универсальная машина для работы с информацией.

Давным-давно, когда деревья были большие, а ученые из Академии Наук еще ходили в школу, в детских журналах часто устраивали конкурсы как дети видят будущее.

В очень популярном тогда журнале "Пионер", время от времени, печатались такие футуристические материалы на тему, каким будет мир после 2000-ого года. Там были картинки городов из высоких домов, по крышам которых протекает река, всюду снуют летательные аппараты и самое главное, всю тяжелую и нудную работу за людей выполняют роботы..



И вот, будущее время уже наступило...

Больше полутора десятков лет прошло с начала двухтысячного года, роботы вокруг нас стали обыденным явлением. Никого не удивляет ползающий по квартире робот-пылесос, на автомобилях и самолетах автопилот-робот ведет управление по круиз-контролю. Готовят пищу робот-хлебопечка и робот-мультиварка.

Вообще — что такое робот? Какие роботы бывают? Как ими управлять и как создать робота самому? Давайте подумаем об этом вместе...

- *Так что же такое робот?*
- *В какой момент наши добрые помощники пылесос, автомобиль, кофеварка, так поумнели, что превратились из просто агрегатов в наших интеллектаульных друзей?*
- *Чем отличается робот от неробота?*

1.1 Как роботы стали роботами

Вы конечно хотя бы раз играли игрушками с дистанционным управлением.

Возьмем к примеру обычную радиоуправляемую машинку — это точно не робот: она ездит вперед-назад и в сторону, только после того как на пульте нажмут рычажок в нужную сторону.

Хотя команды движения и приходят на машинку по радиоканалу, но никакой логики там не обрабатывается.

А вот к примеру управляемый с точно такого же пульта и так же по радиоканалу квадрокоптер — он уже может являться полноценным роботом.

Как вы думаете, почему?

Наверное потому, что на борту квадрокоптера стоит чип-микрокропроцессор и в нем, по заранее разработанной и загруженной в запоминающее устройство программе, идет анализ полученных команд, рассчитывается изменения скорости вращения моторов и сравнение текущих параметров с набегающим ветром или появляющимся препятствием.

А пылесос? Когда он становится роботом? Обычный просто работающий пылесос будет долго шуметь стоя на одном месте, если его никто никуда не тянет за шланг.

Но робот-пылесос в виде «таблетки» не нуждается ни в чьем понукании — он сам переползает от одной стены до другой по комнате, причем так хитро, что умудряется пройти над каждым кусочком пола не более чем 2-3 раза. Как это он делает? Откуда робот пылесос берет координаты и запоминает свой путь? Ведь он не имеет навигатора, чтобы узнавать свое положение по глобальной системе координат GPS, он отсчитывает шаги с запоминает направление, рассчитывая куда ему ползти.

Дело в том, что опять же, как и в случае с коптером — в робот-пылесосе есть датчики обратной связи. Они установлены на колесах и их называют энкодеры. При вращении колеса — центральный процессор получает данные об угле поворота и, складывая эти значения в переменную, выстраивает виртуальные координаты движения относительно препятствий в комнате. А дальше, в бесконечном цикле — по этим координатам робот ползет, как бы мысленно закрашивая свой путь, а далее уперевшись в стенку и развернувшись прокладывает маршрут по еще незакрашеному пути. И так до тех пор, пока он не «закрасит» своим следом все свободное пространство, либо пока не кончится заряд батареи на его борту.

Для того чтобы он не завис жалобно попискивая о середине комнаты не в силах доехать до пункта заряда, грамотные разработчики заложили в него еще одну программу, которая постоянно с некоторой периодичностью например раз в минуту, опрашивает контроллер заряда батареи. И как только он достигает критической величины, так сразу же прерывает выполнение программы уборки помещения и отправляет его на базовую станцию: «подкрепится» энергией.

Итак, давайте будем называть роботом то устройство, которое снабжено «мозгами» конечно не такими умными, чтобы принимать участие в дискуссиях с человеком, но в которые можно заложить программу и запустить ее выполнение.

Как вы уже наверно знаете, в компьютере обязательно присутствуют: центральный процессор (чип), оперативная память и постоянная память. Тоже самое есть и у нас в любом роботе. Но если в компьютере стоит мощный процессор, который может делать тысячи бестолковых операций в секунду, и при этом потребляет кучу энергии, то в роботах как правило ставят для экономии ресурсов и средств вычислительные системы с более простой архитектурой.

Термин – **Программа** — это код, написанный и отлаженный в поисках ошибок, на каком одном из алгоритмических языков, например Си. В учебных целях используют языки программирования, например, КУМИР или Scratch — с ними мы познакомимся в дальнейшем.

Программу в робота загружают. Способы загрузки бывают разные, например, при помощи программатора[1], или по радиоканалу. Загрузка программы в устройство еще называется прошивка[2] устройства.

Загруженная в робота программа, когда он выключен, хранится в ПЗУ. Как только подаем питание, так процессор сразу же из ПЗУ отправляет весь этот программный код в оперативное запоминающее устройство и запускает выполнение программы в бесконечном цикле, пока не совершится выход по прерыванию.

Таким образом, роботом можно назвать то устройство, которое мы с вами можем запрограммировать на выполнение каких-то действий.

1Аппаратно-программное устройство, предназначенное для записи/считывания информации в постоянное запоминающее устройство (однократно записываемое, флеш-память, ПЗУ, внутреннюю память микроконтроллеров и программируемые логические контроллеры)

2Программа хранящаяся в постоянной памяти устройства.
Эти программы могут меняться, поэтому время от времени возникает необходимость перепрошивать (обновлять) память компьютерных устройств.

Например, обычный пылесос Циклон — не робот, а робот-таблетка — точно уже робот. Потому что ползает по программе с учетом обратной связи от датчиков.

И старый автомобиль Волга ГАЗ-21, к примеру, точно не робот,

а роботом, можно считать современный гугл-автомобиль, на который уже установлен бортовой вычислительный комплекс и активный круиз-контроль. И такой автомобиль может не только сохранять постоянную скорость на трассе, но и фиксировать постоянное расстояние до впереди идущего автомобиля и реагировать в автоматическом режиме на внезапно появляющееся препятствие, например котенка выскочившего из-за угла. Это уже точно робот.

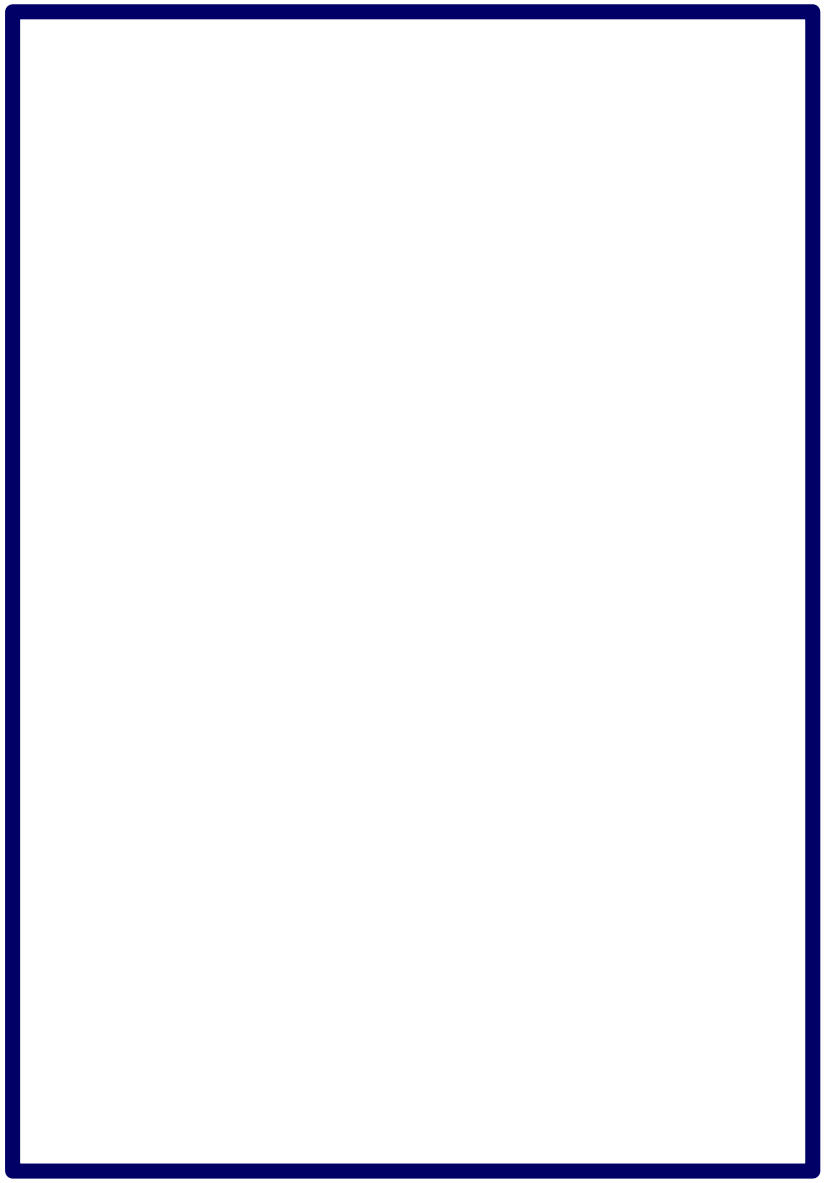
1.2 Задание: «Что нам стоит...Проектируем робота»

Нарисуй, или составь из различных частей картинок и наклей собственного робота. Придумай роботу имя, придумай что он будет делать, придумай команды, который он сможет выполнять.

Этого робота зовут:

Этот робот умеет:

Робот знает команды:



§ 2 Эпизод второй. Модель схемы аппарата Морзе. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=3

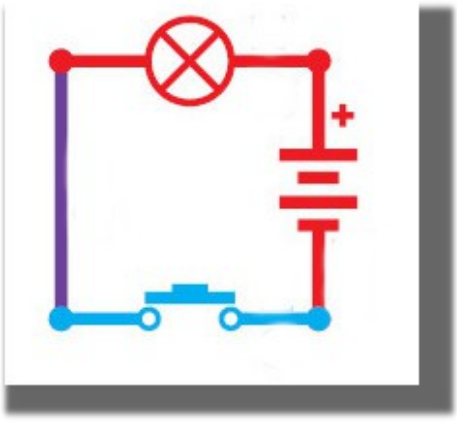
Тема по информатике: Информация вокруг нас. Код, кодирование информации.

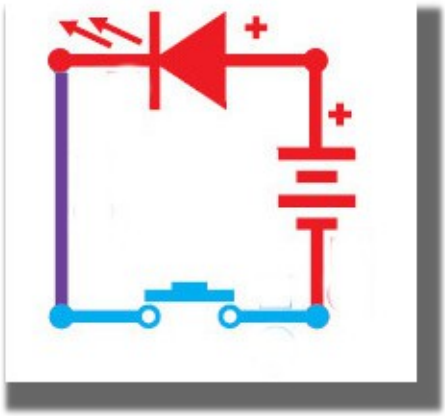
Прошло столько времени, а мы еще так ничего собственными руками еще и не собирали, а ведь настоящие инженеры обязательно должны иметь опыт работы руками. Вот и Морзе собирал свой аппарат «из чего попало». Просмотрите фрагмент мультфильма «Алло, вас слышу»

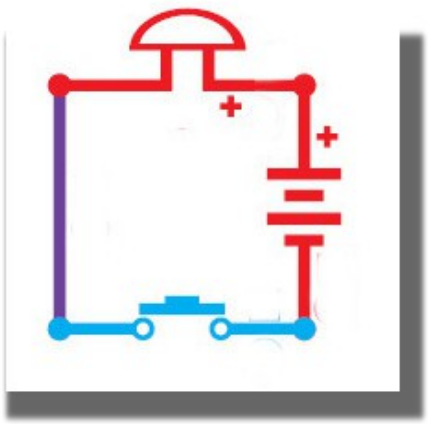
Ну мы не будем, конечно использовать что попало для нашей работы, а попробуем создать аппарат для передачи звуковых и световых сигналов на расстоянии с помощью набора электронного конструктора.

Немного истории: телеграф был первым средством связи за пределами слышимости человеческого голоса. До него связь была визуальной: индейцы подавали сигналы с помощью дыма, на флоте применяли сигнальные флажки, а на железной дороге – рычажный семафор. В темноте и при плохой видимости визуальные сигналы были бесполезны. Но когда они были видны, то оказывались самой быстрой формой связи, поскольку передавались со скоростью света. Электрический ток распространяется по проводнику почти с такой же скоростью. Первый приемник телеграфных сообщений Сэмюэла Морзе был снабжен двумя электромагнитами, которые приводили в движение перо, писавшее на бумажной ленте точки и тире. Часовой механизм протягивал ленту через аппарат.

Попробуйте спроектировать аппарат, который сможет передавать сообщения визуально или с помощью звука, кодируя их кодом Морзе. Обратите внимание, что светодиод и генератор звука будут работать только при соблюдении правильной полярности (+ на светодиоде должен подключаться к + элемента питания)







Выполните на реальных приборах задания №86-87 рабочей тетради по информатике 5 класса, передайте и расшифруйте сообщения, полученные от товарищей.

86. Дана кодовая таблица азбуки Морзе:

А	·—	И	··	Р	·—·	Ш	————
Б	—···	Й	·———	С	···	Щ	——·—
В	·——	К	—·—	Т	—	Ъ	·——·—·
Г	——·	Л	·—··	У	··—	Ы	—·——
Д	—··	М	——	Ф	··—·	Ь	—··—
Е	·	Н	—·	Х	····	Э	··—··
Ж	···—	О	———	Ц	—·—·	Ю	··——
З	——··	П	·——·	Ч	———·	Я	·—·—

Расшифруйте следующие записи.

1) ·—— ·· —· ———· · ··· — · ·—·

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

8) —— —·—— ———— —··—

☐ ☐ ☐ ☐

9) ·——· ·—· ·· —· — · ·—·

☐ ☐ ☐ ☐ ☐ ☐ ☐

10) ··· —·— ·— —· · ·—·

☐ ☐ ☐ ☐ ☐ ☐

87. Зашифруйте с помощью азбуки Морзе:

Своё имя	
Название любимого школьного предмета	

§ 3 Эпизод третий. Знакомимся с программированием. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=4

Тема по информатике: Информация вокруг нас. Обработка информации.

Ранее, мы с вами обсудили и пришли к соглашению, что роботом агрегат становится тогда, когда его можно запрограммировать.

Скажите пожалуйста, а вы умеете программировать?

Да-да, писать настоящие настоящие компьютерные игры или создавать программное обеспечение для планетоходов?

Наверное вы думаете, что программирование это очень трудное занятие, доступное лишь для тех, кто имеет очень хорошую математическую и техническую подготовку.

Но, представьте себе: игру для компьютера, можно не только найти среди установленных игр операционной системы или скачать в Интернете – что-то можно создать и самому!

Мы с вами будем учиться работать в среде программирования Scratch, освоим работу с несложным графическим редактором, изучим технологию мультипликации, заставим слушаться нас настоящих роботов и научимся программировать, зачастую даже не замечая, что учимся программированию.

Знаете, используя среду программирования Scratch можно создавать собственные библиотеки и творчески использовать чужие разработки. В сообществе Scratch все пользователи, делятся своими находками, словно пирожками на встрече кулинаров – вы можете попробовать любой пирожок и использовать понравившийся рецепт в дальнейшем, вносить в рецепт свои изменения. А можете предложить свой эксклюзивный торт, который другие пользователи захотят украсить разными финтифлюшками, но исходное авторство остается за вами....

Итак, вперед – пацаны выбирают Scratch! (и девушки тоже :)

3.1 Знакомство с Scratch. Как управлять объектом

Scratch – такой язык программирования, в котором программу нужно собирать из разноцветных команд, цепляя их друг к другу. Команды сгруппированы в в разноцветных ящиках.

Запустим программу, выбрав на рабочем столе симпатичную мордочку кота:

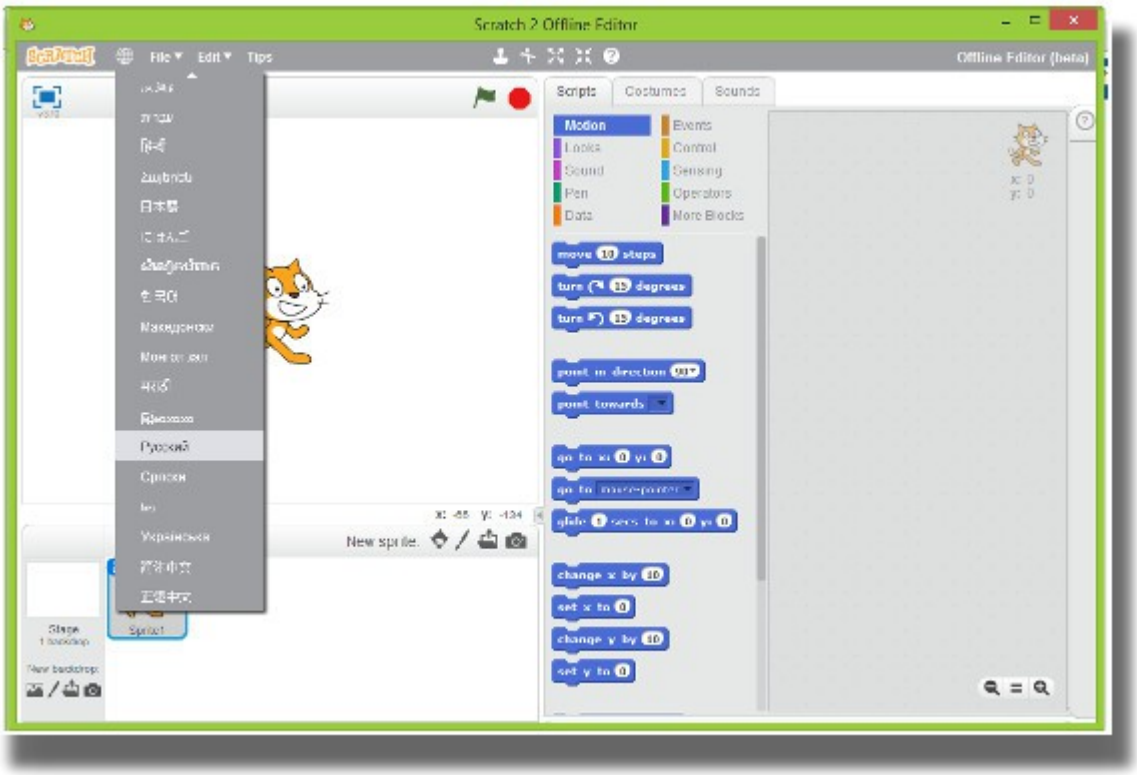
У нас откроется окно, разделенное на три зоны:

- в центре – зона разноцветных ящичков с командами;
- справа – непосредственно поле управления (включает зону скриптов (Scripts), зону костюмов (Costumes) и зону звуков(Sound);
- и слева – поле, где будет разворачиваться наше действо, и где удобно разместился наш исполнитель, пока имеющий вид Кота.

Как видим, все команды первоначально представлены на английском языке. Конечно, для дальнейшего профессионального программирования удобнее с самого

начала работать на английском языке, однако если вы пока не сталкивались с английским письменным, можно для начала переключиться и на русский язык.

Для этого, сверху, в главном меню щелкнув на кнопке с изображением земного шара, выберем, опустившись вниз русский язык:

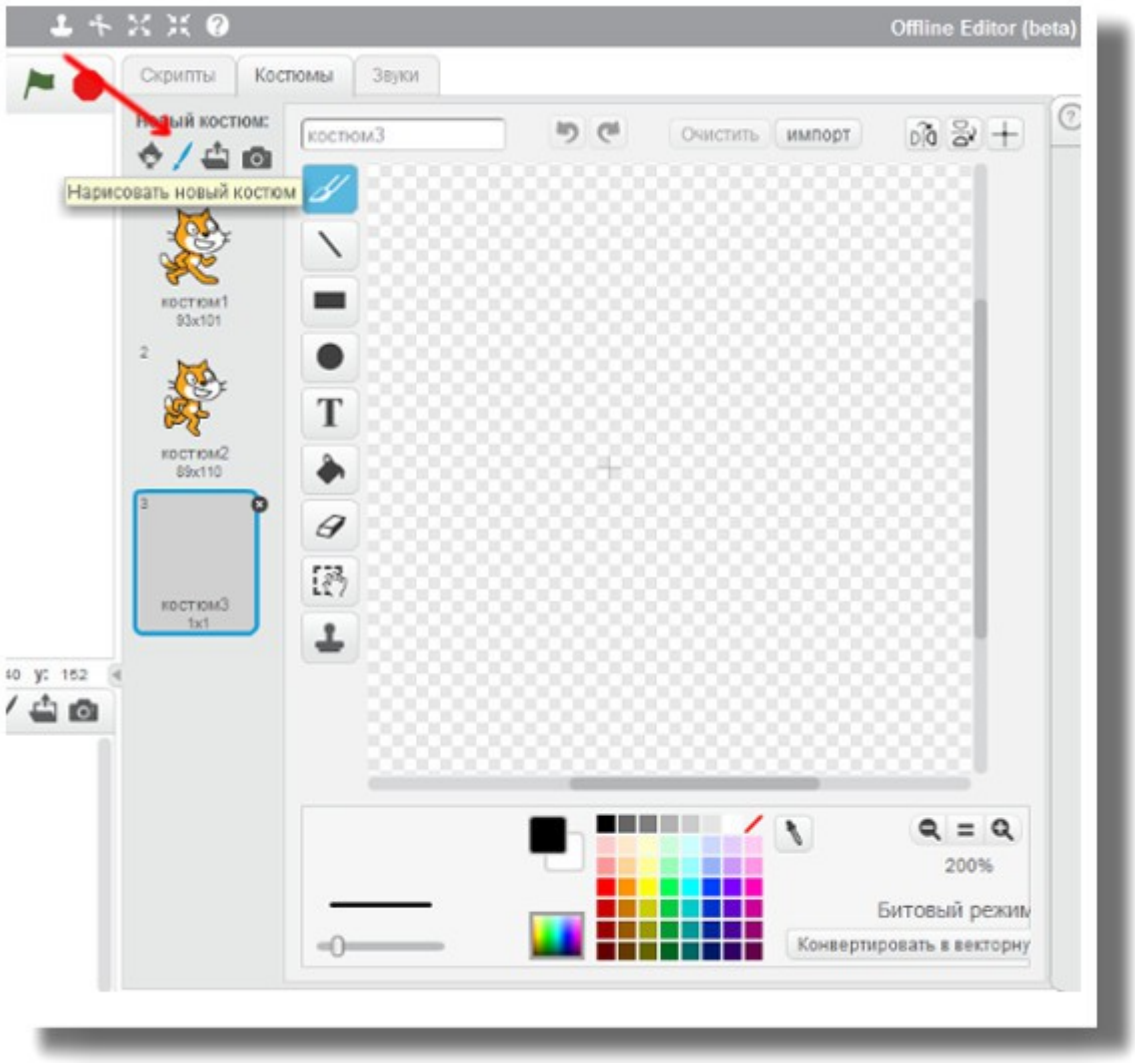


3.2 Графический редактор Scratch. Изменяем исполнителя

(изменение формы представления информации)

Хотя в библиотеке Scratch и есть много интересных исполнителей, попробуем наши первые программы создавать для своего собственного исполнителя – «Умного карандаша».

Загрузим новый проект Scratch, переходим на вкладку **Костюмы**, выбираем кнопку в виде кисточки **Нарисовать новый костюм**

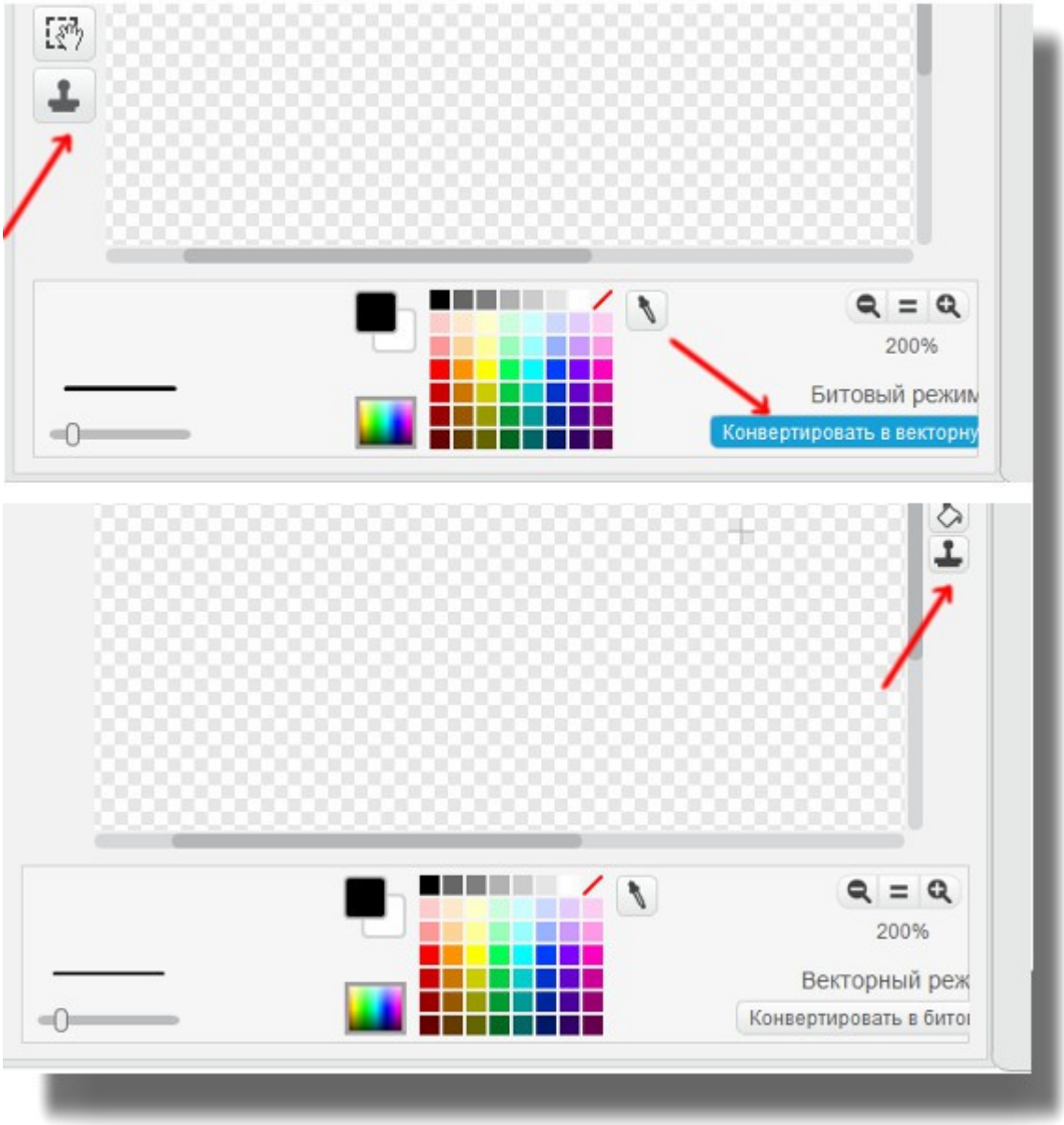


и у нас открывается окно графического редактора.

Нужно отметить, что в среде Scratch имеется два встроенных графических редактора: растровый и векторный. В нескольких словах: отличие заключается в том, что в растровом редакторе изображение складывается из отдельных точек – пикселей, а в векторных редакторах, изображение строится из отдельных графических примитивов: линий, прямоугольников, окружностей, и как следствие, картинки получаются более мультяшными, подробнее об этом будем говорить на следующих уроках.

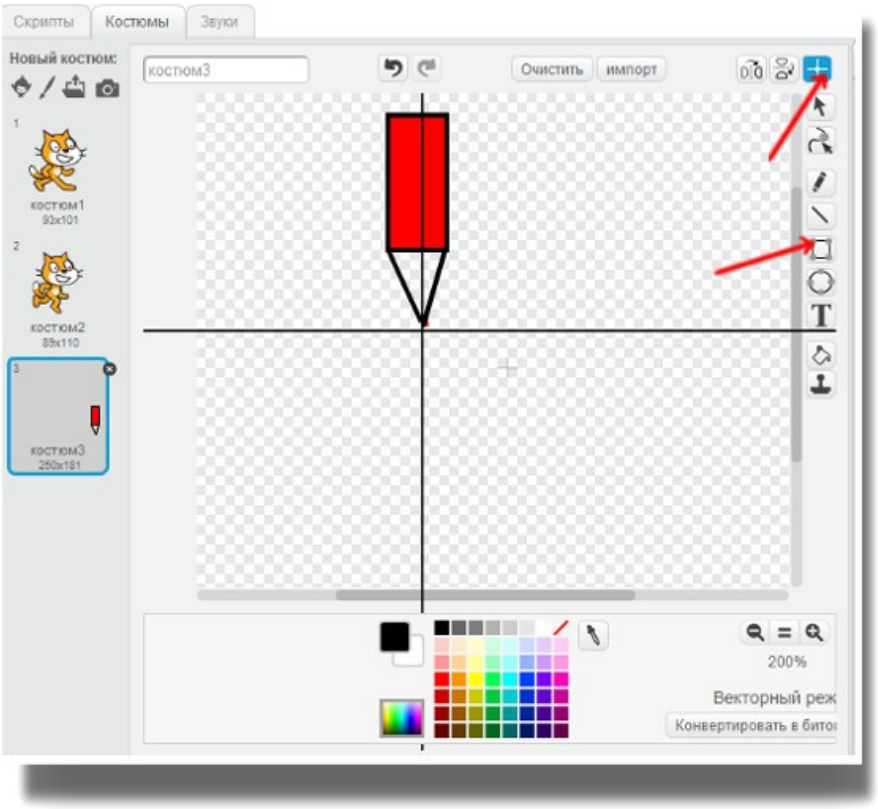
С растровым редактором мы поработаем чуть позже, а вот костюм нового исполнителя карандаша, нарисуем в векторном редакторе, для чего в правом нижнем углу, щелкнем на кнопке **Конвертировать в векторную графику**.

Обратите внимание, инструменты графического редактора переместились с левой



стороны в правую, а кнопка приобрела вид: **Конвертировать в битовый режим**. Т.е. вернуться в растровый режим мы можем вновь щелкнув на этой же кнопке.

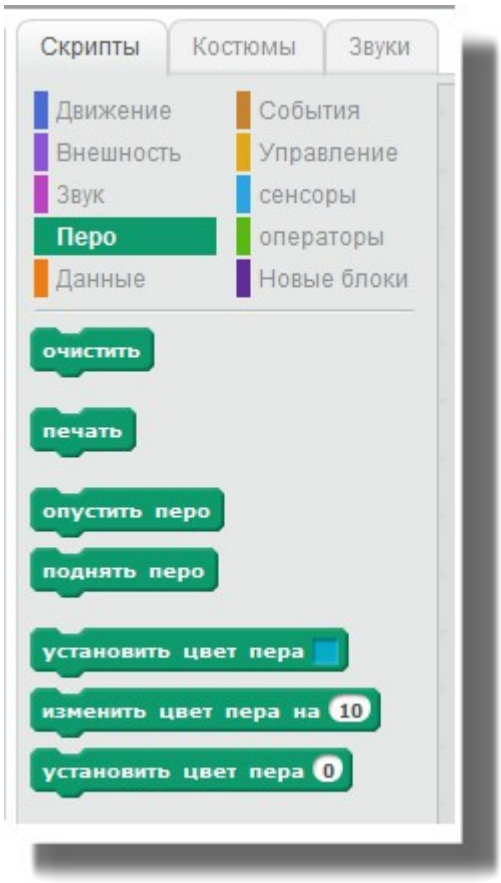
Пользуясь инструментами векторного графического редактора нарисуйте карандаш, причем, конец грифеля определите центром костюма: кнопка в виде перекрестья в правом верхнем углу – это необходимо, чтобы карандаш рисовал у нас именно кончиком грифеля, а не тупым концом или вообще серединой.



3.3 Учим карандаш рисовать

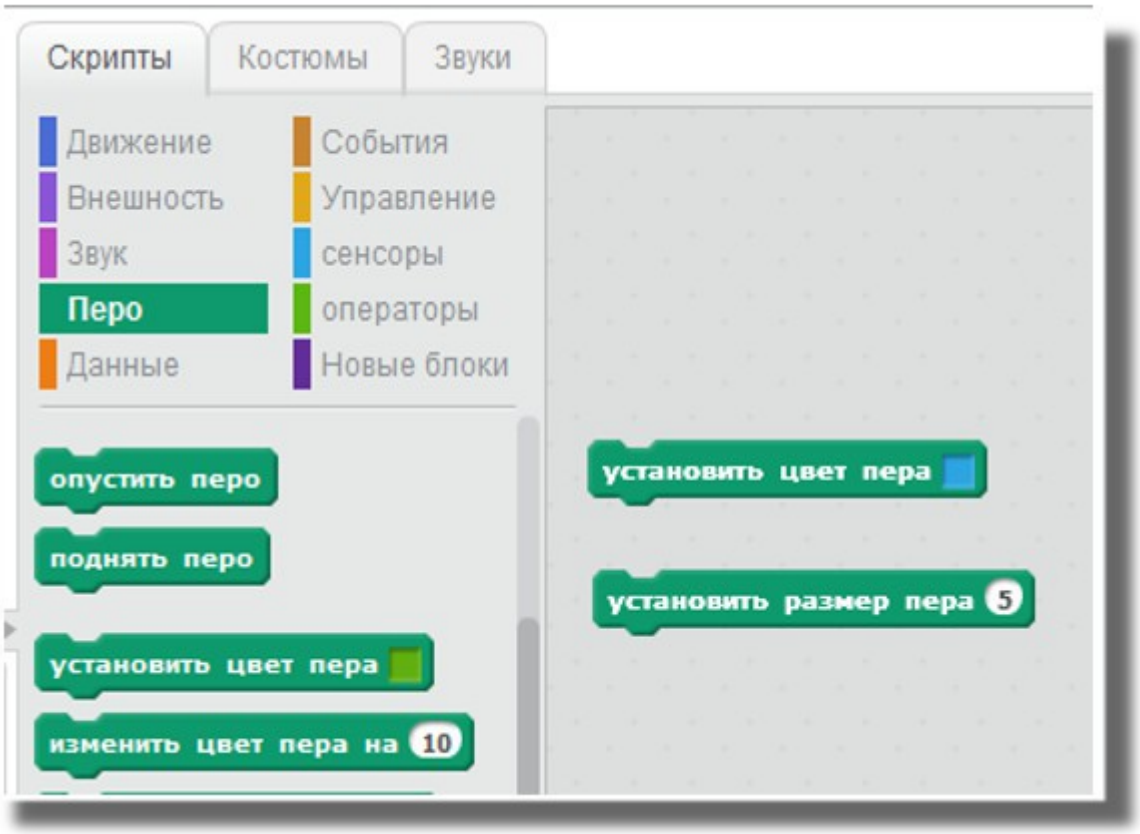
Итак новый исполнитель создан – теперь наша задача, научить его рисовать по программе.

Для рисования нам потребуются команды из темно-зеленого ящика **Перо**:



Понятно, что по команде **Опустить перо**, исполнитель оставляет след, при использовании команды **Поднять перо** – следа не остается, ну и чтобы в ходе отладки программы нас не путали старые следы рисования, лучше всего, сразу после запуска на выполнение программы давать команду **Очистить** – начинать работу с чистого листа.

Вдобавок, мы можем изменить цвет пера и толщину линии командами:



Для изменения цвета, щелкаем в квадратике цвета, а затем на рабочем поле подбираем цвет наиболее близкий к желаемому. Установить необходимый размер пера можно в соответствующем поле с числом.

3.4 Анализируем программу

Чтобы исполнитель двигался по кругу, необходимо создать программу, в которой после каждого шага, исполнитель слегка изменял бы направление – шагнул и чуть-чуть повернулся, шагнул и снова повернулся. И естественно, к такой конструкции необходимо применить команду цикла.

Поскольку мы хотим, чтобы наш Карандаш-исполнитель постоянно рисовал круг, можно использовать команду бесконечного цикла *Всегда*.

Если же нужно, чтобы пройдя круг один раз, исполнитель остановился, или вообще прошел по дуге, не замыкая полного круга, используем циклический оператор *Повторить()раз*, где параметр определяет число повторенных действий.

Чтобы программа была нагляднее предложим нашему исполнителю оставлять за собой след, то есть, прежде, используем команды рисования из темно-зеленого ящика **Перо**.

3.5 Задание. Движение по кругу.

Создайте исполнителя в виде карандаша.

Разработайте для исполнителя-карандаша программу движения исполнителя по кругу, проанализируйте результаты работы, изменяя по-очереди каждый параметр, определите как изменить диаметр круга, при каких значениях диаметр уменьшается, в каких случаях увеличивается. Заставьте исполнителя пройти по дуге.

Разработайте программу движения исполнителя по кругу, проанализируйте результаты работы, изменяя по-очереди каждый параметр, определите как изменить диаметр круга, заставьте исполнителя пройти по дуге.

Шпаргалка_02_001.pdf

Результат может иметь вид как в файле: 02_01.sb2 .

3.6 Сохранение результатов

Ну теперь, нужно сохранить наши с таким трудом достигнутые результаты. Выбираем команду Файл- Save as (Сохранить как), (в принципе можно воспользоваться и просто командой Сохранить, но если работа идет с чужим проектом, изменяя программу, всегда нужно пользоваться командой **Сохранить как**).

§ 4 Эпизод четвертый. Система координат экрана. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=5

Тема по информатике: Информация вокруг нас. Метод координат.

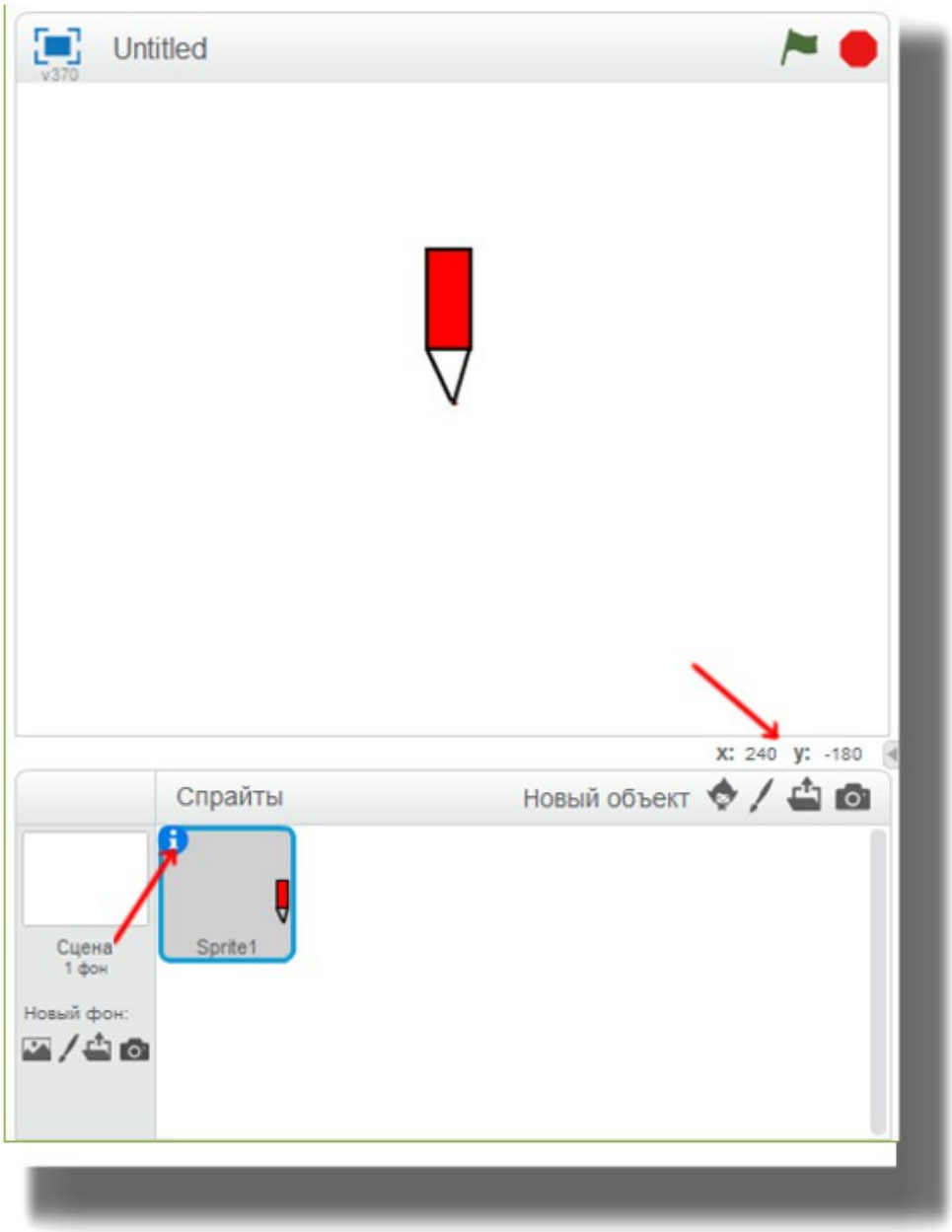
4.1 Координатная площадь Scratch

Когда мы с вами учили нашего исполнителя рисовать, при нескольких запусках программы наши рисунки размещались все время в разных местах рабочего поля, и иногда даже не хватало места для размещения желаемого изображения на рабочем поле.

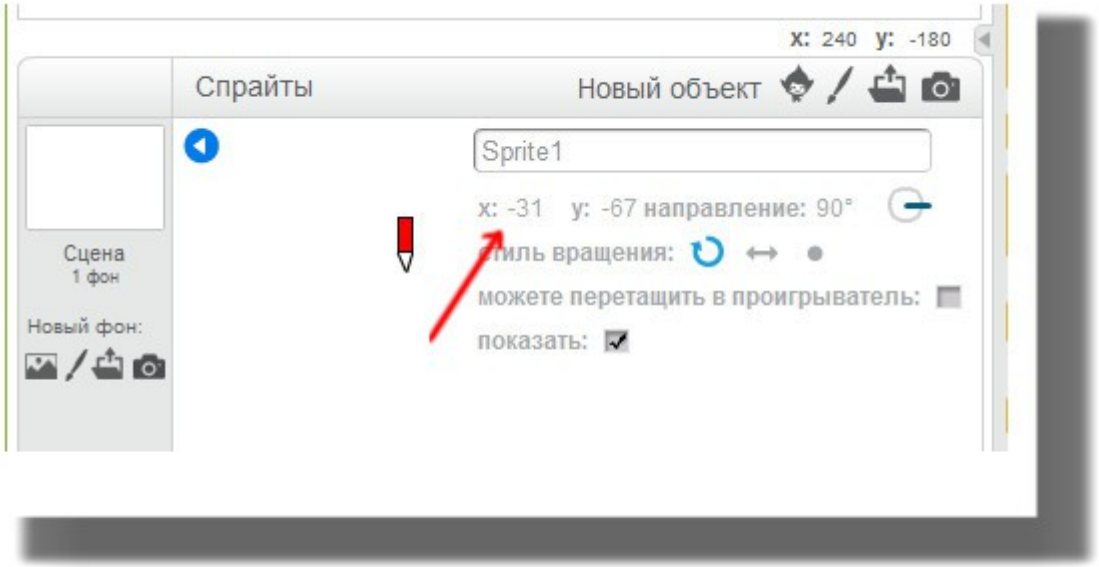
Но что делать, если мы захотим, чтобы круг все время находился в одном и том же месте, например, в центре рабочего поля? Как нам поступить?

На уроках математики вы наверняка сталкивались с понятием числовой оси и системы координат.

Рабочее поле Scratch представляет собой прямоугольную область. Если поводить курсором мыши, то в правом нижнем углу рабочей области увидим, как меняются координаты курсора по оси X и по оси Y.



Координаты исполнителя можно увидеть, если открыть информацию о спрайте:



Если ухватить мышкой исполнителя, и переместить его на новое место – координаты объекта Sprite1 X и Y изменятся.

4.2 Программа рисования координатных осей

Инструкции к выполнению: определите для начала, размеры рабочего поля, для чего проведите курсором мыши, максимально близко к краю рабочего поля. Минимальные (отрицательные значения), максимальные (положительные значения) и ноль будут теми значениями, которые необходимы для рисования осей (запишите их в тетрадь).

Чтобы не оставалось никаких лишних линий перед выполнением программы очистим экран командойочистить (темно-зеленый ящик).

Командой из синего ящика *установить x в (0)* установим значение 0 по горизонтали, командой *установить y в (...)*, установим минимальное значение по оси Y.

Дадим команду *опустить перо* и командой *перейти в x:(0) y:(...)*, перейдем в точку с координатой X равной 0, и координатой Y равной максимальному значению Y.

Ось X построена. Поднимем перо командой *поднять перо*. Теперь нужно переместиться в точку с минимальным значением координаты X и нулевым значением Y.

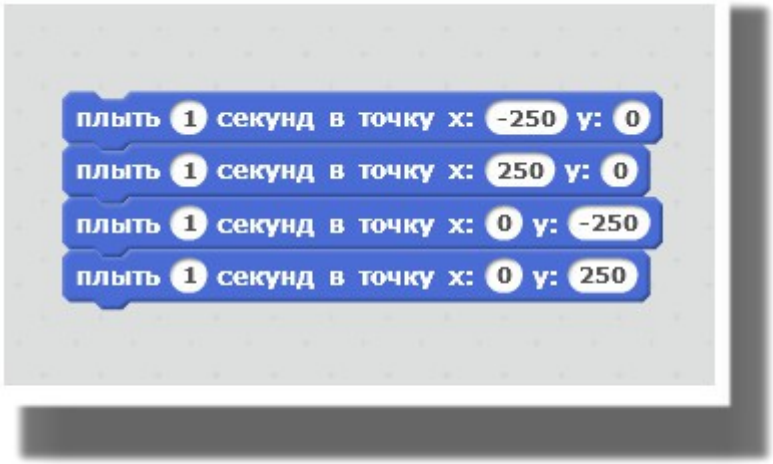
Можно воспользоваться командой *перейти в x:(...) y:(0)*, или вновь установить значения*установить x в (...), установить y в(0)*.

Опустим перо и перейдем в точку с максимальным значением X и нулевым значением Y.

Результат работы должен иметь следующий вид: см файл 02_02.sb2

4.3 Плавное перемещение объекта

Зная координаты точки можно составить программу, плавно, перемещающую объект в заданную точку экрана.



Проведите эксперименты, изменяя координаты перемещения, изменяя время перемещения. Результаты запишите в тетрадь.

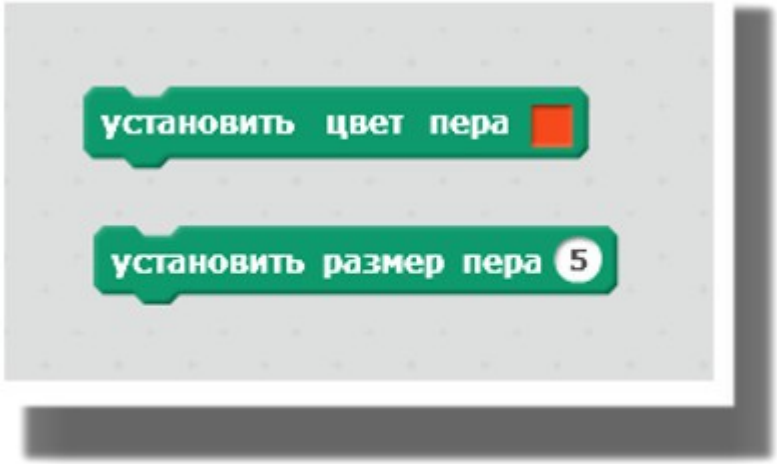
4.4 Создание простейших графических изображений на экране

Итак, чтобы создать программу, рисующую на экране некоторое изображение, достаточно задать координаты главных точек этого изображения и опустив перо дать команду переместиться последовательно в каждую из точек.

На листе в клетку, начертите систему координат, нарисуйте букву **M** и расставьте координаты каждой точки.

Составьте программы рисования других букв алфавита.

Цвет и размер пера задаются командами:



4.5 Продолжаем рисовать

Рассмотрим задание из рабочей тетради по информатике для пятого класса.

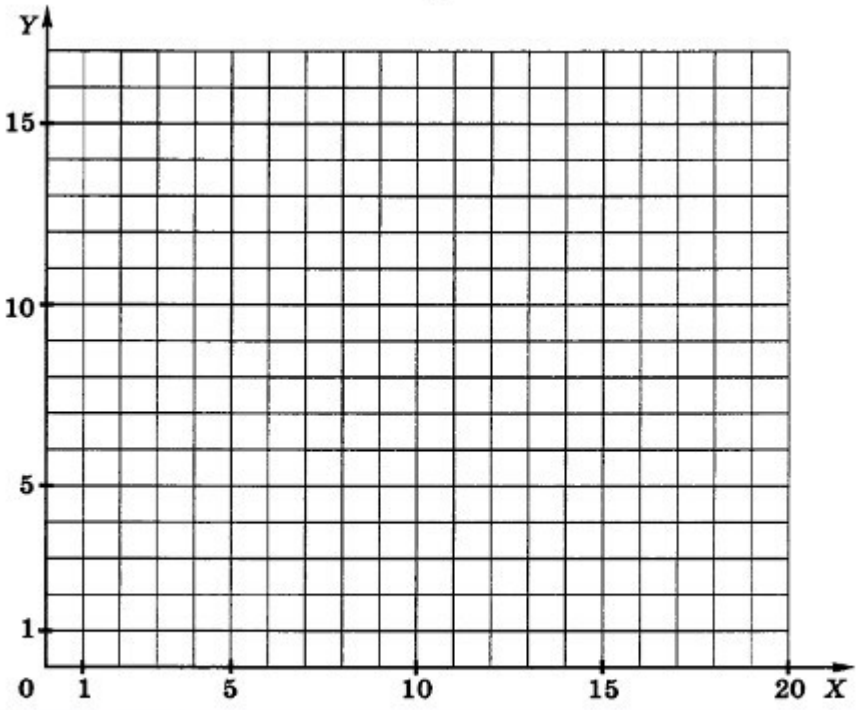
Вариант 3

Отметьте точки:

1(3,6), 2(6,3), 3(15,3), 4(18,6), 5(10,6), 6(10,16), 7(13,16), 8(12,15), 9(13,14), 10(10,14), 11(16,6).

Соедините точки:

1 – 2 – 3 – 4 – 1.
5 – 6 – 7 – 8 – 9 – 10 – 11.

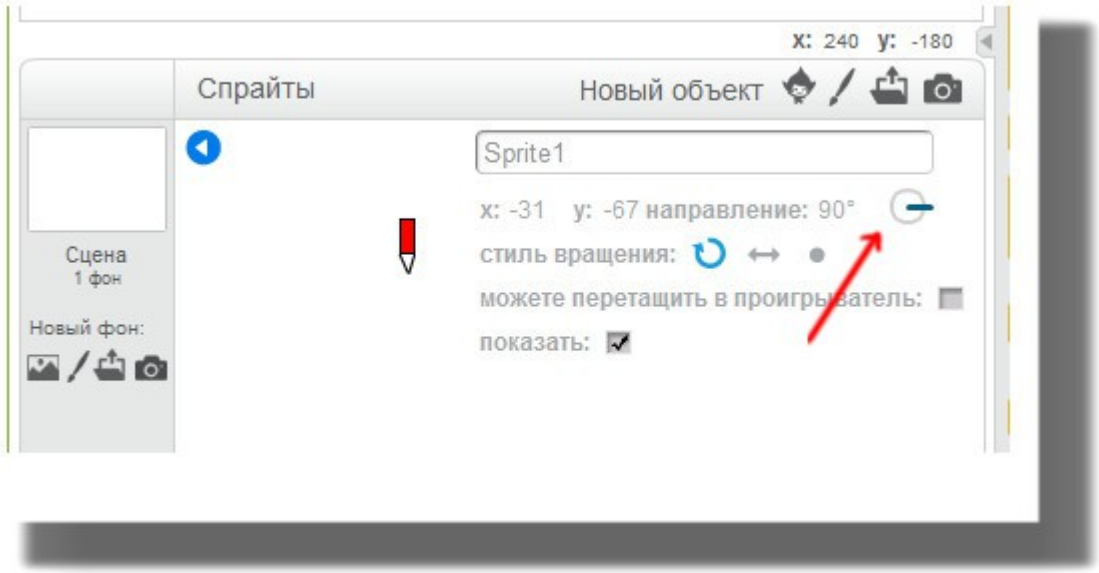


По координатным точкам, создайте программу, рисования изображения. Только обратите внимание, что для программирования в среде Scratch, шаг равный единице очень мал и результат будет плохо виден, поэтому чтобы рисунок был крупнее, все координаты увеличим в десять раз.

4.6 Программирование направления движения

Второй способ создания программным способом рисунка на экране, это изменить направление движения исполнителя и задать число шагов.

Если взглянуть на информацию о спрайте, то рядом с координатами исполнителя находится указатель направления движения исполнителя.



Если ухватить за синюю стрелку направления исполнителя и повращать, то будет изменяться значение направления в градусах.

Изменение направления задаются командами изменения направления из синего ящика:

Например, меняя параметр данной команды (угол на который будет поворачиваться наш исполнитель) и, изменяя количество шагов, можно добиться передвижения по криволинейным траекториям.

Чтобы движение было наглядным, да и вообще для отладки программы, очень удобно вставить при каждом повороте паузу командой из желтого ящика **Управление** *Ждать()*.

Нужно обратить внимание, на наиболее важные параметры данной команды: если поворот исполнителя задан 90 или -90, то исполнитель будет двигаться вправо или влево, соответственно угол 0 или 180 дадут направление вверх или вниз.

§ 5 Эпизод пятый. Вспомогательные алгоритмы. Передача управления. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=6

Тема по информатике: Информация вокруг нас. Метод координат.

Зачастую, в программе с несколькими исполнителями удобно предусмотреть управление для каждого исполнителя своим способом.

Экранная графика – передача управления

5.1 Рисуем сложную картинку

Задача: создать средствами команд рисования мультфильм «Море». На небе светит солнце, на море волны, на волнах кораблик.

Проанализируем, что мы умеем делать:

Солнышко рисовать умеем, кораблик тоже рисовали. Да и волны изобразить из нескольких полукругов пожалуй, получится... (Если что, можно взглянуть на файл 04_01.sb2) (**Шпаргалка_04.004.pdf**)

Как же все эти отдельные рисунки свести в одну общую картину?

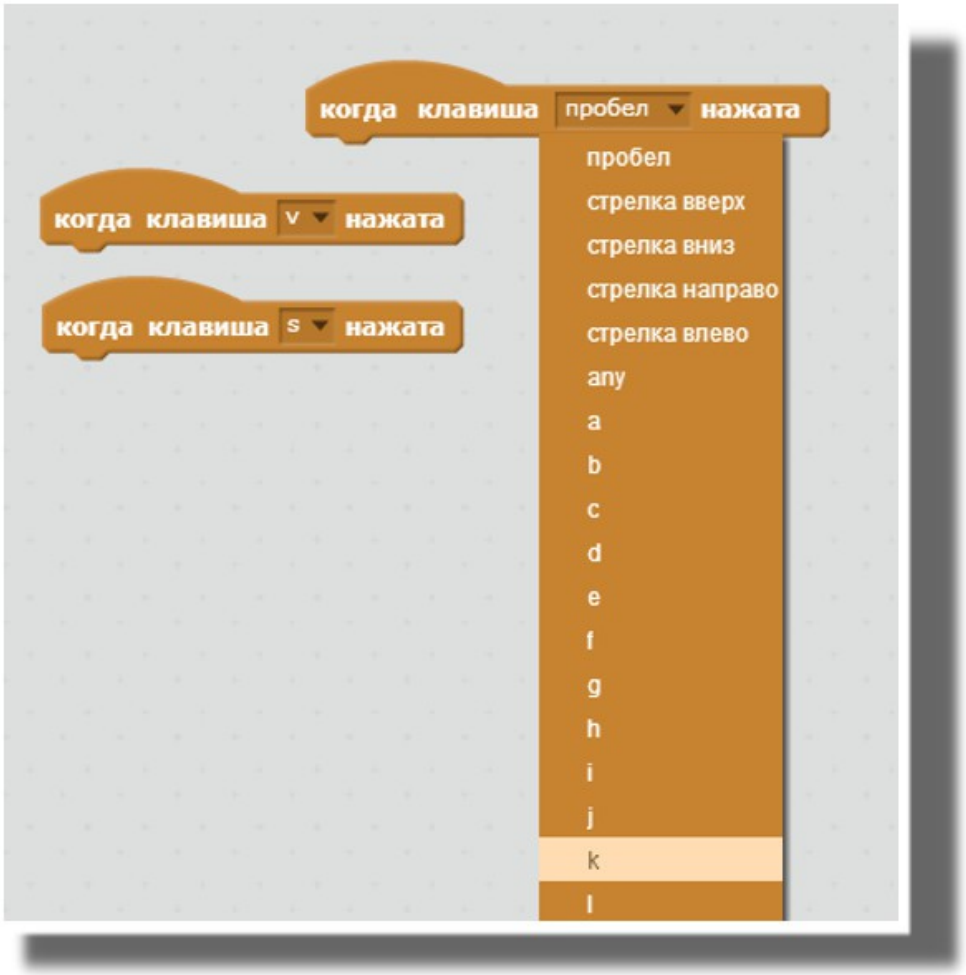
5.2 Первый путь

Для решения такой проблемы существует несколько вариантов:

Вариант первый: Все рисунки рисуются одним исполнителем.

В этом случае, чтобы не прогонять рисование всех фигур каждый раз при отладке программы, для каждой рисованной фигуры удобно задать свою собственную клавишу управления:

Например, солнышко будет рисоваться при нажатии клавиши s, кораблик, при нажатии клавиши k, волны при нажатии клавиши v. Эти команды находятся в ящике **События:**



Обратите внимание, что выбор клавиш с буквами для управления доступен только для букв в латинской кодировке (не забудьте переключить клавиатуру, когда будете тестировать программу). И после рисования каждого рисунка, не забывайте команду *Поднять перо*, чтобы при переходе исполнителя в новую точку, не оставалось лишних почеркушек на экране.

А команда *Когда щелкнут по зеленому флажку*, теперь нам нужна только чтобы очистить все наше творчество (начинаем рисовать с чистого листа) и красиво позиционировать в центр экрана нашего исполнителя.

Результат выполнения программы может иметь вид, близкий к такому:

- s – рисует солнышко,
- k – рисует кораблик,
- v – рисует волну,
- зеленый флажок – очищает все рабочее поле.,

Посмотрите результат работы в файле 04_02.sb2

В программировании такой прием называется использование **вспомогательных алгоритмов**. То есть наша глобальная задача – нарисовать красивую картинку моря, разбивается на несколько подзадач: нарисовать солнышко, нарисовать, волны, нарисовать кораблик.

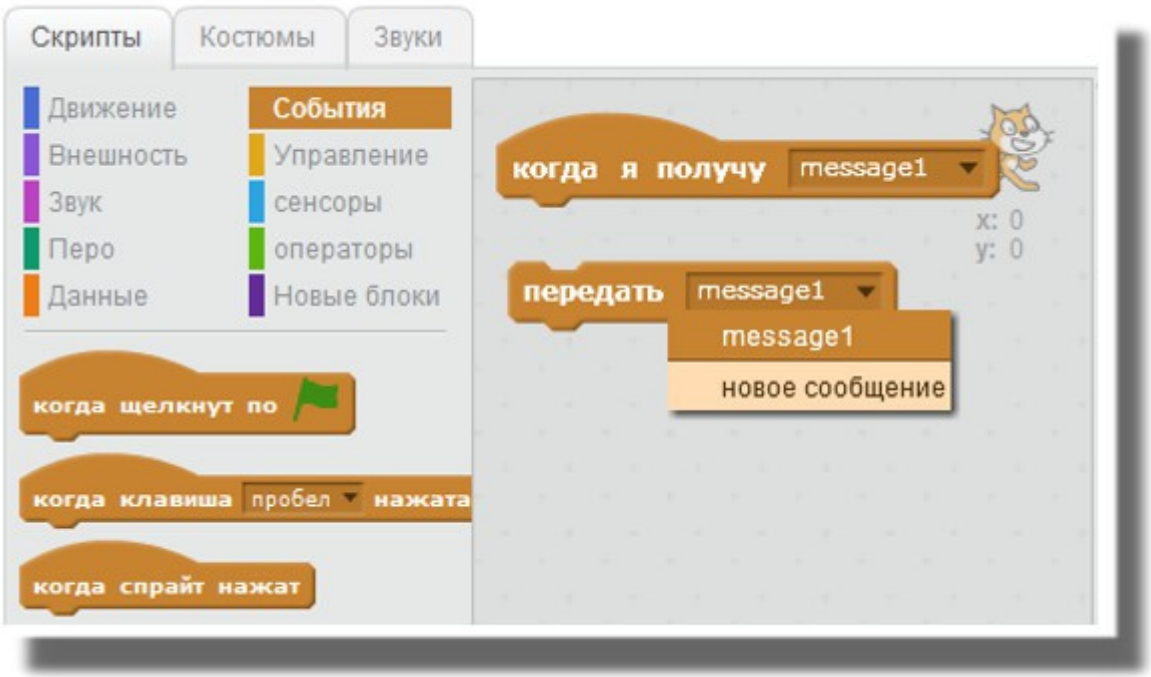
Попробуйте самостоятельно сконструировать программу рисования моря с описанным выше управлением для одного исполнителя.

5.3 Второй путь

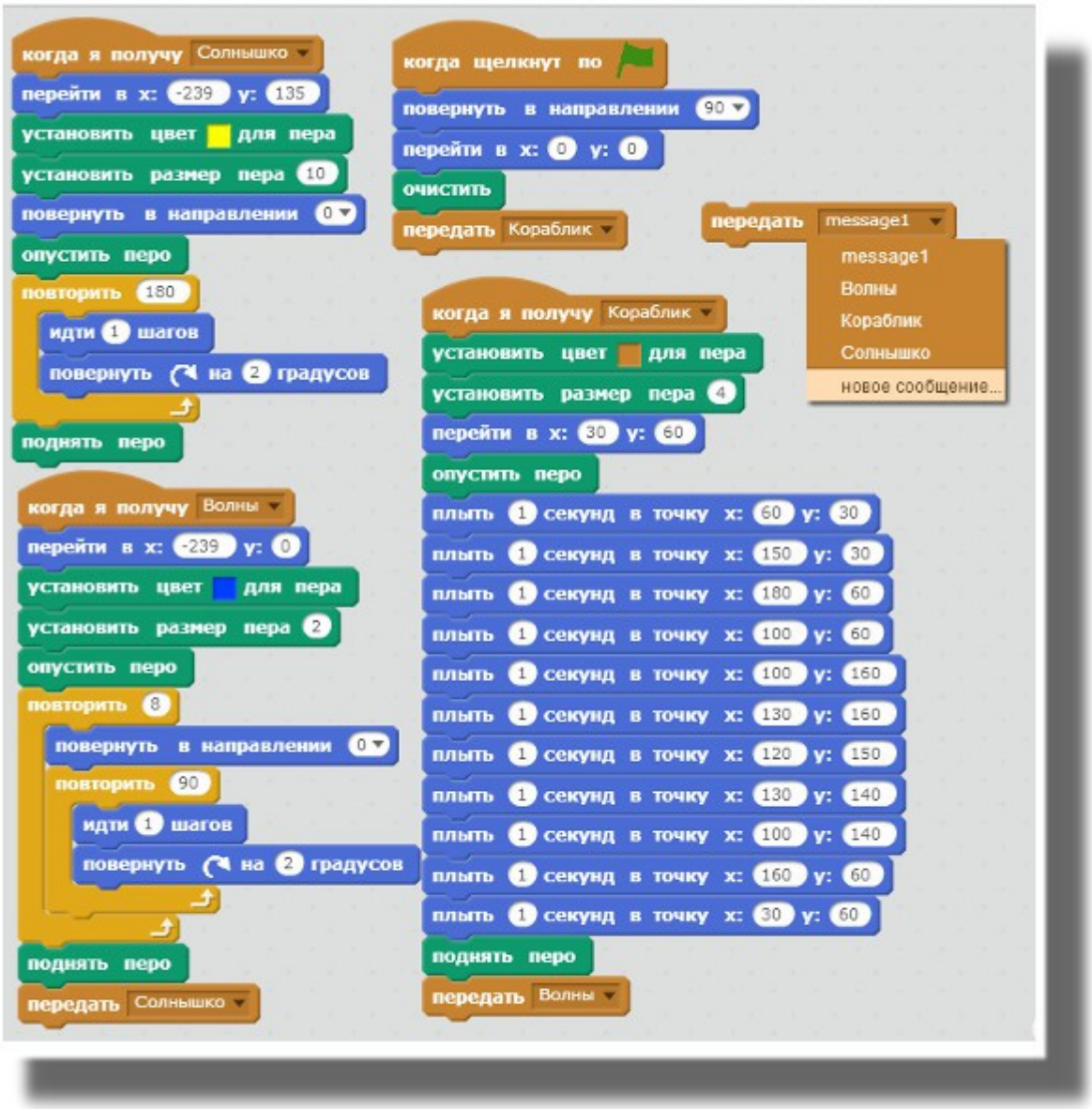
Таким образом, у нас даже получилось нечто вроде интерактивного мультфильма: почти как: «О'кей Гугл – нарисуй мне кораблик» (кстати, может у вас получится задать управление и с помощью звуковых команд? – попробуйте!)

Но хотелось бы, чтобы исполнитель сам начинал рисовать волны после того как закончит рисовать кораблик, потом нарисовал бы солнышко, а там глядишь, и еще что-то потребуется нарисовать, без нашего вмешательства.

В данном случае удобно использовать команды из ящика **События**: *Когда я получу сообщение ()* и передать сообщение(). Нужное сообщение мы придумываем сами.



Тогда программа может принять такой вид:



Комментарий к программе: при нажатии на зеленый флажок, исполнитель очищает экран и передает сообщение начать выполнение программы рисования кораблика (название процедуры *кораблик*, мы задаем сами, при выборе команды *передать новое сообщение*).

Когда исполнитель получает сообщение *Кораблик* (это результат работы команды *Когда я получу сообщение (Кораблик)*), он начинает выполнять процедуру рисования кораблика, закончив поднимает перо и передает сообщение – волны, и действия повторяются аналогичным образом.

В нашем случае, исполнитель один, сам передает, сам получает сообщения и выполняет действия, предусмотренные программой. См.файл 04_03.sb2

Но что нам мешает передавать сообщения не одному, а разным исполнителям?

5.4 Передаем сообщения разным исполнителям

Заведем три исполнителя-карандаша, раскрасим их разными цветами, и дадим ответственность одному за кораблик, второму – за волны, третьему за солнышко.

Чем это удобнее? – программа у каждого исполнителя стала короче, стало удобнее ориентироваться в общей программе – и вообще, пусть каждый делает свое дело, а мы будем руководить общим процессом!

Тогда и каждому исполнителю можно усложнять задачи: например, пусть отвечающий за солнышко еще рисует лучи, а за волны – рисует не одну волну, а много.

И программу теперь может делать не один человек, а целая команда – всем веселее!

5.5 Задание. Геометрические фигуры.

Попробуйте еще самостоятельно создать программу, где исполнитель рисует квадрат и треугольник. Задайте управление рисованием квадрата клавише k, а рисование треугольника клавише t, например:

§ 6 Эпизод шестой. Графический редактор Scratch. Растровая графика. Работа с фоном. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=7

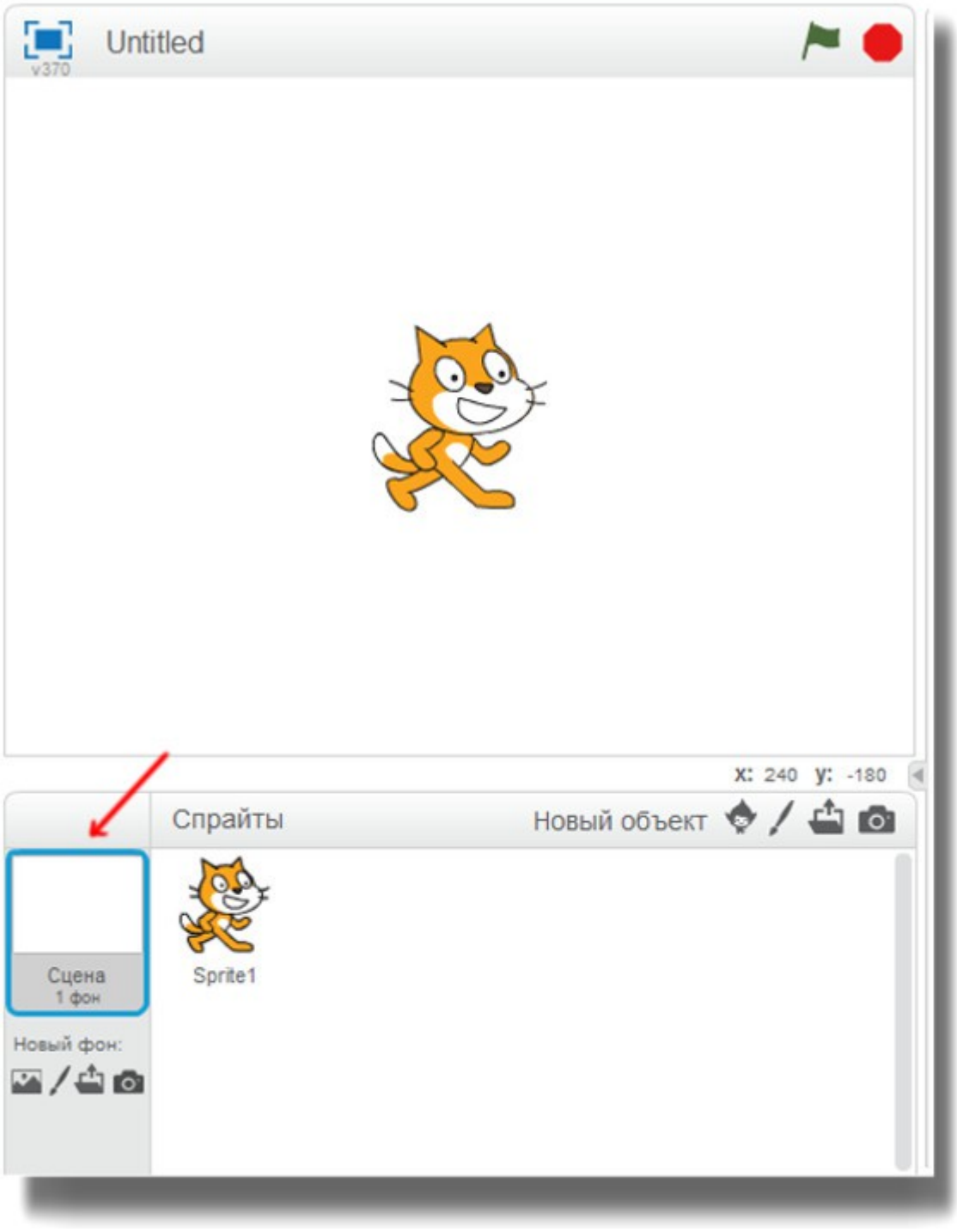
Тема по информатике: Компьютерная графика. Простейший графический редактор.

6.1 Что бы не было скучно

Мы научились немного управлять исполнителем, научились средствами векторного графического редактора создавать простого нового исполнителя, но ходит наш исполнитель на скучном пустом фоне. Попробуем сделать наше произведение более симпатичным. Добавим пространства в котором может обитать наш Исполнитель.

Запустим новый проект, оставив исполнителя-Кота.

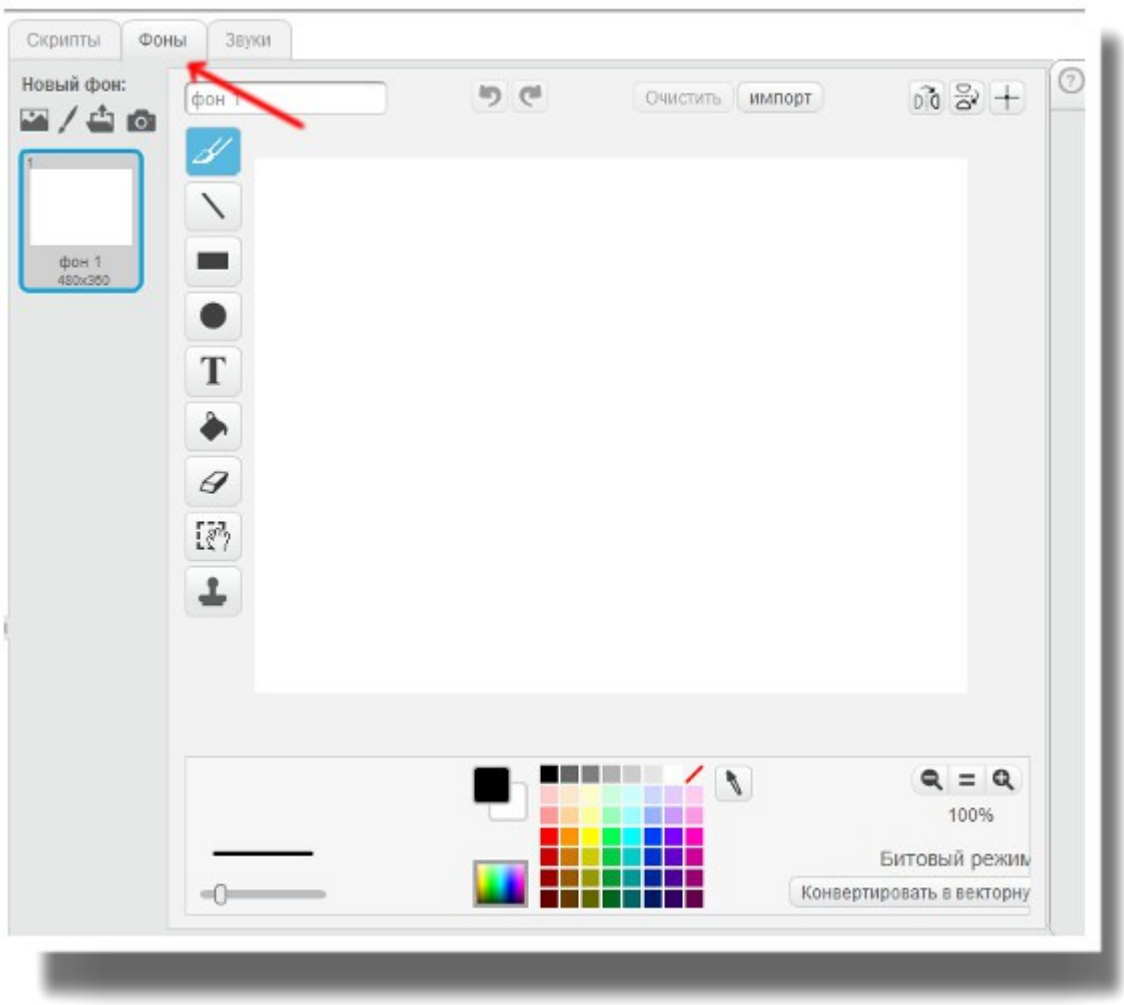
Чтобы настроить фон нужно выбрать кнопку **Сцена** в виде белого прямоугольника слева внизу.



Фон, так же как и спрайты (различные варианты Исполнителей) может быть импортируемым (вставленным из коллекции) и изменяемым объектом.

Мы можем сами нарисовать фон, на котором будет происходить действие, взять картинку из коллекции, встроенной в программу Scratch, или вообще использовать подходящее изображение или готовую фотографию.

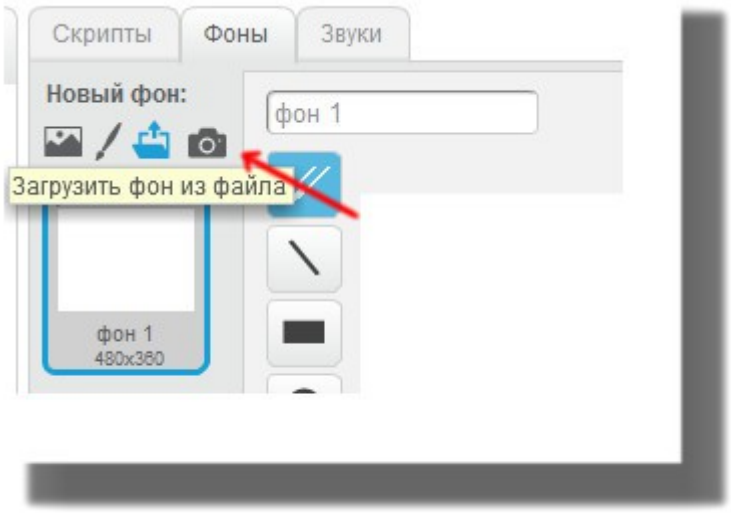
После выбора меню **Сцена** открываются вкладки для управления фоном. Аналогично работе со спрайтами на вкладке **Скрипты**, составляется программа для управления фоном, на вкладке **Фоны** осуществляется изменение внешнего вида фона, на вкладке **Звуки**, можно подобрать звуки и музыку, которые будут сопровождать наше произведение в целом.



6.2 Готовим фон и звук

Для изменения сцены на вкладке **Фоны** можно воспользоваться кнопками:

- Выбрать фон из библиотеки
- Нарисовать новый фон
- Загрузить фон из файла
- Получить новый фон с камеры
-

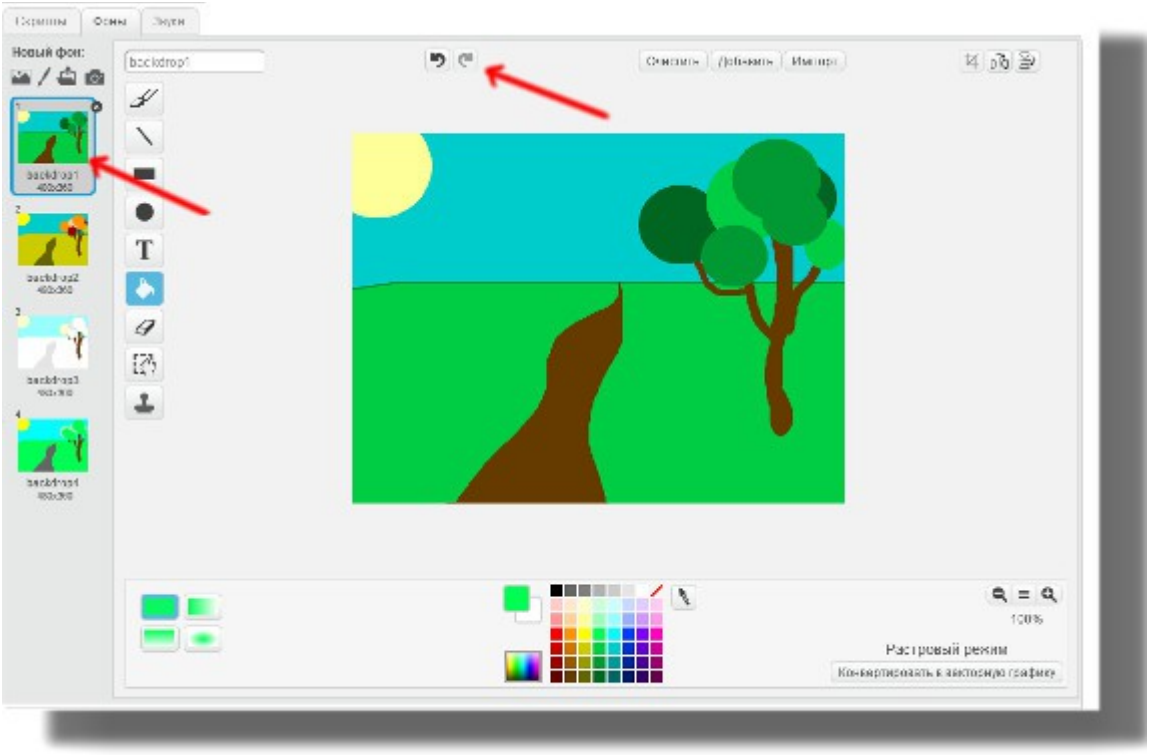


По умолчанию мы выходим в режим графического редактора, где можно самостоятельно создать изображение.

Попробуем создать программу, демонстрирующую смену времен года

Аналогично, как и в случае работы со спрайтами, мы имеем векторный и растровый графические редакторы.

На этот раз, пользуясь средствами *растрового* графического редактора, создадим изображение летнего пейзажа.



Отменить ошибочное действие можно при помощи команд *Отменить* и *Повторить* в виде полукруглых стрелок.

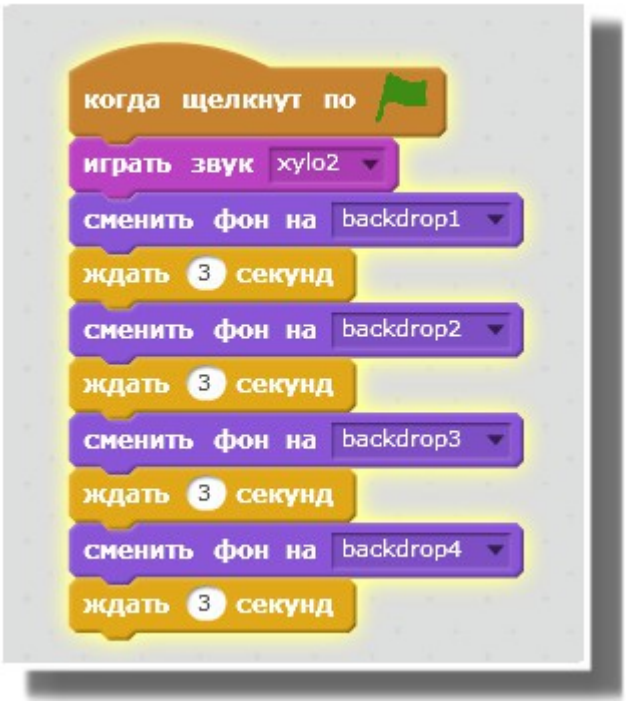
Можно нарисовать сначала летний пейзаж, щелкнув правой кнопкой на пиктограмме Фон1, выбрать команду *Дублировать* и повторить это действие четыре раза. Средствами графического редактора Scratch, каждый пейзаж преобразуем по-

очереди в осенний, зимний и весенний, то есть создадим четыре изображения, соответствующие временам года.

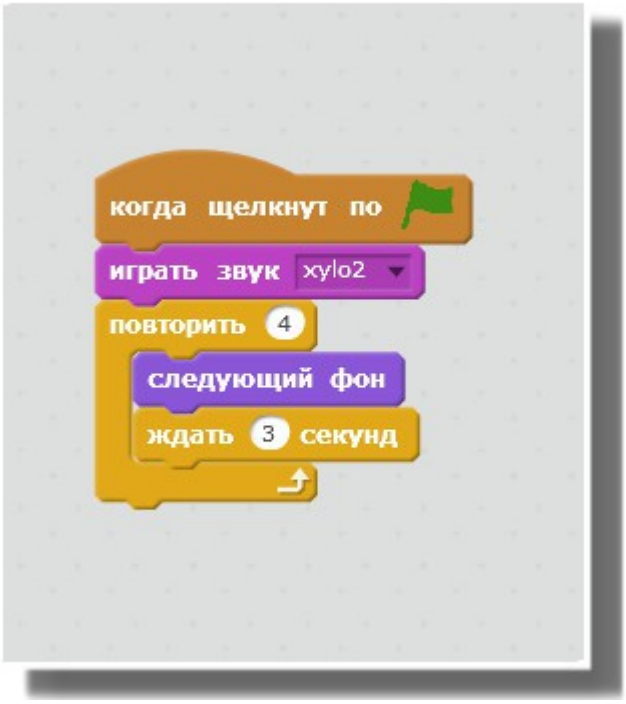
Перейдя на вкладку **Скрипты**, сконструируем программу, переключающую каждый новый фон через три секунды. .

6.3 Конструируем скрипт

Наша программа может иметь такой вид:



или такой:



Комментарий к программе:

- Первая команда – *когда щелкнут по зеленому флажку* – запускает программу на выполнение при нажатии на зеленый флажок.
- Вторая команда – *играть звук* – запускает на выполнение выбранный звук (в нашем случае мы выбрали xylo2).
- Далее наши программы слегка различаются: в первом случае реализован линейный алгоритм, – *сменить фон на ()* – переключает на определенный фон (первый, второй и т.д.) и *ждать (3) сек*.
- Во втором случае используется циклический алгоритм, повторяющийся четыре раза, причем команда смены фона (*следующий фон*), просто указывает исполнителю взять следующий фон и выдержать паузу, независимо от того с какого времени года наша программа стала демонстрировать Круглый год.

Возможный результат выполнения программы см в файле 03_02.sb2.

Вопрос: а кто в данном случае будет исполнителем? Кот у нас просто скромно стоит в уголке, карандаш тоже не рисует ничего. Кто же исполняет на команды? Попробуйте самостоятельно найти ответ на этот вопрос...

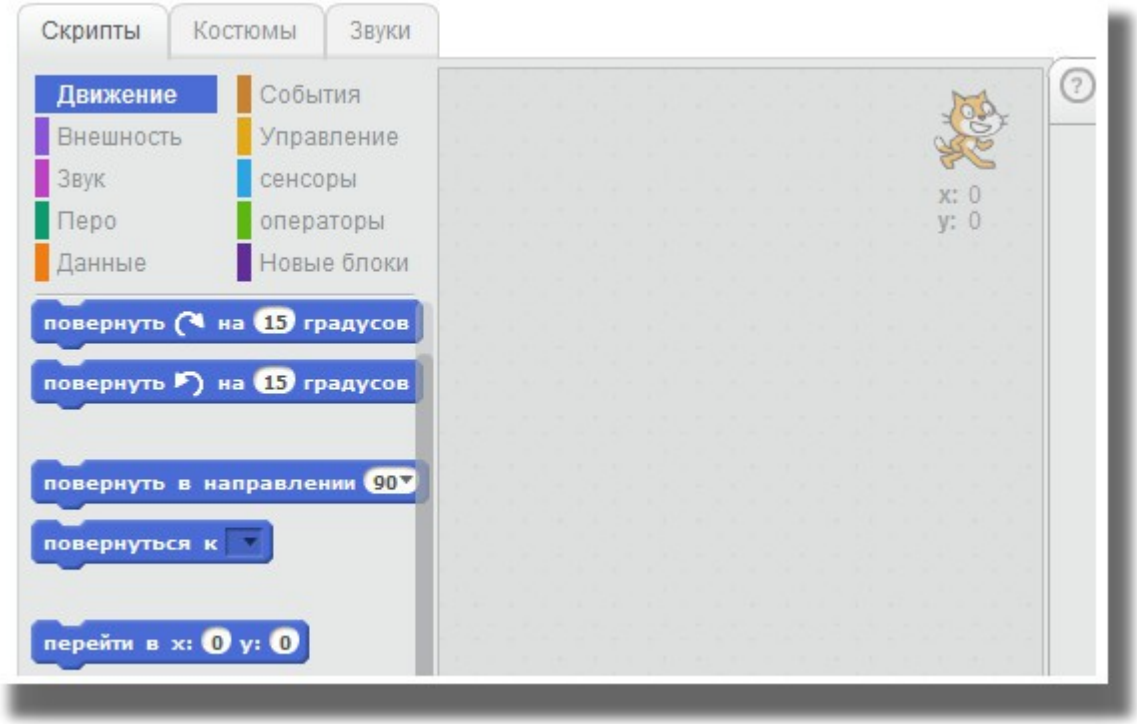
Другой вариант – из папки «Коллекции фоновых рисунков» или из личных фотографий загрузите каждое изображение: летнего, зимнего, весеннего и осеннего пейзажей.

§ 7 Эпизод седьмой. Учимся управлять объектом-Котом. 5 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=8

Тема по информатике: Информация вокруг нас. Разработка плана действий и его запись.

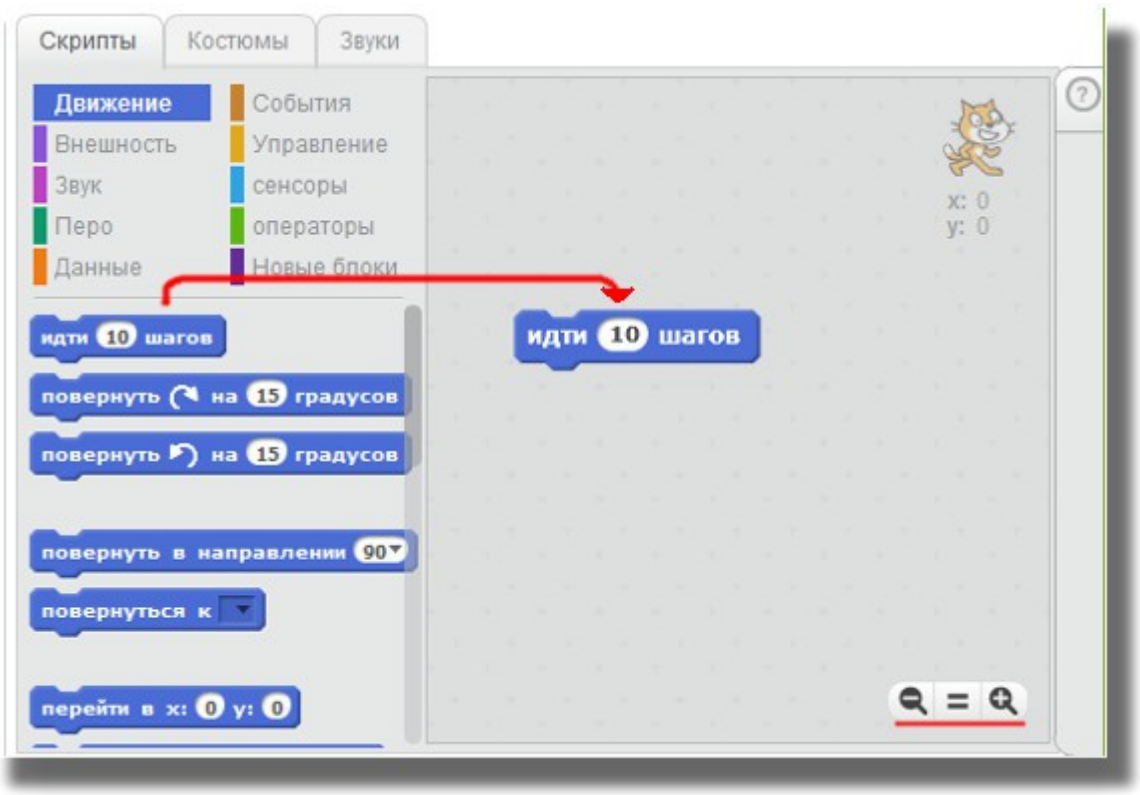
Итак, теперь самое время обратить внимание на основного исполнителя Scratch – Кота. Запустили программу, выбрали русский язык:



Как видим, по умолчанию у нас активен синий ящик **Движение**, в котором сгруппированы команды, отвечающие за движение исполнителя.

Составим нашу первую программу в среде Scratch: научим кота ходить.

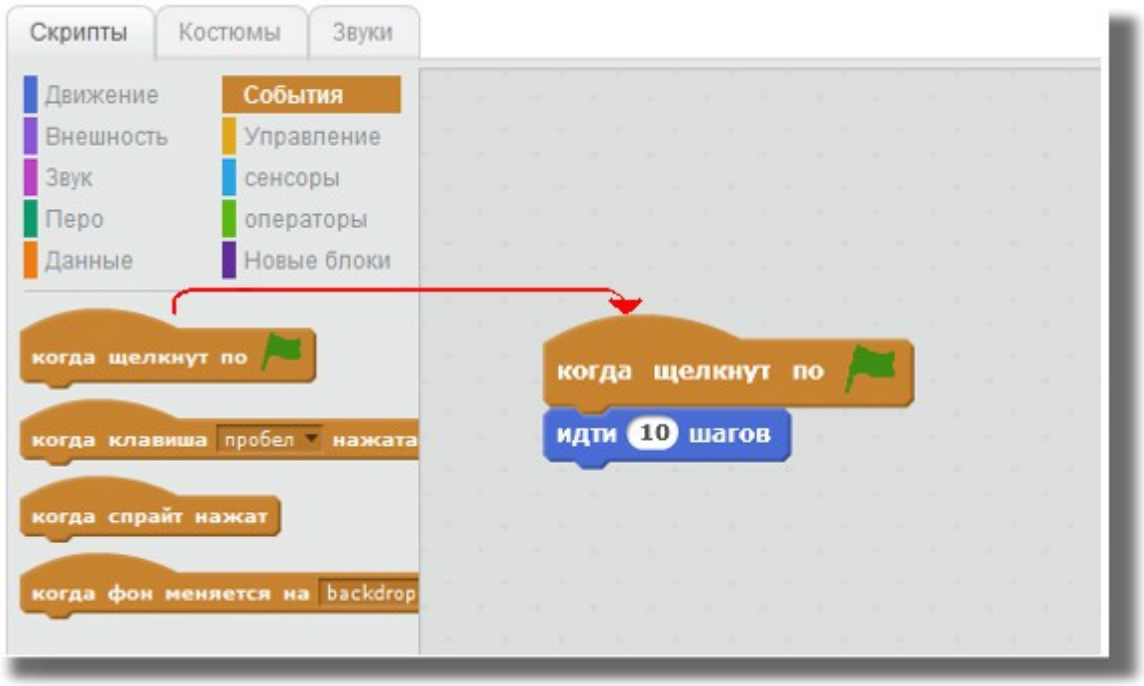
Для этого из ящика Движение, перенесем в поле скриптов команду *Идти 10 шагов*. Все, исполнителем можно уже управлять. Щелкнем мышкой, пару раз на этой команде, исполнитель-Кот чуть сдвинется на рабочем поле. Конечно он сдвинется совсем чуть-чуть, но вот такие маленькие шажки у нашего исполнителя.



Кстати, если команды кажутся слишком мелкими, увеличить отображение можно, выбрав лупу со знаками плюс или минус в правом нижнем углу. Знак равно восстанавливает отображение блоков команд по умолчанию.

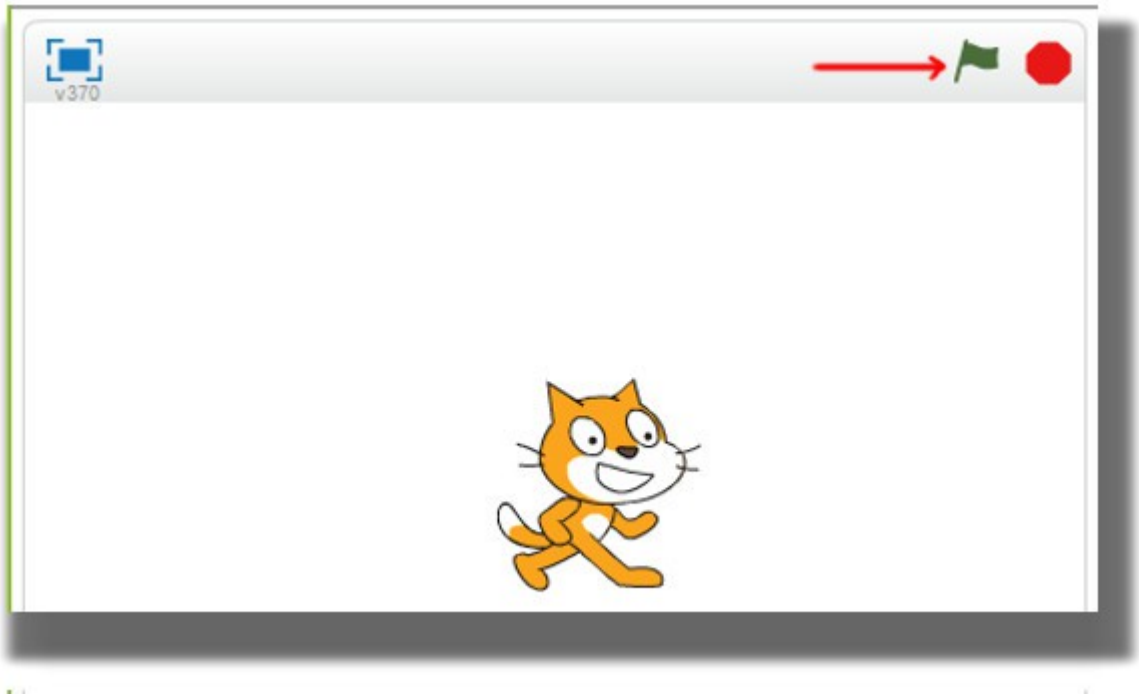
Но как-то несерьезно, представлять программу, которая состоит из одной команды. Пусть одна команда, но и она должна реагировать на определенное событие.

Посмотрим внимательнее на наши ящики – Оп! У нас есть ящик **События!** Перейдем в оранжевый ящик **События**, и перенесем первое предлагаемое событие в поле скриптов



Выберем событие: «*Когда щелкнут по зеленому флажку*». В оранжевом ящике присутствуют различные события, по которым может начать выполнение программа-скрипт: например, когда будет нажата определенная клавиша, или когда нажмем на самого исполнителя, но для начала рассмотрим универсальный запуск программы.

Мы запускаем программу на выполнение щелчком на кнопке в виде зеленого флажка.



Кстати, остановить выполнение программы можно щелчком на красной восьмигранной кнопке.

7.2 Тестируем программу

Теперь нашу самую первую программу можно протестировать.

Несколько раз нажмем на Зеленый Флаг. Наш исполнитель-Кот сдвинется на очень маленькое расстояние, то есть десять шагов – это очень небольшая величина параметра.

Изменив значение этого параметра на большее, затем на меньшее, пронаблюдаем что происходит, каким образом перемещается наш кот.

Попробуем задать отрицательное значение параметра. В какую сторону стал двигаться на исполнитель?

Задайте несколько различных значений параметра – шагов на которые должен перемещаться исполнитель, и проанализируйте результаты.

7.3 Продолжаем эксперименты

Ну, теперь можно перейти к следующим экспериментам…

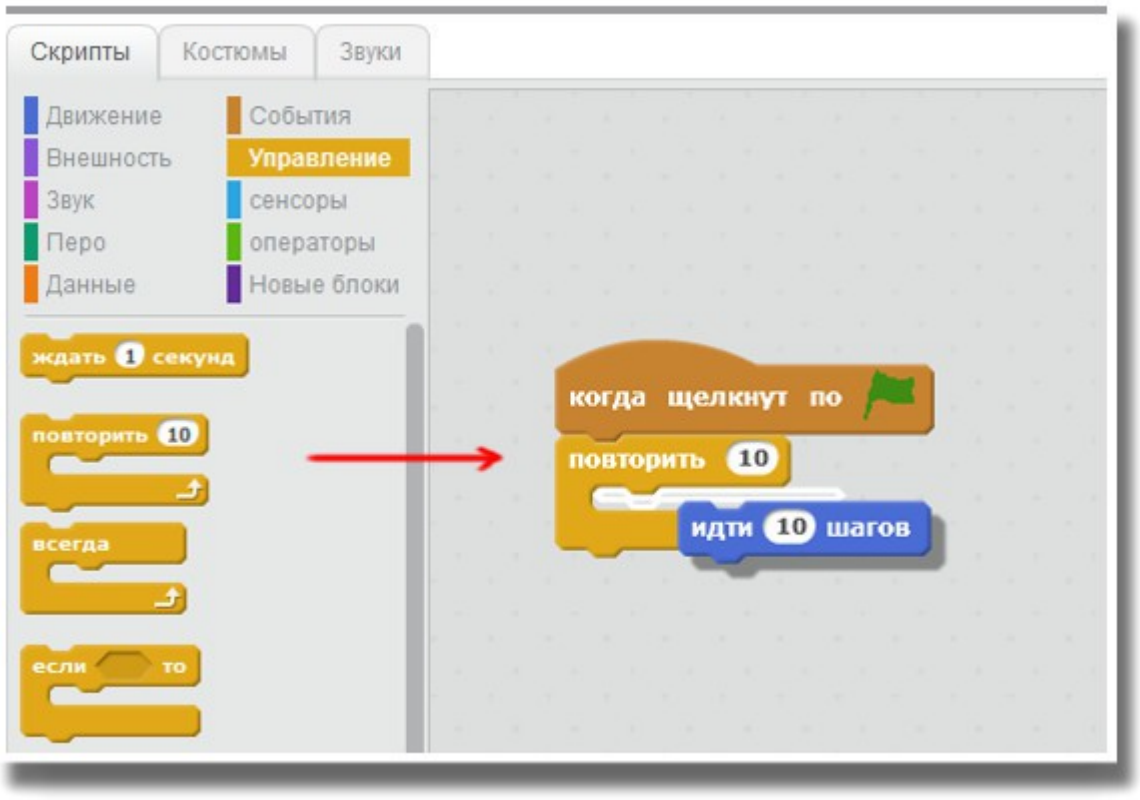
Очень хотелось бы, чтобы наш исполнитель двигался не только при понукании в виде щелчка на Зеленом Флажке, а хоть что-нибудь сделал самостоятельно (естественно под нашим мудрым руководством).

Итак, какие у нас имеются действия?

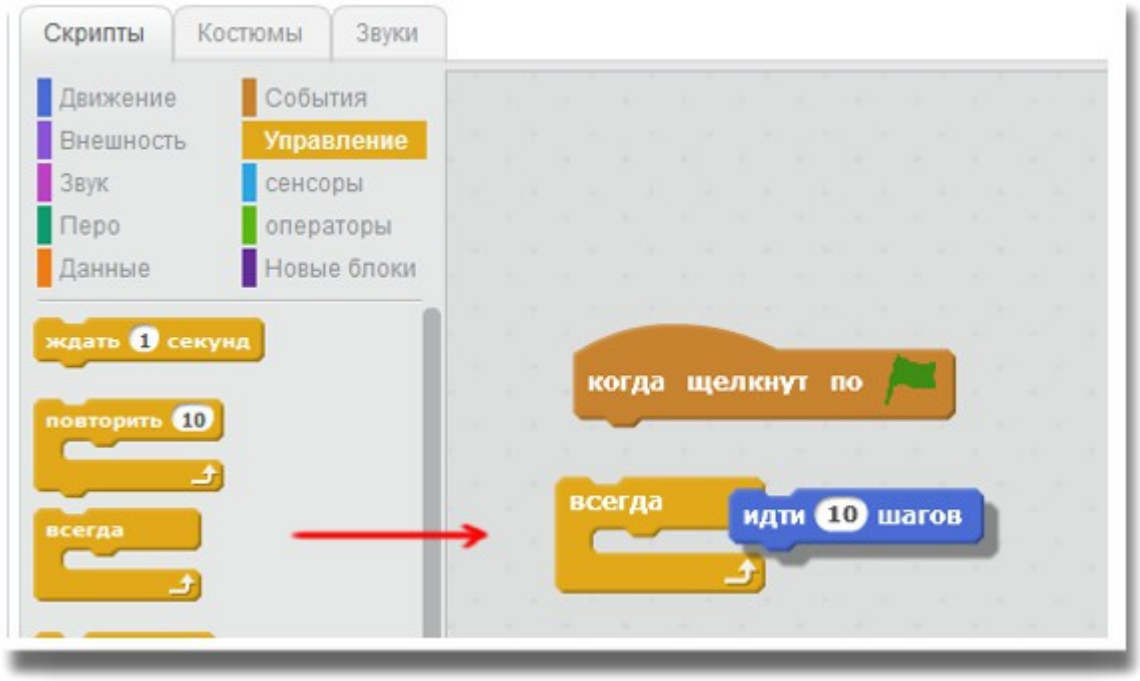
– Команда исполнителю шагнуть.

Шаг-шаг-еще шаг – мы постоянно повторяем одно и то же действие с точки зрения программирования мы имеем дело это же циклическим набором действий или циклом, когда команда повторяется несколько раз.

К командам цикла в среде Scratch относится такая команда из ящика **Управление** как *повторить()*, имеющая вид скобки: набор команд помещенных внутрь скобки команды будет повторяться столько раз, сколько указывает параметр в данном случае у нас 10 раз.



Вновь проведем серию экспериментов изменяя, теперь параметр цикла. Обычно люди все увеличивают и увеличивают число повторений, поэтому можно предложить к использованию еще одну команду, циклического алгоритма, которая будет повторять действия постоянно: *всегда*.



В данном случае Исполнитель будет постоянно двигаться вперёд, пока не упрется в стенку.

Проведем вновь серию экспериментов, останавливая Исполнителя щелчком на красном восьмиугольнике, возвращая в центр экрана, и запуская вновь после изменения параметра команды движения *идти()шагов*.

Вывод: скорость перемещения Исполнителя зависит от параметра команды *идти() шагов*.

Но все-таки хотелось бы более плавного перемещения. Попробуем вставить внутрь цикла паузу – команду*ждать*()секунд, тоже с параметром, который возможно менять.

Результат будет иметь вид:

Попробуем поизменять параметры команды *ждать* () секунд: паузу, например, поставить меньше одной секунды – одну десятую секунды (0.1). Результат: перемещение стало более плавным.

7.5 Поворачиваемся

Двигаться-то наш исполнитель двигается, но уж слишком мало он умеет: дошел до стенки и остановился, не зная, что делать.

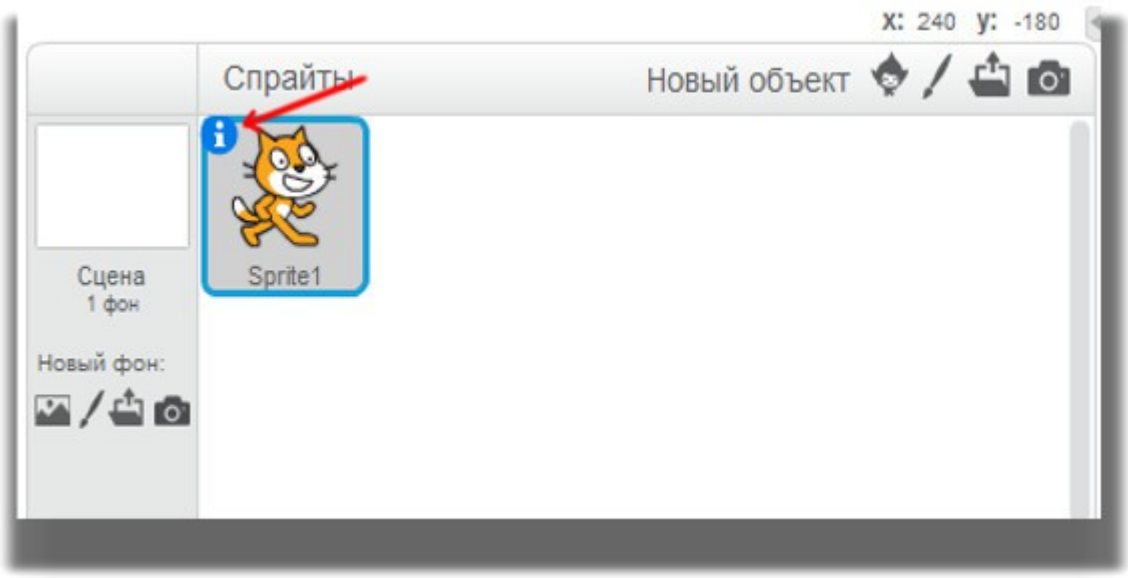
– Какой выход? – нужно нашего исполнителя учить дальше.

Аккуратно, вслух, проговорим, что бы мы хотели, от Кота: «Повторяй шаги постоянно, но если перед тобой стенка, то развернись, и иди в обратную сторону». Оп! Мы произнесли заветную фразу и у нас появилось конструкция «Если… то…» – действия будут выполняться, **Если** выполняется некоторое условие!

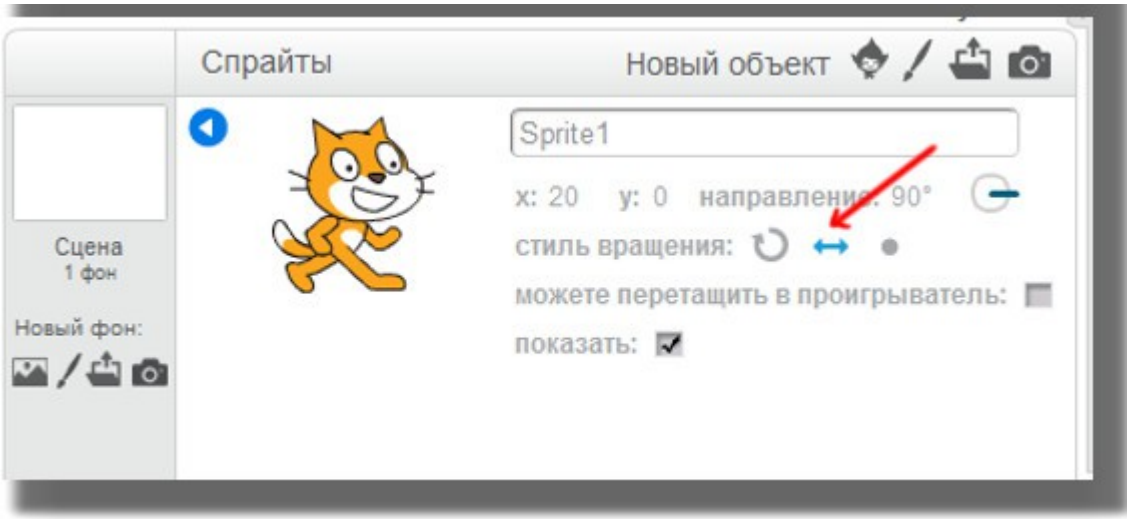
Для данного случая, когда впереди стена, в среде Scratch существует своя специальная команда, которая лежит в синем ящике **Движение** *если на краю, оттолкнуться*. Разместим ее внутри команды *всегда*:

Ура! Теперь наш Кот не замирает тоскливо у стенки, не зная, что делать ему дальше, а бегает по полю. Вот только уж очень неестественно он бегает – переворачивается вверх ногами. Чтобы избежать этого, нужно изменить некоторые свойства объекта: запретим вращение спрайта, установив возможность движения Коту только вправо-влево.

Чтобы зайти в настройки Исполнителя, в меню Спрайты, на пиктограмме Кота щелкнем на букве i (информация):



Откроется подробное меню настройки спрайта: давайте изменим свойство стиль вращения на стиль отражения.



7.6 Настройка анимации

Ну наконец, кажется, подобрали все подходящие параметры и наш кот плавно движется от одного края экрана к другому… Но все равно чего-то не хватает. Какой-то Кот не живой, лапами совсем не шевелит. Что же необходимо, чтобы мы наблюдали эффект анимации? Эффект движения возникает, когда, имеем несколько изображений слегка отличающихся друг от друга и периодично сменяют друг друга.

Сейчас нам понадобится вкладка Костюмы:

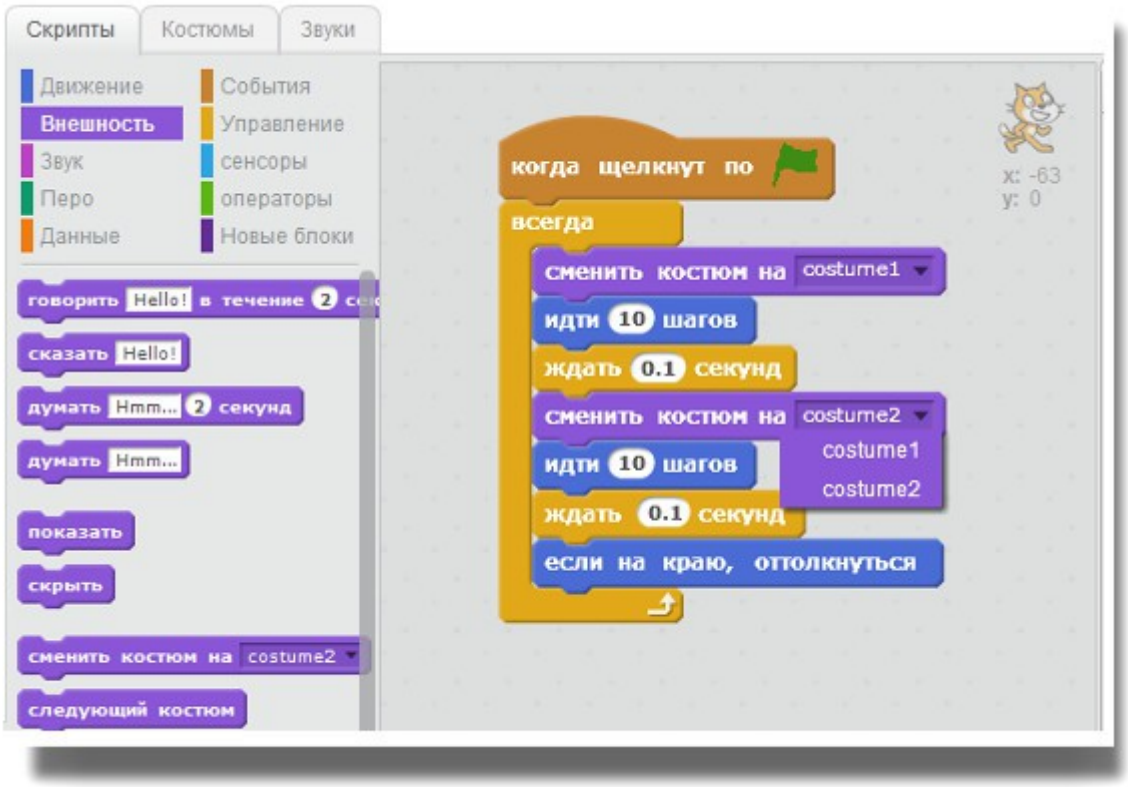
Мы видим два отличных друг от друга изображения нашего милого Кота. Если мы хотим, чтобы Кот двигался как живой, мы должны дать ему команды менять поочередно эти костюмы.

Команды управления костюмами лежат в фиолетовом ящике Внешность. Команда *сменить костюм()*, регулирует какой именно костюм будет надет на нашем исполнителе

Анализируем созданную программу: сначала должна стоять команда запуска программы, затем команда исполнителю надеть первый костюм, затем команда шагнуть, надеть второй костюм, вновь шагнуть, проверить не находится ли перед исполнителем стена и наконец вся перечисленная конструкция должна повторяться. Примерный вид программы приведен ниже:

Не забудем изменить название костюма в команде исполнителю надеть очередной костюм.

Результат может иметь следующий вид:



Не забываем сохранить результаты.

§ 8 Эпизод восьмой. Интерактивное взаимодействие объектов. 5 класс

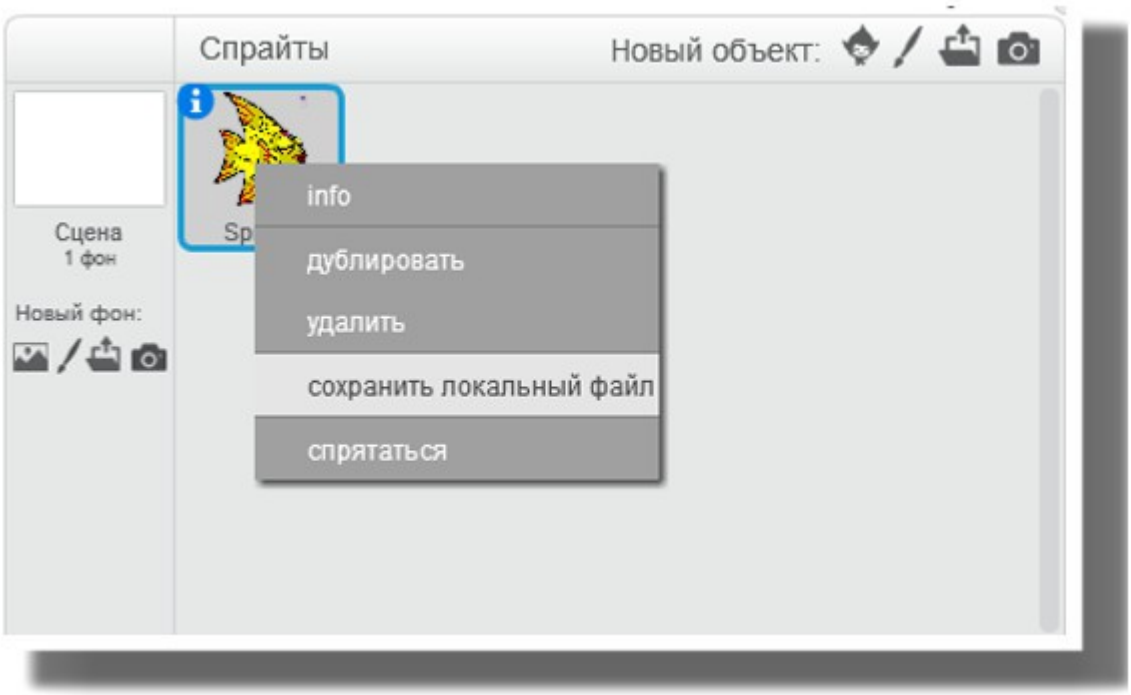
Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=9

Тема по информатике: Информация вокруг нас. Разработка плана действий и его запись.

8.1 Программирование взаимодействия объектов. Готовим объекты для для анимации

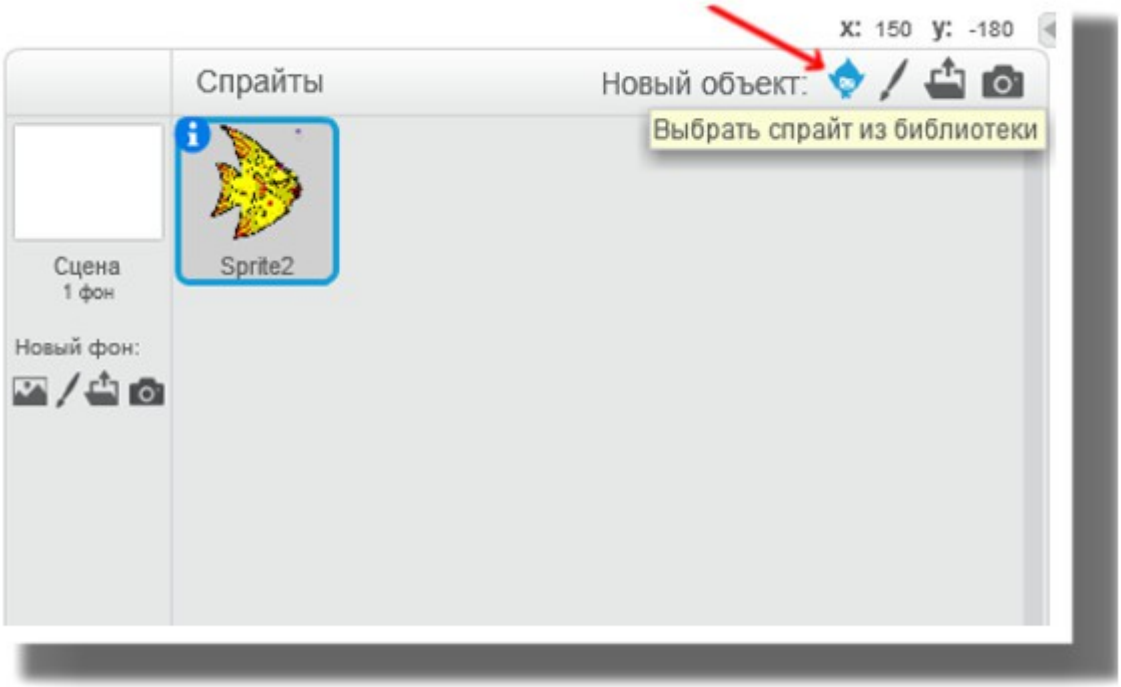
Помните сказку, про печальную рыбу-солнце, которую рассказывала мама-медведица медвежонку Умке? Но медвежонок мог только слушать, а сейчас мы сможем и показать эту сказку с помощью программы Scratch.

Изображение акулы есть в коллекции спрайтов, а вот печальную рыбу-солнце, которая не может увернуться от зубов акулы, придется подготовить самим, нарисовав яркую желтую рыбу, или раскрасив подходящее изображение. (Впрочем, эту рыбу-солнце, можно загрузить из спрайта 05_01.sprite2)

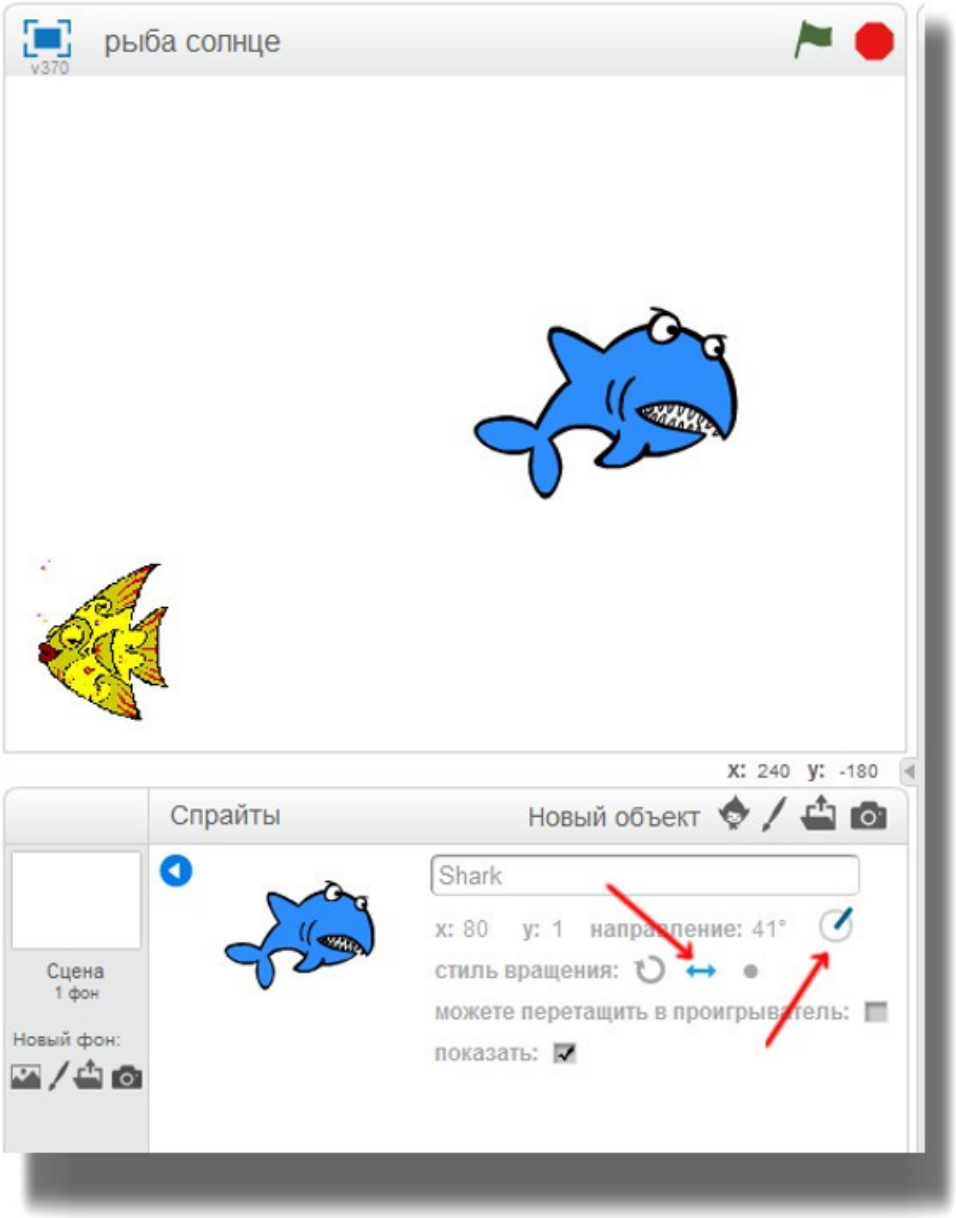


Итак, запустим программу Scratch. Удалим исполнителя Кота и загрузим сохраненную рыбу-солнце, или любую рыбу из коллекции спрайтов и раскрасим ее в желтый цвет.

Затем добавим новый спрайт – Акулу, щелкнув по кнопке – Выбрать спрайт из библиотеки.



Создадим программу постоянного движения Акулы от одного края экрана до другого. Скорость движения зададим не очень большую, а чтобы акула не переворачивалась вверх животом, вращение в свойствах спрайта, заменим на отражение



Вдобавок, чуть изменим направление, благодаря этому, наша акула станет плавать по всему рабочему полю, а не только вправо-влево.

8.2 Настраиваем движение объектов

Программа для акулы может выглядеть таким образом:

Для рыбы-солнца программы выглядит подобным же образом, только скорость ее будет меньше – значит нужно уменьшить количество шагов, либо добавить паузу команду *ждать()*.

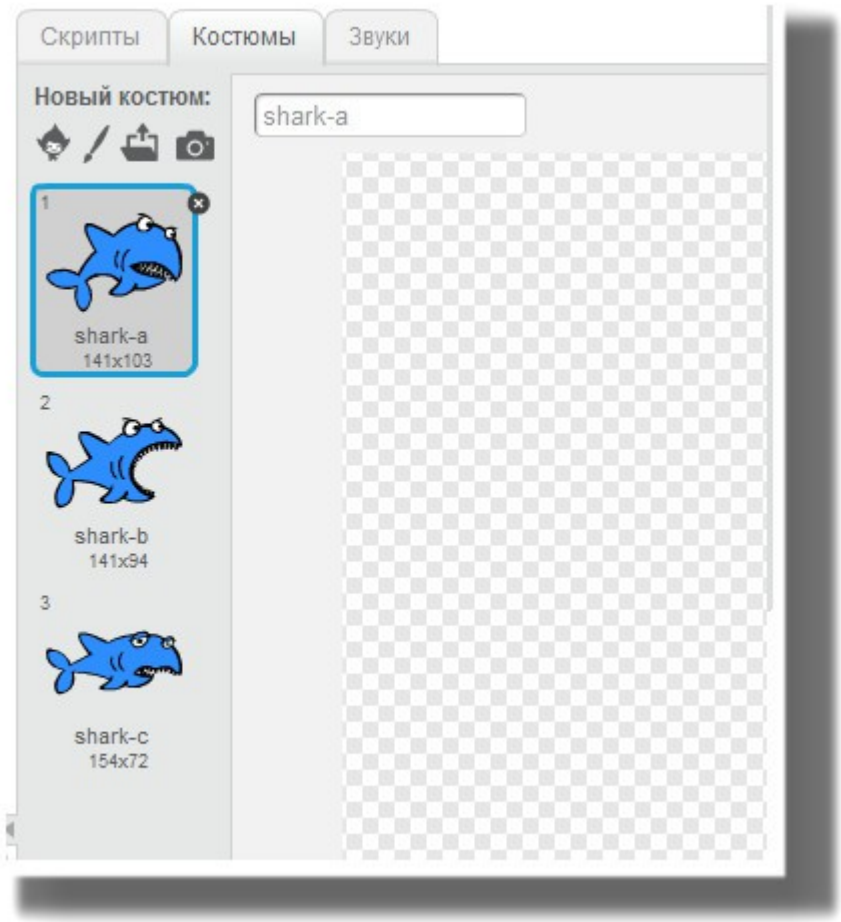
И плавать она будет у нас только прямо – поэтому можно ей программно задать направление: сразу после начала выполнения программы из синего ящика выберем команду: *повернуть в направлении 90о*.

Таким образом, **для рыбы программа** будет выглядеть примерно так:

8.3 Добавляем действия акуле и рыбе

Теперь, настроим программу таким образом, чтобы при приближении к рыбе акула открывала пасть. Для этого нам необходимо запрограммировать работу других костюмов акулы.

Перейдем на вкладку редактирования костюма и убедимся, что для выбранного спрайта акулы присутствуют необходимые костюмы:



Теперь четко сформулируем, что же именно мы хотим: пусть как только акула коснется рыбы – акула откроет свою пасть, затем съест эту несчастную рыбу-солнце, т.е. рыба исчезнет и сытая акула поплывет дальше..

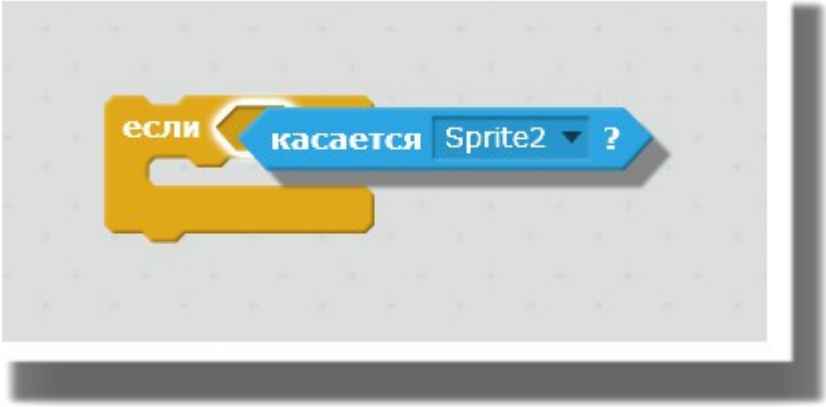
Итак, что нужно сделать?

Если Спрайт1 (акула) коснулся Спрайт2 (рыба-солнце), тогда объект акула должен сменить свой костюм на shark1-b.

Так-так-так – да ведь это же у нас условный алгоритм: если…тогда!

Находим в желтом ящике **Управление** конструкцию *Если...то*, причем видно, что для условия специально заготовлено место, куда можно вставить условие только определенной шестиугольной формы.

Из ярко-голубого ящика **сенсоры** выберем условие соприкосновения с объектом рыба (*касается Sprite2*), и вставим голубой шестиугольник в конструкцию условного алгоритма.



Изменим программу для акулы, и, поскольку, в процессе, она должна надевать второй костюм, предусмотрим, чтобы сразу после запуска программы, акула всякий раз была в первом костюме, для чего в начале программы используем команду *сменить костюм на (shark1-a)*.

Теперь у нас, как только акула коснулась рыбы, она раскрыла свою страшную пасть и поплыла с разинутой пастью дальше. Но мы хотим, чтобы акула съела эту несчастную рыбу-солнце, и закрыв рот, сытая, плыть дальше.

Ну, на самом деле, компьютерный объект-акула съесть компьютерный объект-рыбу никак не сможет, а вот послать сообщение, чтобы объект-рыба исчезла, стала невидимой – это мы умеем.

Итак, после соприкосновения объектов выдержим небольшую паузу, создадим с помощью команды *передать сообщение* сообщение *am*, и отправим его рыбе. После получения этого сообщения объект-рыба станет невидимым, а объект-акула наденет третий костюм, будет сытая и довольная.

Для этого нам потребуются еще две команды из фиолетового ящика **Внешность**:

скрыть – объект становится невидимым

показать – объект сделать видимым

Теперь, после встречи со злой акулой рыба исчезает навсегда, и даже после запуска программы снова, ее не видно, поэтому, чтобы всякий раз при запуске программы печальная рыба-солнце была бы видна вновь, в начале программы для рыбы-солнце добавим команду *показать*.

Итоговый скрипт для акулы:

Итоговые скрипты для рыбы:

8.4 Наводим глянец

Далее вставим или нарисуем фон в виде морского дна, добавим морских обитателей и зададим им разнообразные виды движения.

Примерный вид рабочего поля программы может быть таким как в файле 05_01.sb2.

Задание: выполните описанные в эпизоде действия, причем исполнители, не обязательно могут быть те, которые вы только что прочитали: например это может быть летучая мышь, которая поедает бабочку, или дракон, сжигающий теннисный мячик.

§ 9 Эпизод девятый. Активные и пассивные действия объекта. Программирование прямоугольника. 6 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=10

Тема по информатике: Объекты и системы. Признаки объектов: свойства, действия, поведение, состояния.

Ключевые слова:

1.Объект.

2.Свойства объекта.

3.Активные действия объекта.

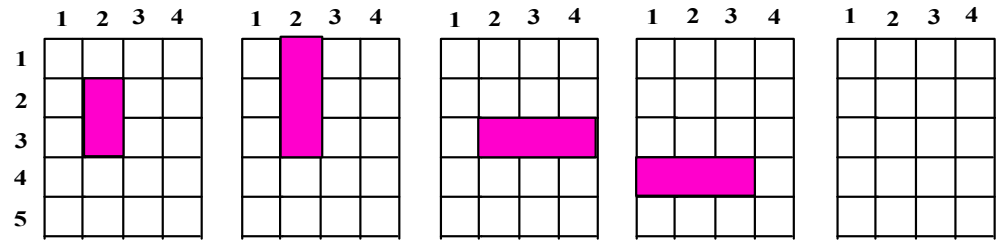
4.Пассивные действия объекта

9.1 Управление действиями объекта

Рассмотрим прямоугольник как некоторый объект, если подходить с точки зрения того что мы уже изучили, то данный объект может характеризоваться следующими параметрами: у него может изменяться размер – т. е. прямоугольник может занимать одну-две-три и более клеток; у него может меняться положение – может располагаться горизонтально или вертикально, и он может находится в некотором месте поля, определяемым столбцом и строкой.

Будем считать, что положение вертикально соответствует положению «встать», а горизонтальное положение соответствует положению «лечь»

Заполните таблицу, расписав, каждое состояние прямоугольника и нарисуйте соответствующий объект в пятом состоянии

					
Атрибуты объекта	Первое состояние	Второе состояние	Третье состояние	Четвертое состояние	Пятое состояние
СТОЛБЕЦ	2		2		4
СТРОКА	3		3		5
РАЗМЕР	2				4
ПОЛОЖЕНИЕ	«встать»		«лечь»		«встать»

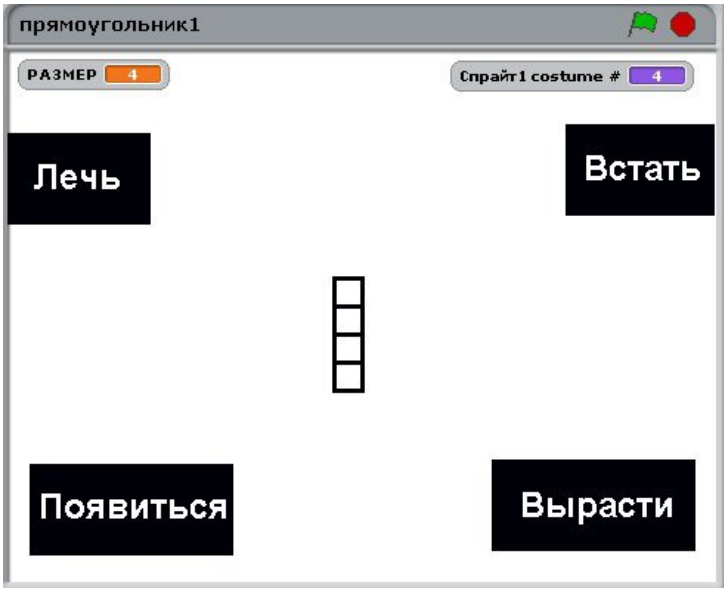
Давайте проанализируем результаты нашей работы: и попробуем спланировать работу такой программы: имеется объект прямоугольник который может менять свое состояние, по командам, которые подаются с помощью кнопок, соответствующих командам.

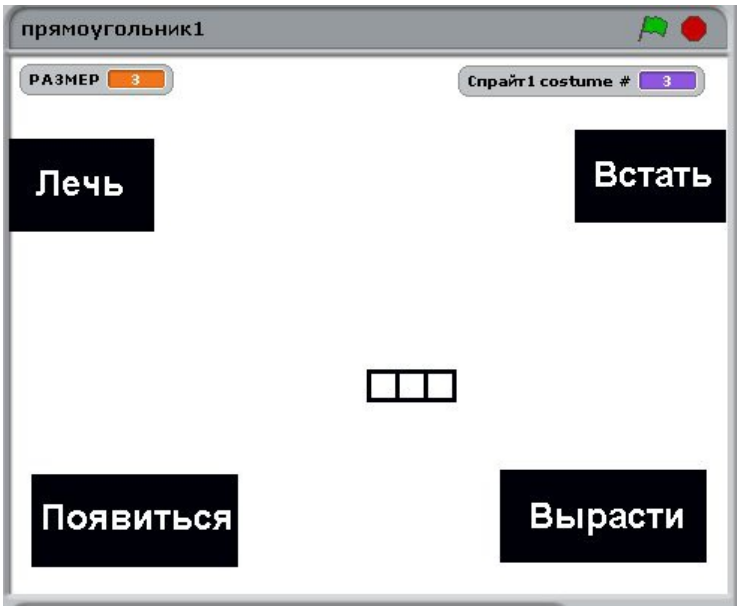
Рассмотрим состояния прямоугольника: чем отличается первое состояние от второго?

Какие объекты имеются в программе?

Какие атрибуты прямоугольника и каковы могут быть значения этих величин?

Какие действия предусмотрены в программе?





Ясно, что на прямоугольник мы можем воздействовать командами: Появиться, Вырасти, Лечь, Встать, и для объекта-прямоугольника это будут пассивные действия. В свою очередь, прямоугольник может появляться, изменять размер, принимать вертикальное и горизонтальное положения. И для прямоугольника это будут уже активные действия.

Давайте попробуем разработать в среде Scratch, программу управления объектом-прямоугольник

Для программирования нам могут потребоваться следующие подсказки:

При запуске программы (щелчке на зеленом флажке) прямоугольник должен исчезать. Необходимые команды – **(Ящик Внешность)** **Спрятать** объект – он становится невидимым **Показать** объект – сделать видимым	Прямоугольник получает сообщения и изменяет свое состояние: **Необходимые команды – ((Ящик События)): Когда я получу сообщение ()**
Рассмотрим События (действия): **Появиться, Вырасти, Лечь, Встать** Каждое из событий передает сообщение	

объекту Прямоугольник: Необходимые команды – (**Ящик События)**: **Передать** ()	
Событие **Появиться** **(**выполняется только один раз**)** Прямоугольник должен стать видимым, оказаться в центре поля, и надеть первый костюм (одна клетка). Необходимые команды: **Показать** – Показать объект – сделать видимым **Перейти x: () y: ()** **Сменить костюм на ()**	Событие **Вырасти** Прямоугольник надевает следующий костюм Необходимые команды **(Ящик Внешность)** **Следующий костюм**
Событие **Встать** Прямоугольник принимает вертикальное положение Необходимые команды (**Ящик Движение**) **Повернуть в направлении ()**	Событие **Лечь** Прямоугольник принимает горизонтальное положение Необходимые команды (Ящик Движение) **Повернуть в направлении ()**

§ 10 Эпизод десять. Моделирование работы системы объектов в среде Scratch. Фантастическое животное Кирт, как система объектов. 6 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=11

Тема по информатике: Объекты и системы. Системы объектов. Система и окружающая среда.

Возьмем достаточно сложный объект и рассмотрим его как систему. Например, если мы будем рассматривать велосипед как систему объектов, то мы можем назвать множество элементов, из которых будет состоять эта система. Причем, мы можем воздействовать на отдельные элементы и каждый элемент может совершать определенные действия: на педаль мы можем давить сверху вниз, колесо может вращаться по часовой стрелке.

В свою очередь, в целом у системы «Велосипед» появится новый вариант действия – когда мы нажимаем на педаль вертикально вниз, и удерживаем руль, вся система совершает поступательное движение вперед.

Таким образом мы можем наблюдать **системный эффект: появление у системы свойств, которыми не обладают элементы системы в отдельности.**

Теперь попробуем запрограммировать сами работу некоторой системы объектов. Рассмотрим существо Кирт – вымышленное животное с пятью щупальцами[3]. Длинным щупальцем с ярко-красным жалом Кирт ловит и жалит добычу, а на каждом из коротких

3 Идея задания почерпнута у А.В.Горячева "Информатика в играх и задачах 5 класс" Баласс 2013

щупалец у него рот, в который он захватывает добычу, увеличиваясь в размерах при каждой пойманной рыбке.

Рассмотрим деятельность этого существа как работу системы. Давайте подумаем, из каких элементов должна состоять системы и каким образом должен работать каждый элемент системы, чтобы наблюдался системный эффект:

Что мы наблюдаем: первое – объект рыба, которая случайным образом перемещается по рабочему полю, второе объект Кирт который состоит из собственно туловища, нескольких щупалец с ртами, и одним щупальцем с ядовитым жалом.

Кирт увеличивается в размерах каждый раз задевая ядовитым щупальцем рыбку.

Попробуем ответить на следующие вопросы:

- Какими свойствами обладает объект Рыба?
- Какие действия может выполнять объект Рыба?
- Какие объекты составляют подсистемы хищника Кирт?
- Какие свойства объекта Кирт?
- Какие действия объекта Кирт?
- Какие свойства объекта Ядовитое щупальце?
- Какие действия объекта Ядовитое щупальце?
- Какие свойства объекта Щупальце?

- Какие действия объекта Щупальце?

§ 11 Эпизод одиннадцать. Робот как натурная модель. 6 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=12

Тема по информатике: Алгоритмика. Понятие исполнителя. Назначение, среда, режим работы, система команд исполнителя.

Ключевые понятия:

SmartCar как формальный исполнитель.

Непосредственное и программное управление исполнителем.

Управление платформой SmartCar с помощью пульта управления.

Принцип хранимой программы.

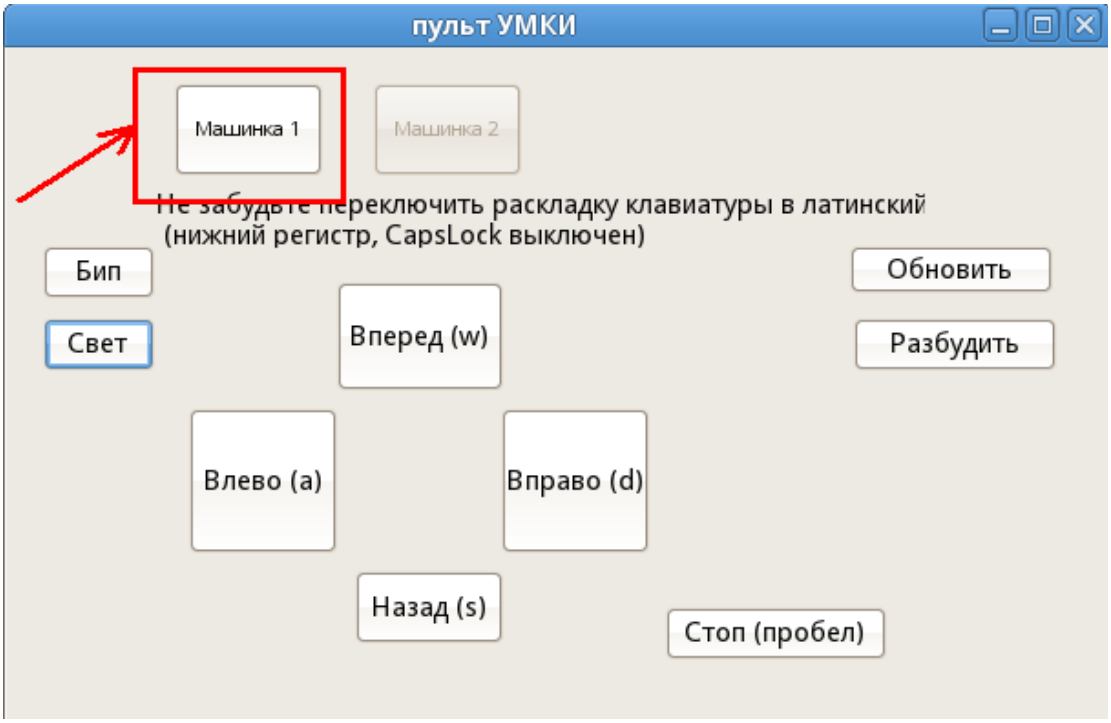
Соответствие информационной и натурной модели. Исполнитель Робот как информационная модель; исполнитель SmartCar, как натурная модель – анализ сходства и и различия.

Обеспечение занятия:

- роботизированная платформа SmartCar;
- USB-адаптер;
- ноутбук;
- программное обеспечение Smartcar.exe;

11.1 Простейшее управление платформой

Объяснение способа запуска программы, знакомство с пультом управления. Первое знакомство лучше начать с уровня Start.



Пульт управления одной платформой. Уровень Start.

От простого управления платформой SmartCar с поворотами и движением вперед-назад, можно перейти к:

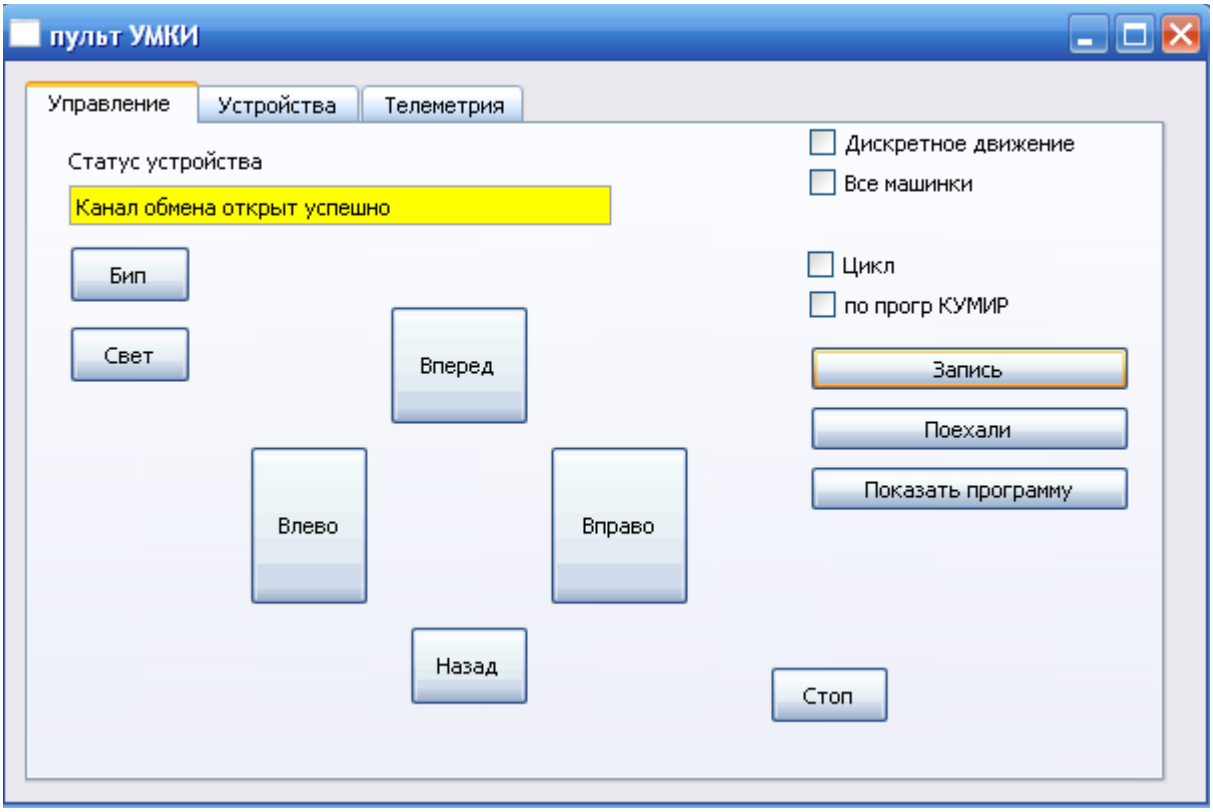
1. езде змейкой с объездом препятствий;

2. парковке в гараж;
3. попробовать загнать шарик в гараж;
4. попробовать перевезти шарик на платформе, и т.п.

Постановка проблемы: Чем же наши роботы отличаются от простой радиоуправляемой игрушки? Как мы уже говорили раньше роботом можно не только непосредственно управлять, но и робот должен запомнить ряд действий и затем повторить эти действия, возможно и не один раз.

Теперь целесообразно перейти на базовый уровень управления роботизированными платформами, который позволяет провести серию экспериментов по программному управлению платформами.

После запуска программы на уровне Base появляется рабочее окно с тремя вкладками: «Управление», «Устройства», «Телеметрия»

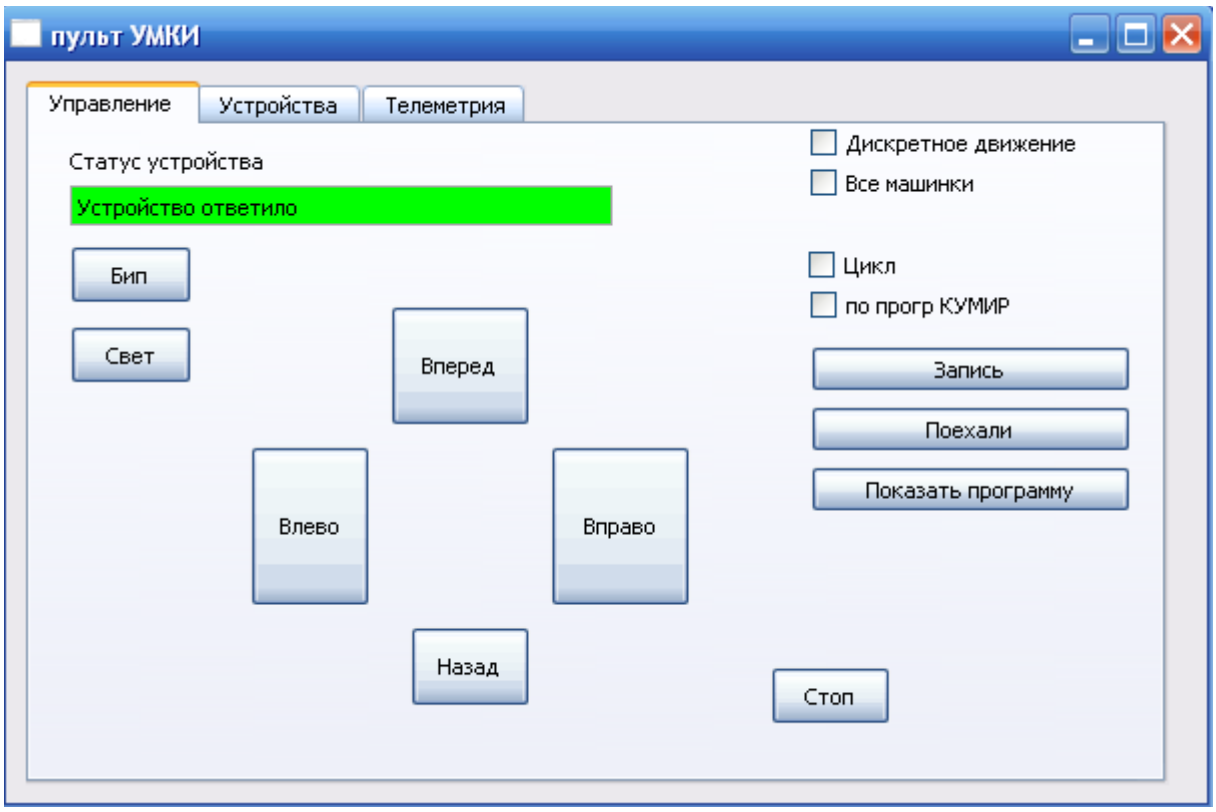


11.2 Закладка «Управление»

Первая вкладка «Управление» позволяет работать в уже известных направлениях: вперед, назад, влево, вправо. Кроме этого, появились новые возможности:

11.3 Статус устройства

При запуске программы в окне «Статус устройства» появляется надпись желтого цвета: «Канал обмена открыт успешно», это означает, что программа запущена. Далее, если произвели включение платформы, окно статуса меняет цвет на зеленый и появляется надпись: «Устройство ответило», что показывает готовность платформы к работе.



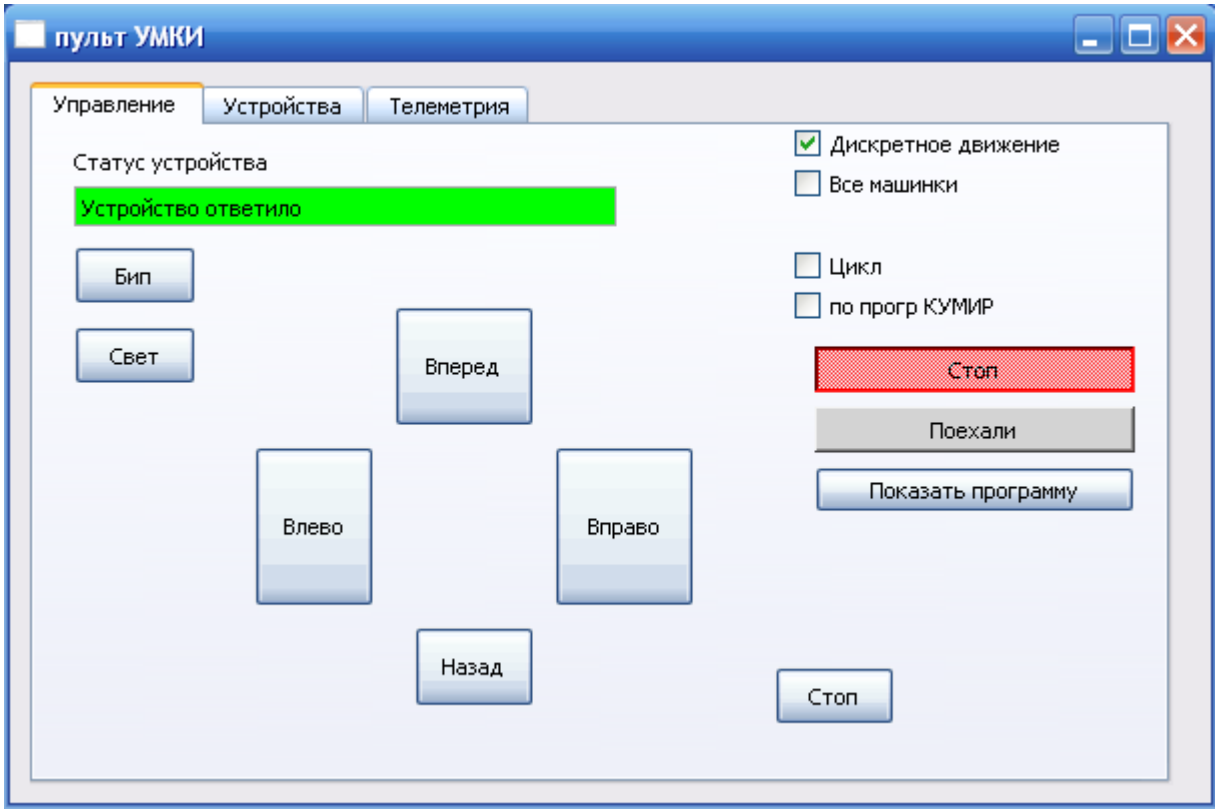
11.4 Кнопка «Запись»

При нажатии на кнопку происходит запись в файл сценария перемещения машинки.

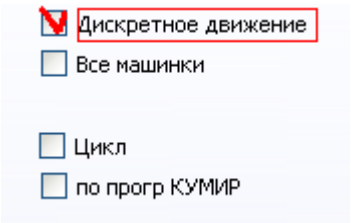
Чтобы выполнить запись нужной траектории движения машинки, надо нажать кнопку «Запись», выполнить несколько команд в дискретном режиме, затем отжать копку «Запись» (кнопка неактивна), нажать кнопку «Поехали». Машинка должна повторить команды, которые были выполнены при ручном управлении. В случае установки режима Цикл (установить чекбокс команды Цикл), выполненный однократно набор команд будет повторяться постоянно. В сценарий записи можно включать также команды звука и света.

11.5 Кнопка «Поехали»

При нажатии кнопки происходит движение по записанному сценарию.



11.6 Дискретное движение



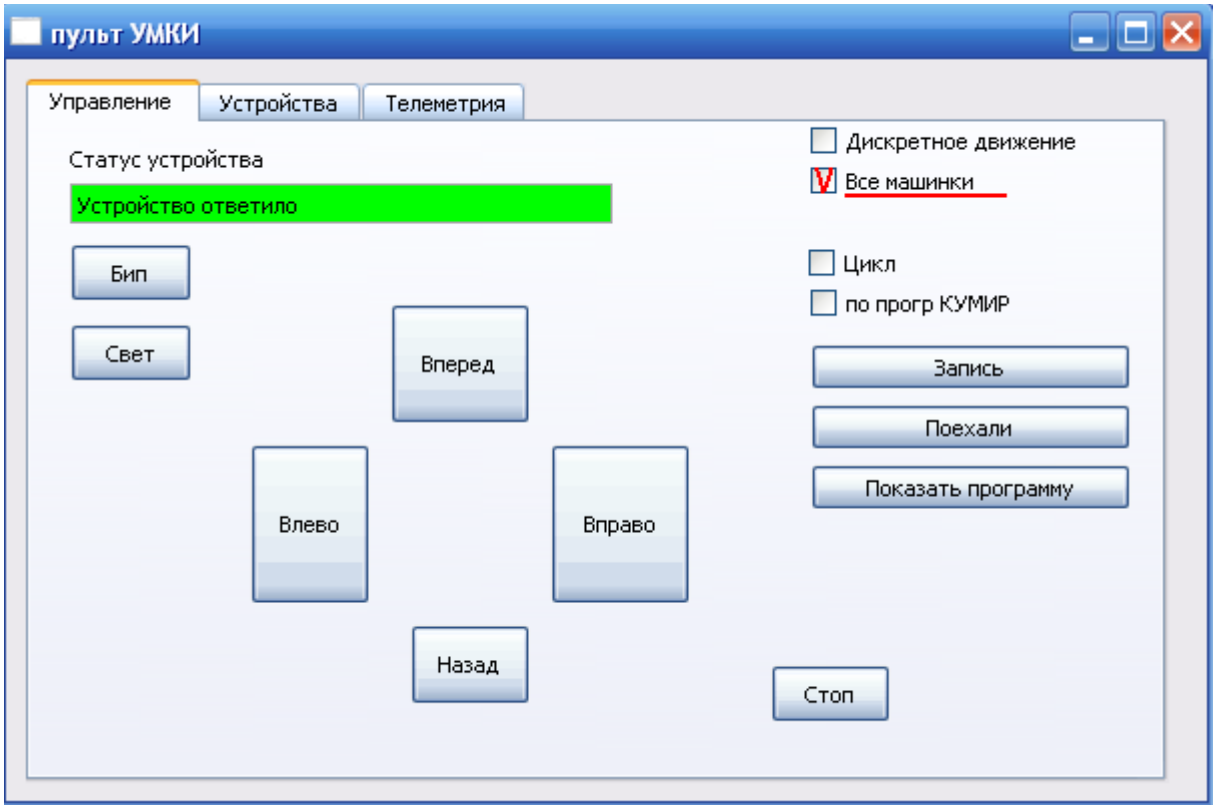
Выбор данной функции позволяет машинке совершать движения дискретно, т.е. порциями.

Если снять галочку, машинка движется, пока нажата клавиша.

Словарь! Дискретный сигнал – это сигнал, параметр которого принимает конечное число значений, меняющееся через определенные промежутки времени (скачками).

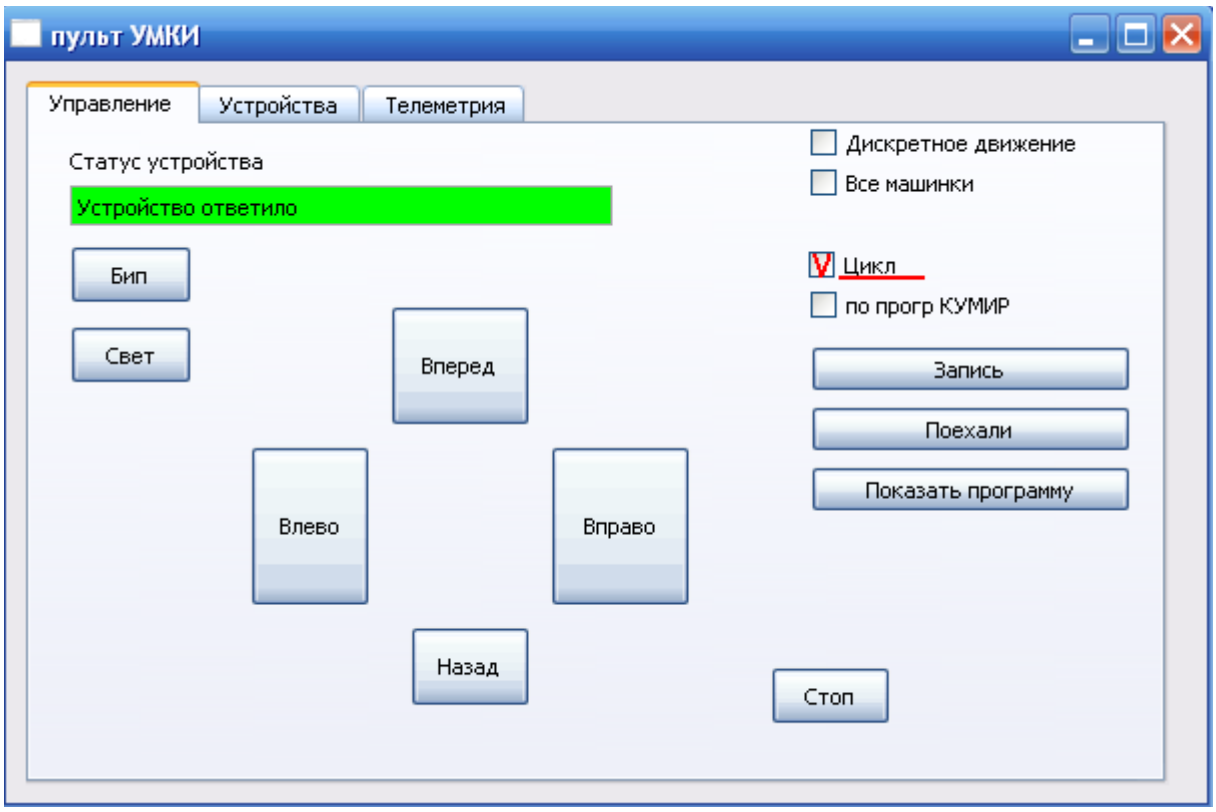
11.7 Все машинки

Выбор функции «Все машинки» позволяет управлять всеми доступными платформами одновременно .



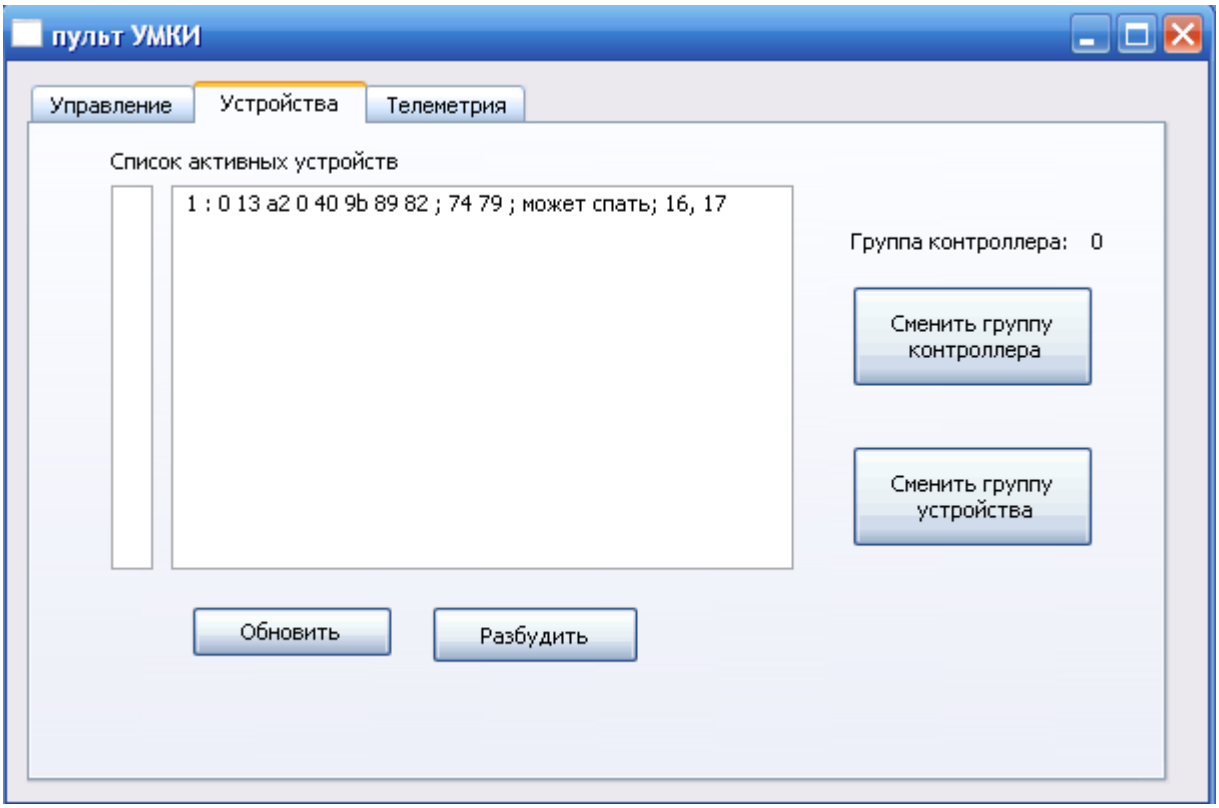
11.8 Цикл

Выбор функции «Цикл» позволяет исполнять программу движения машинки в бесконечном цикле (рис.10).



11.9 Вкладка «Устройства»

В поле «Список активных устройств» перечисляется весь набор доступных машинок. Каждой машинке присвоен свой уникальный МАК- дрес (рис.12). Выбор машинки происходит выделением мышью. Может быть выбрана одна или несколько машинок.



§ 12 Эпизод двенадцать. Управление роботом из среды Snap. 6 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=13

Тема по информатике: Алгоритмика. Управление исполнителями с помощью команд и их последовательностей.

Среда программирования Snap. Сходство и различие со средой программирования Scratch

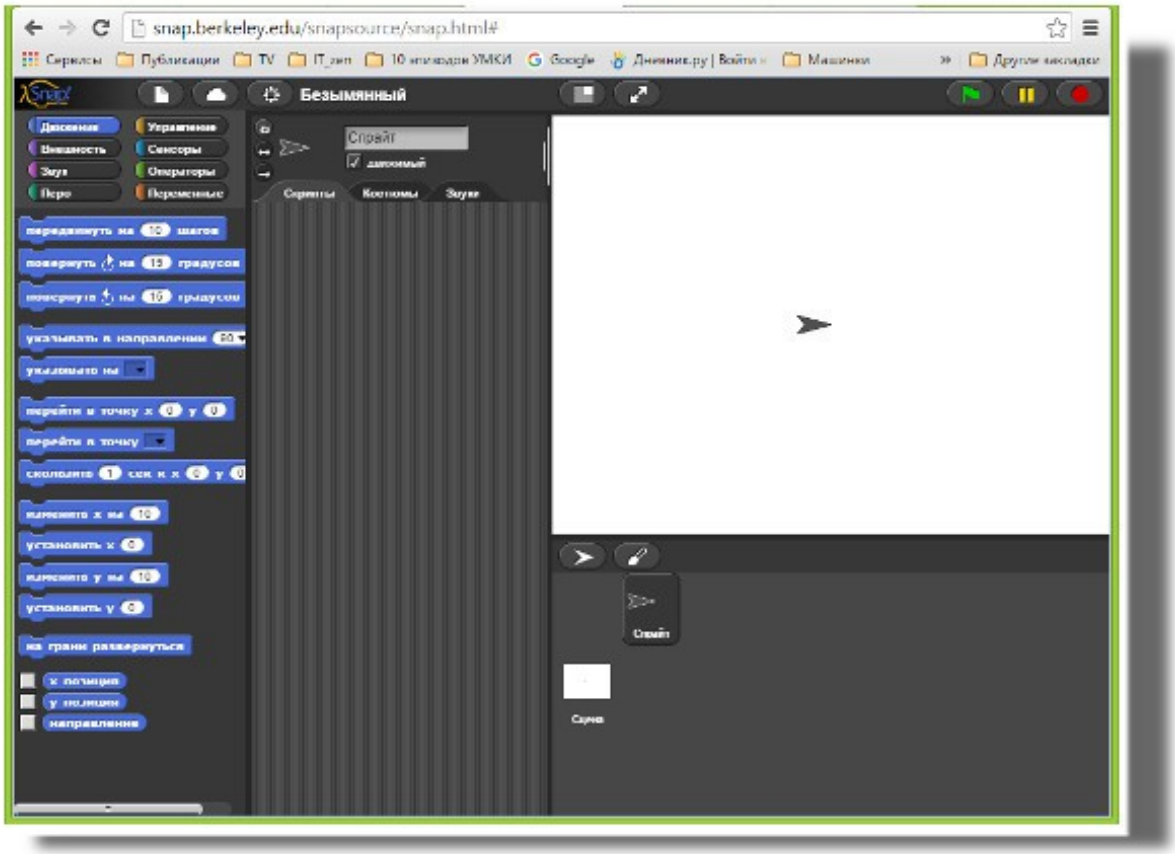
Управление платформой SmartCar из среды Snap. Составление линейных и циклических алгоритмов на языке Snap для платформы SmartCar

Snap http://snap.berkeley.edu/ http://s4a.cat/snap/

Мы уже достаточно научились управлять нашим формальным исполнителем котом, научились управлять роботизированными платформами с помощью пульта управления, но как же научить наши умные машинки двигаться по программе, которую мы сами бы и составляли?

К сожалению программа Scratch версии 2 не позволяет управлять физическими устройствами, но такую возможность дает ряд программ использующих идеологию программирования с которой мы уже достаточно хорошо знакомы. Сюда можно отнести и Scratchduino и S4A, но мы будем работать с программой Snap.

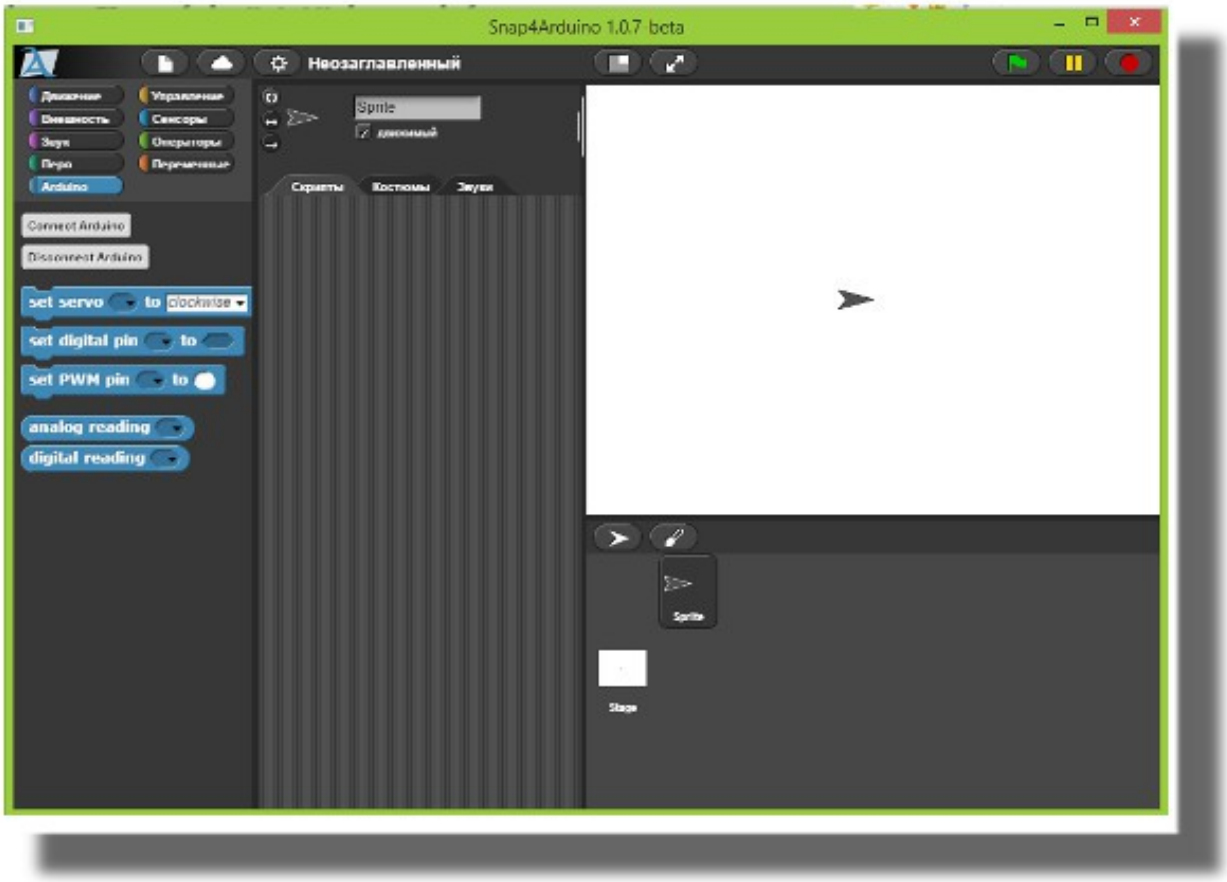
Рассмотрим интерфейс программы:



Как видим, ящики команд похожи: Движение, Управление, Внешность, Переменные и др. Есть поле скриптов, только исполнитель принял теперь вид не Кота, а

маленькой стрелки, но для управления роботом это даже удобнее: куда показывает стрелка, туда и будет двигаться наша платформа. Можно даже попробовать составить простенькую программу на этой платформе. Заставьте, например стрелку двигаться отражаясь от крев экрана

Теперь осталось подключить к компьютеру робота и попробовать самим составить для него программу в среде Snap. Дистрибутив Snap нужно скачать и установить на локальном компьютере. Удобнее, если мы подключим SmartCar, оснащенный контроллером Arduinio. Запустим программу:



Как видим у нас появился еще один ящик: Arduino. С помощью него можно подключиться к роботизированной платформе, управлять сигналами и моторами.

§ 13 Эпизод тринадцать. Подробнее о переменной. 6 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=14

Тема по информатике: Алгоритмика. Составление алгоритмов для управления исполнителем.

Ну наконец, мы смогли составить простые программы для управления движения роботом, но чтобы решать реально возникающие задачи наших знаний недостаточно. Поэтому, прежде, чем займемся дальнейшим управлением робота, подробнее попробуем научиться решать отдельные задачи, на реализацию базовых алгоритмов с среде программирования.

А одним из основных понятий, связанных с программированием является понятие переменной. Нужно отметить, что все дальнейшую работу можно выполнять как в среде Scratch, так и в среде Snap

13.1 Переменные

Давайте вспомним детское стихотворение Германа Сапгира про Принцессу и Людоеда.

Погода была ужасная,
Принцесса была прекрасная.
Днем, во втором часу
Заблудилась принцесса в лесу.

Вдруг видит - полянка прекрасная,
На полянке - землянка ужасная,
А в землянке сидит людоед.
Заходи, говорит, на обед.

Достает он нож - дело ясное.
Вдруг увидел, какая прекрасная.
Людоеду сразу стало худо.
Уходи, говорит, отсюда.
Аппетит, говорит, ужасный,
Слишком вид, говорит, прекрасный.

И пошла потихоньку принцесса,
Прямо к замку вышла из леса.
Вот какая легенда ужасная,
Вот какая принцесса прекрасная.

...А может, все было наоборот?

Погода была прекрасная,
Принцесса была ужасная.
Днем, во втором часу
Заблудилась принцесса в лесу.

Вдруг видит - полянка ужасная,
На полянке - землянка прекрасная,

А в землянке сидит людоед.
Заходи, говорит, на обед.

Достает он нож - дело ясное.
Вдруг увидел, какая ужасная.
Людоеду сразу стало худо.
Уходи, говорит, отсюда.
Аппетит, говорит, прекрасный,
Слишком вид, говорит, ужасный.

И пошла потихоньку принцесса,
Прямо к замку вышла из леса.
Вот какая легенда прекрасная,
Вот какая принцесса ужасная.

...А может, все было наоборот?

И попробуем сконструировать это стихотворение в среде Scratch:

Загрузите файл 06_01.sb2 и запустите на выполнение. Сначала расставьте определения выбирая из первого ящика подходящие слова и размещая их на зеленых квадратах с цифрой 1, разместите подходящие определения на желтых квадратах с цифрой 2 из второго ящика.

Прочитав получившееся стихотворение, нажмите кнопку «Готово».

Некоторое время мы имеем возможность полюбоваться на прекрасную принцессу, после чего, появляется кнопка «А наоборот?». Если нажать на нее – мы увидим, что ситуация кардинально поменялась: все прекрасное стало ужасным, и наоборот: все ужасное стало прекрасным.

Фактически же, мы просто изменили номера ящичков в которых хранятся наборы слов «Прекрасная» и«Ужасная».

То есть, только от номера ящика будет зависеть, прекрасную или ужасную сказку мы с вами будем слушать.

13.2 Как считать очки

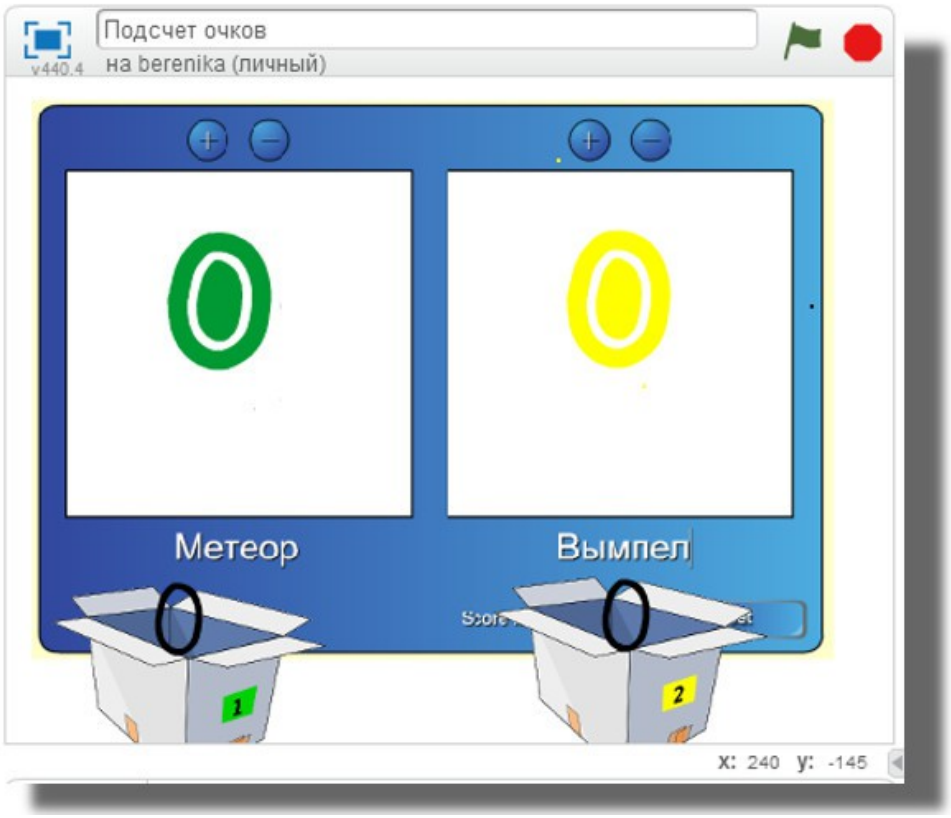
Давайте рассмотрим задачу, когда нам необходимо автоматизировать подсчет очков: например соревнуются две команды: Вымпел и Метеор, каждая из которых стремится забить гол в ворота противника, увеличивая у себя количество очков:

Итак, проанализируем ситуацию...

Сначала у нас на табло счет: 0:0

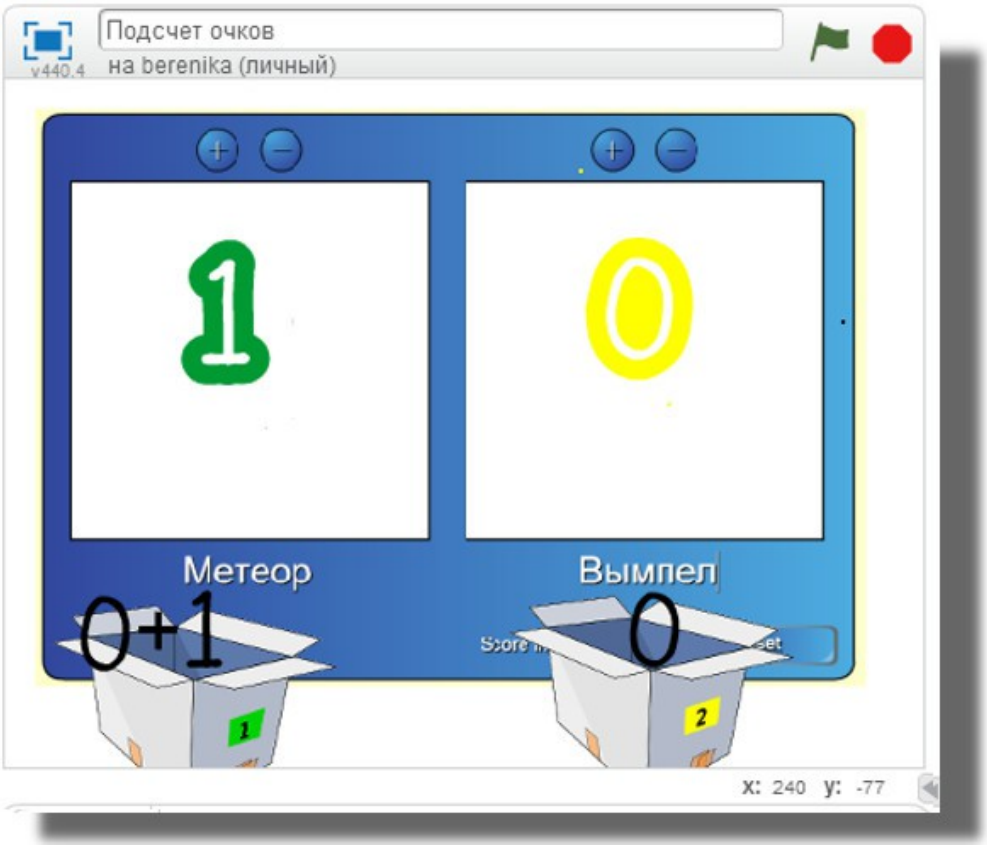
То есть, в ящичках где хранятся очки и команды Вымпел и команды Метеор лежат значения равные нулю.

Таким образом, если мы назовем ящик с номером 1, например М (Метеор), а ящик с номером 2, например В (Вымпел), то в начале игры, мы можем написать, что М=0 и В=0.

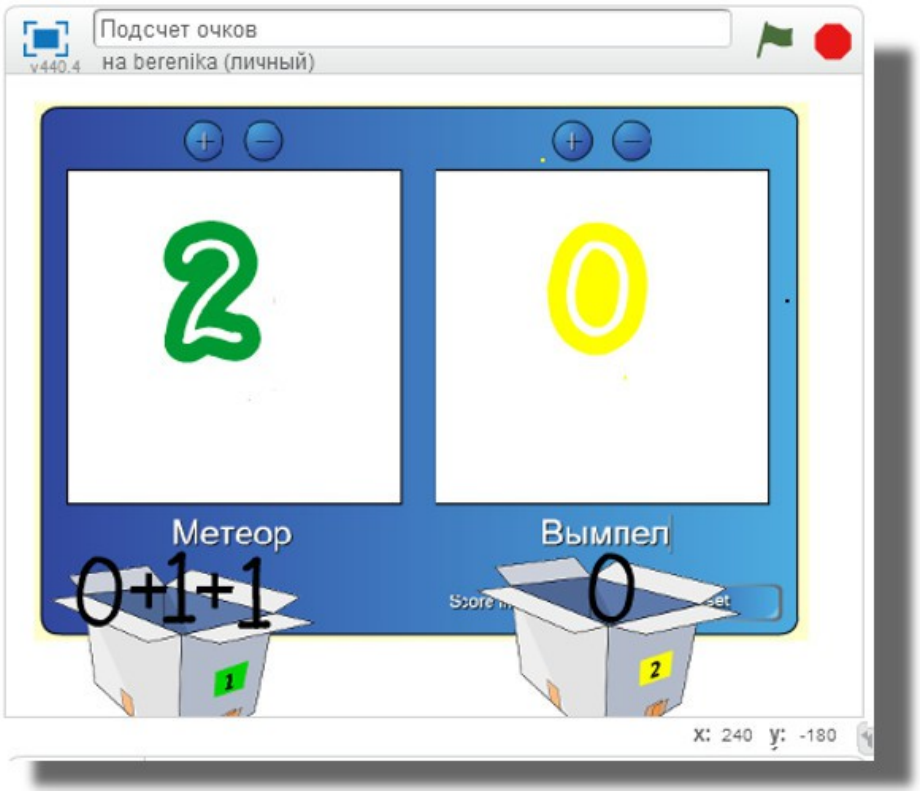


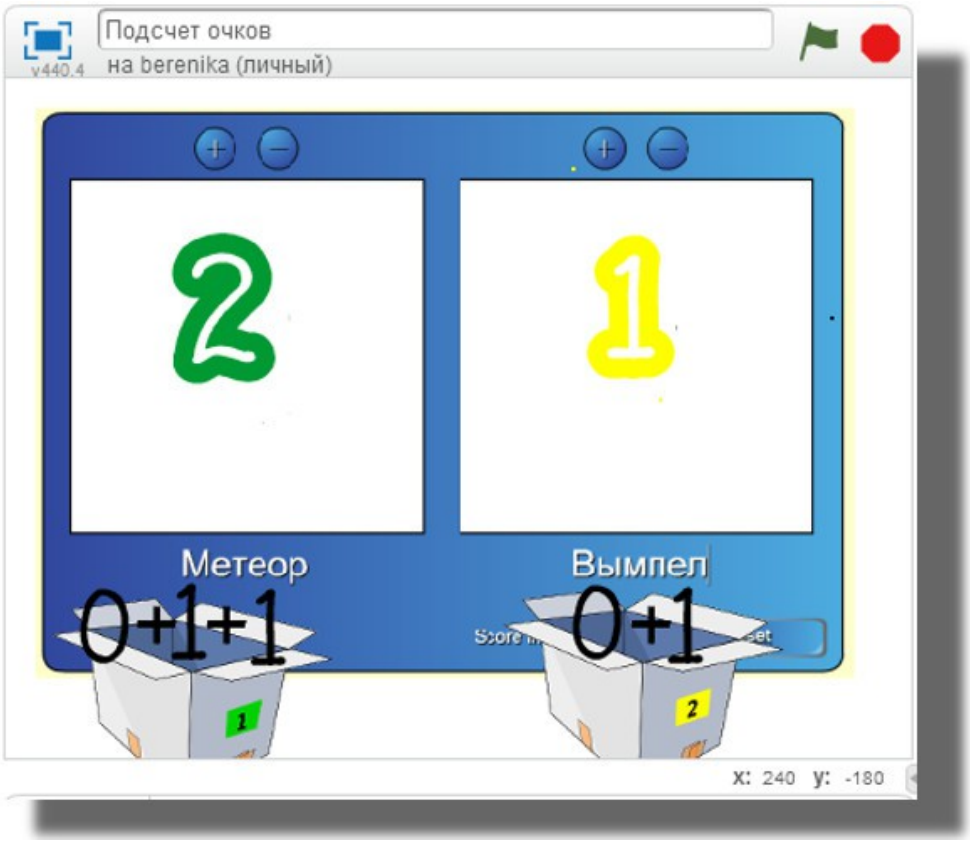
Затем команда Метеор получает гол в свои ворота, и числовое значение которое лежит в ящичке М увеличивается на единицу. Тогда для команды Метеор мы можем написать очень интересную штуку: M=M+1.

С точки зрения математики, такое выражение выглядит бессмысленным, а вот с точки зрения подсчета очков, и, соответственно, с точки зрения программирования, эта конструкция уже имеет определенный смысл – мы увеличиваем значение которое лежит в ящичке М на единицу.



Вот Метеор получил еще один гол, и опять наша конструкция M=M+1 имеет глубокий смысл: у метеора уже было пропущено одно очко, и значение в ящичке M вновь увеличено на единицу, то есть, теперь там лежит уже двойка.





Аналогично, и для команды Вымпел можно построить подобную же конструкцию: В=В+1.

Поскольку это первый гол в ворота Вымпела, значение в ящике В увеличилось пока на одну единицу, и счет стал 2:1. И так далее можно, продолжать на протяжении всего матча – при полученном голе, увеличиваем значение ящичка М или ящичка В. Запомним пока, вид конструкции М=М+1, он кстати, так и называется – **счетчик**, в дальнейшем мы еще не раз будем его использовать.

А пока обратимся к ящичкам: как видим, числа которые лежат в ящичках могут постоянно меняться (а могут и не меняться – счет в игре зависит от мастерства команд :). В программировании, подобные ящички получили название – **Переменные**. Как вы поняли, чтобы отличать один ящичек от других каждый должен иметь собственное имя, и мы должны договориться что же мы будем закладывать в эти ящики: числа, символы или вообще какие-то экзотические значения.

Строго говоря, определение переменной выглядит таким образом:

Переменная — именованная область памяти, которую можно использовать для осуществления доступа к данным и изменять значение в ходе выполнения программы.

Понятно, что прежде чем записывать счет в матче, нужно определиться где у нас необходимые ящички, и только потом начинать игру, так и прежде чем составлять программу, переменную необходимо объявить: дать ей имя, определить ее тип (договориться, что конкретно, мы будем закладывать в ящички).

13.3. Квадратные узоры

А теперь, потренируемся...

Задача: сконструировать алгоритмы заданий для исполнителя умеющего рисовать следующие изображения:

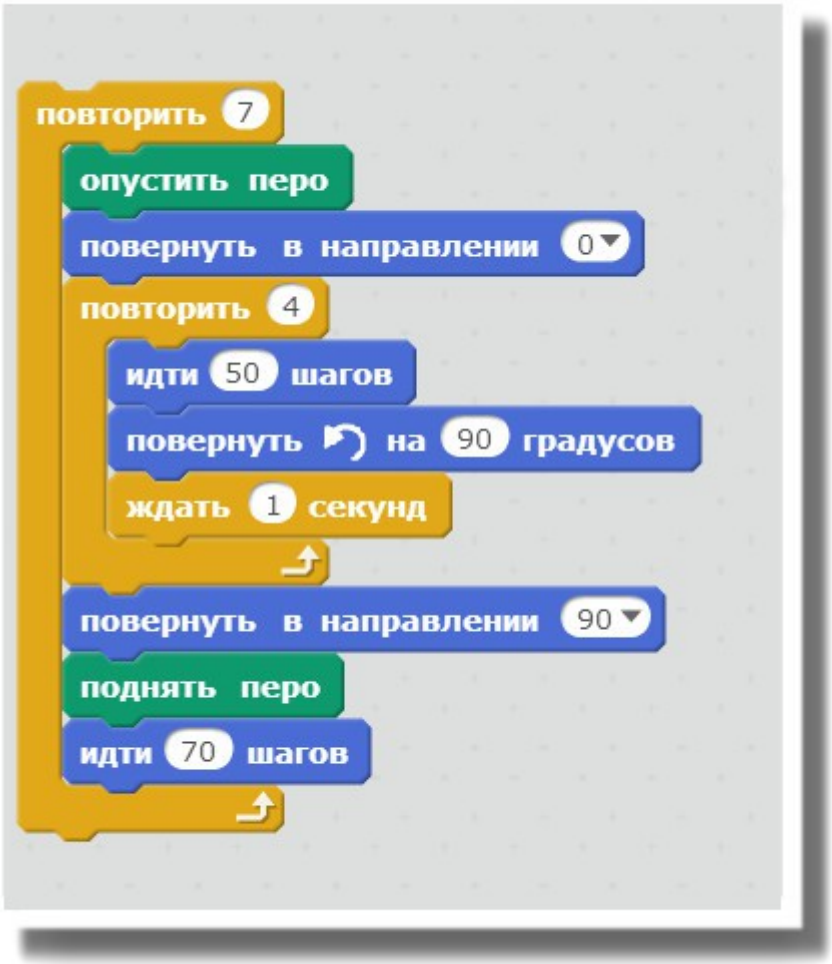
1. Создать программу, рисующую на экране квадрат.
2. Создать программу рисующую на экране ряд квадратов.
3. Создать программу рисующую на экране ряд уменьшающихся квадратов.

1. Первая программа, самая простая, никаких особых приемов не требует: нужно придти в определенное место, опустить перо, пройти некоторое количество шагов, повернуться на 90º, и повторить эти действия четыре раза

Комментарий к программе:

После заданий установок для исполнителя (размещение в точку с ориентировочными координатами (-180, 0), задания направления движения, установки размера пера, очистки экрана и команды опустить перо) задан цикл с параметром равным 4 (повторить четыре раза) движение на 50 шагов и поворот на 90 градусов.

2. А вот следующая программа, рисования на экране ряда квадратов, будет чуть длиннее, но в принципе тоже ничего сложного, просто после окончания рисования одного квадратика нужно поднять перо и чуть сместиться вправо, и снова повторить созданную конструкцию необходимое количество раз:



Комментарий к программе:

Фрагмент первой программы помещен еще в один цикл, который повторяется семь раз. Таким образом, мы получили конструкцию, называемую вложенные циклы, когда один циклический алгоритм размещается внутри другого циклического алгоритма.

Теперь попробуем изменить программу таким образом, чтобы квадратики раз за разом, уменьшались, типа матрешки...

Опять используем предыдущую программу.

Анализируем, что бы нам хотелось сделать для реализации задания: нарисовать один квадрат, поднять перо сдвинуться в сторону и вновь рисовать квадрат, сторона у которого немного уменьшена, и так повторить несколько раз. Конечно, все это возможно проделать вручную, повторив цикл рисования квадрата столько раз, сколько уменьшающихся квадратиков нам требуется.

Однако, в программировании существует значительно более изящное решение данной проблемы, если использовать **переменные**.

Вспомним, что Переменная в программировании, это место в памяти компьютера, куда на время можно поместить некоторое значение например некоторое число или какой-то набор символов.

Задается переменная с помощью Оранжевого ящика **Данные**, командой *Создать переменную*. При создании переменной необходимо задать ей имя. Помните, мы задавали для ящичков команд названия М и В, таким образом, мысленно, можно представить переменную в виде ящичка, в котором будем хранить различные числа, значения которых могут меняться.

Итак, командой *Создать переменную*, создадим переменную, задав ей имя, например, А. В начале программы зададим значение переменной А равное 50.

При рисовании квадрата же сторону квадрата предложим исполнителю рисовать длиной А. Поскольку А у нас равно пятидесяти, при рисовании первого квадрата ничего не изменилось.

А вот после завершения процедуры квадрат, мы добавим из оранжевого ящика **Данные** команду Изменить значение А на (-5), (для нашего случая).

То есть, при выполнении процедуры "квадрат" второй раз, значение переменной А будет уже не 50, а на 5 меньше (50-5=45).

Таким образом, второй квадрат будет со стороной 45, у третьего квадрата сторона будет еще меньше на 5 и равна 40.

То есть мы реализуем конструкцию **счетчик (А=А-1)**, которую рассматривали, когда считали очки, только сейчас мы уже не прибавляем, а вычитаем параметр счетчика, и значение переменной уменьшается.

При необходимости, изменяя всего только один параметр в команде Повторить(), мы можем добиться рисования достаточно большого количества квадратов разных размеров.

Вот какая удобная штука – переменные!

13.4. Попробуем сами?

Задания для самостоятельного выполнения.

Создать программу, рисующую пунктирную линию

Создать программу, рисующую на экране квадрат.

Создать программу рисующую на экране пирамиду из уменьшающихся квадратов.

Создать программу, рисующую на рабочем поле изображение снеговика.

Создать программу, рисующую на экране елочку, из уменьшающихся треугольников.

Создать программу, рисующую на рабочем поле изображение снежинки.

Меандр – один из геометрических орнаментов. Он состоит из разнообразных изломов прямой линии под прямыми же углами и из сочетаний таких изломов. Название Меандр происходит от названия реки, образующей в своем течении множество извилин. Иногда меандром называют также орнамент, состоящий из закругленных изгибов одной линии, имеющих форму концентрических завитков или составляющих подражание морским волнам.

Попробуйте разработать программу, рисующую полоску орнамента – меандра.

13.5 Поговорим с роботом? Организация диалога и еще несколько слов о переменной

Итак, подведем итог тому, что мы узнали о переменных.

В математике переменные – это данные, которые меняют свои значения. При решении задачи на компьютере это так же верно, но в данном случае такое свойство переменных это уже следствие. В программировании, переменная – это небольшая область в оперативной памяти компьютера, куда во время работы программы можно занести и хранить в закодированном виде некоторое значение (число, либо набор символов), которым в дальнейшем можно пользоваться и которое можно изменять.

Место такой структурной единицы памяти в общем объеме оперативной памяти определяется адресом – номером ячейки памяти в двоичной системе счисления. Однако, пользоваться двоичными номерами очень неудобно, поэтому при написании программ на алгоритмическом языке программирования стали пользоваться более удобным заданием способом – определением имени переменной. То есть, имя переменной – это название места (ячейки) в оперативной памяти, используемое в программе вместо адреса записанного в двоичной системе счисления.

Создается переменная в среде Scratch командой Make,из оранжевого ящика **Данные**.

При выборе команды создания переменной открывается диалоговое окно, предлагающее присвоить имя переменно, и становятся доступны операторы предлагающие установить и изменить значение созданной переменной, показать и спрятать переменную. По аналогии с «взрослыми» языками программирования имя переменной лучше задавать в латинской кодировке.

Сразу после создания, никакого конкретного значения переменной в ячейке памяти нет, пока оно не будет туда занесено, (правда, по умолчанию в среде Скретч только что созданная переменная имеет значение равное нулю). Поэтому в любой программе использовать переменную можно только после того, как определено ее значение. В каждом языке программирования существует набор операторов, позволяющих давать переменным значения, т.е. заносить их в ячейки, названные именем переменной. Значение переменной можно присвоить, т.е. занести его в ячейку памяти специально существующей для этого командой присваивания. Для языка Scratch , это команда *set(имя переменной) to ()*. (Установить значение переменной S равным (по умолчанию) нулю).

Итак, подведем итог:

Программа – упорядоченная последовательность действий для компьютера, реализующая алгоритм решения какой-либо задачи.

Язык программирования Scratch включает около сотни команд. Каждая команда является инструкцией, предписывающей компьютеру сделать что-то.

Переменная (variable) – некоторая величина, которая может изменяться, принимая в процессе этого изменения различные значения

Оператор — предписание в языке программирования, предназначенное для задания некоторого действия. Каждый оператор представляет собой законченную фразу языка и определяет некоторый этап обработки данных.

Оператор присваивания.

Основной оператор в любом языке программирования. Оператор присваивания set (имя переменной) to (значение), присваивает переменной новое значение, задаваемое каким-либо выражением.

set S to (3) (задать S значение 3) – эта запись означает, что переменной S присваивается значение 3.

А запись set S to (S+1), означает, что к прежнему значению переменной S прибавляется единица и результат присваивается этой же переменной, то есть если считать, что S было равно 3, то после выполнения второго оператора присваивания значение переменной уже будет 4. Прежнее значение не сохраняется. Аналогом этой команды будет команда change S by (1) (Изменить значение S на 1)

Для удаления созданных переменных нужно использовать команду Delete a variable.

(Поскольку большинство языков программирования построены на основе английского языка, при работе в этом эпизоде, потренируемся в англоязычной среде программирования, да и в дальнейшем будем стараться работать с командами на английском языке)

13.6. Операторы ввода-вывода

Для организации диалога в среде Scratch используются команды из голубого ящика Sensing (Сенсоры): ask () and wait – answer и из фиолетового ящика **Looks** (**Внешность**) – say() и think().

Оператор ask () and wait (задай вопрос и жди) необходим для ввода данных, присваивая введенное значение временной переменной answer во время ввода.

Если поставить следующий оператор ask () and wait то введенный ответ вновь будет записан в переменную answer, и предыдущее значение будет потеряно. Чтобы подобного не происходило, нужно заранее создать необходимые переменные и после получения значения временной переменной answer складывать полученные значения в эти переменные.

Операторы из фиолетового ящика **Looks** (**Внешность**) – say() и think(), предлагают компьютеру вывести информацию на экран дисплея. могут быть представлены в форме: вывести на экран и ждать несколько секунд:

Причем, увидеть какая именной информация хранится в созданных переменных, можно поставив галочку у имени переменной: имя переменной и ее значение будут отображаться на рабочем поле программы.

13.7 Программа общения с исполнителем

Для начала попробуйте составить простейшую программу, предлагающую исполнителю запросить Ваше имя, запомнить его в ячейке Name, затем запросить Ваш возраст, сохранить его в ячейке Vozrast, а затем вычислить год Вашего рождения и сообщить результат, обращаясь к Вам по имени.

Инструкции к выполнению:

Создайте переменные Name и Vozrast и Year (в ней будет храниться вычисленный год рождения), и сразу после запуска программы присвойте переменным значение 0.

После ввода вопроса о вашем имени присвойте переменной Name то значение, которое оказалось во временной переменной answer, аналогично задайте вопрос про возраст и сохраните из переменной answer полученное значение в переменную Vozrast. Затем вычислите в каком году мог родится отвечающий на вопросы исполнителя, для этого из года который идет сейчас вычтите значение переменной Vozrast, и поместите полученное значение в переменную Year.

В конце выведите сообщение с помощью команды say(). Для вывода дополнительных слов удобна конструкция из ярко-зеленого ящика join()(), позволяющая слить воедино символьные строки и переменные.

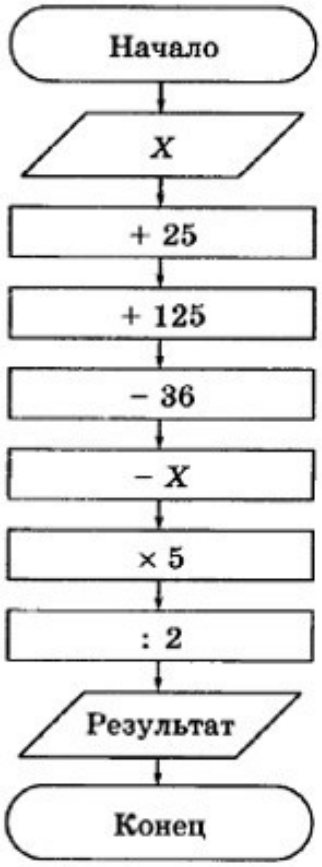
Задание: По подобному образцу напишите программы, вычисляющие скорость, при запрошенном пути и времени прохождения, и плотность вещества при введенных массе и объеме.

§ 14 Эпизод четырнадцать. Программирование базовых алгоритмов. 6 класс

Электронная поддержка: http://umki-dist.ru/course/view.php?id=22&section=15

Тема по информатике: Алгоритмика. Составление алгоритмов (линейных, с ветвлениями и циклами) для управления исполнителем.

183. Выполните устный счёт по блок-схеме для чисел $X = 64; 125; 840$.



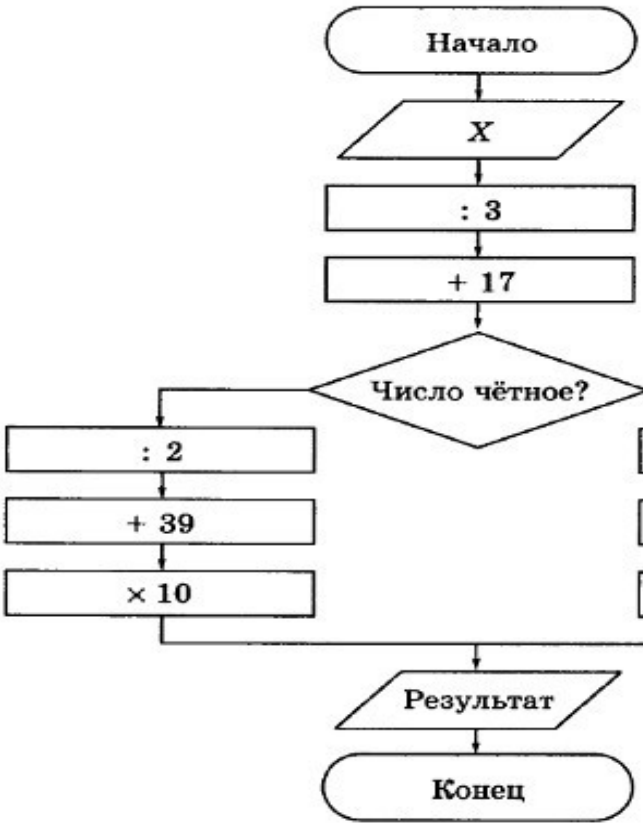
64	125	840

Алгоритм:

1. Ящик События: Запуск программы на выполнение(Когда щелкнут по зеленому флажку)
2. Голубой ящик Сенсоры: Запрос исходных данных (Спросить и ждать)
3. Голубой ящик Сенсоры: Ответ – активитовать отображение на экране
4. Оранжевый ящик Данные: Создаем переменную X, в которой будет храниться результат вычисления
5. Переменной X присваиваем значение Ответ
6. Зеленый ящик Операторы: Задаем команду вычисления

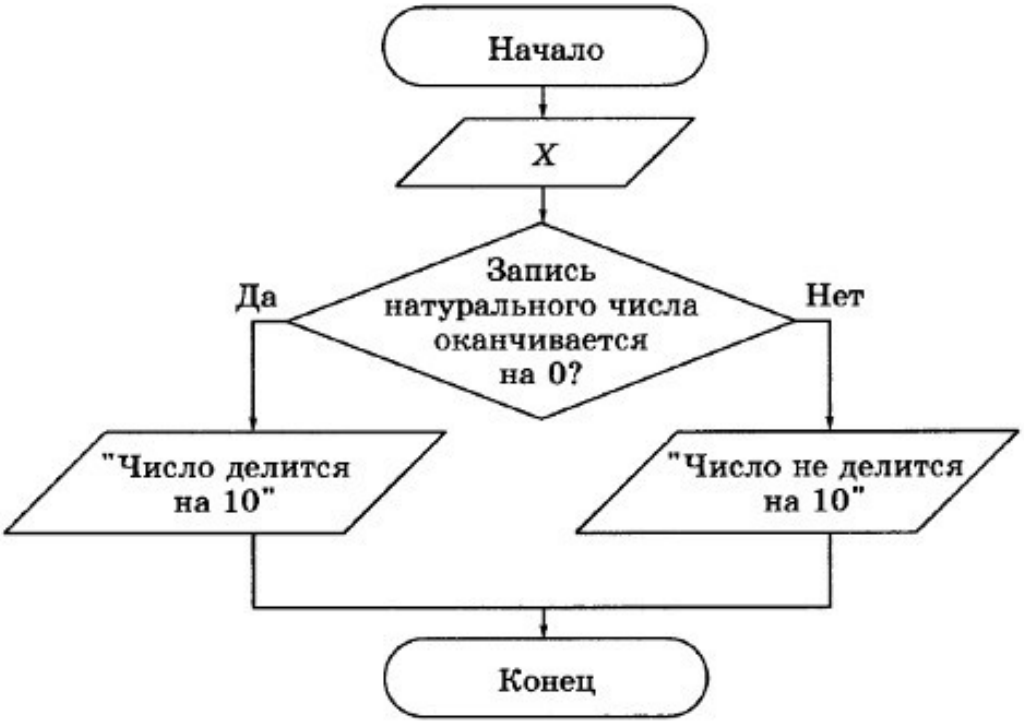
7. Устанавливаем значение переменной
8. Ставим паузы
9. Конструируем алгоритм, согласно блок-схеме
10. Фиолетовый ящик Внешность: Сообщаем результат вычислений и ждем.

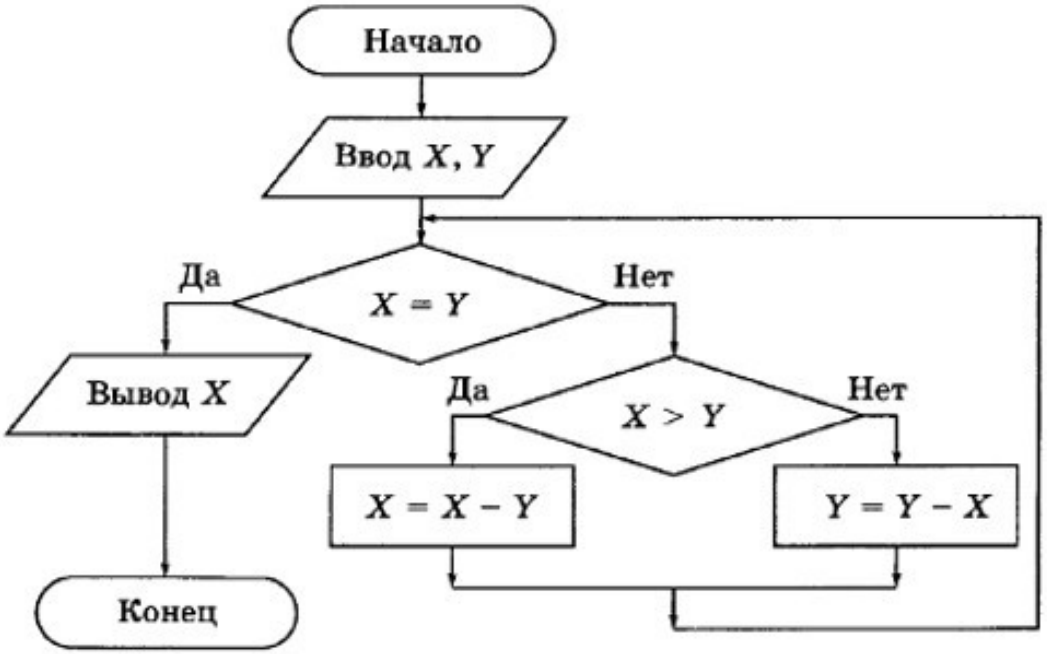
190. Выполните вычисления по блок-схеме для чисел $X = 33; 42; 75$ (запишите ответы в таблицу).



33	42	75

191. Внимательно рассмотрите блок-схему. Сформулируйте в словесной форме признак, о котором в ней идёт речь.





Исходные данные		Промежуточные данные		Результат
X	Y	$X = Y$	$X > Y$	X
5	25			
9	14			

12	90			