Интерактивное взаимодействие объектов. Программа Сластена. Программирование компьютерных игр и интерактивных мультфильмов[1].

Ключевые слова: dublicat, key () arrow pressed, pick random () to (), touching color.

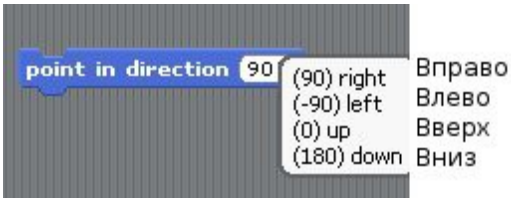
Ну, наконец, мы достаточно знаем для того, чтобы приступить к созданию собственной компьютерной игры.

Мы знаем, как анимировать придуманного героя, как подготовить новые костюмы для его анимации, как герой может перемещаться по экрану, как может меняться вид заднего плана, в зависимости от сценария игры или мультфильма, как объекты могут реагировать на встречу.

Проанализируем, как может выглядеть программа, управления героем с помощью клавиатуры.

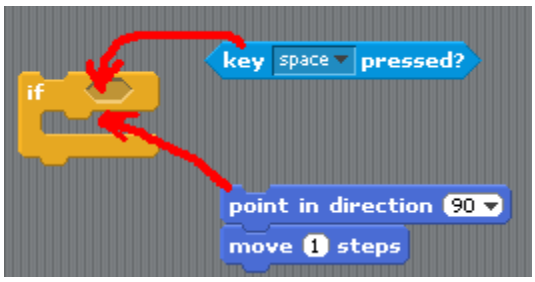
Для примера, возьмем нашего монстрика, которого мы уже учили двигаться

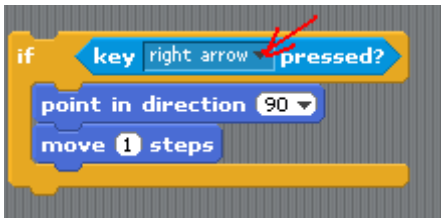
Анализируем задачу: наш сластена должен идти вправо – **если** нажата клавиша на клавиатуре с нарисованной стрелкой вправо, влево – **если** нажата клавиша со стрелкой влево, аналогичная реакция на нажатые клавиши со стрелками вверх и вниз. Вспомним кодирование направления движения:



Команда «`if`» **(если)** начинает свою работу с проверки условия – **если** нажата клавиша (→), **тогда** выполнить действия внутри блока. Из голубого ящика сенсоров блок нажатия клавиши `key right arrow pressed` (**стрелка влево нажата**) размещаем в условии команды `if`. При нажатии клавиши со стрелкой наш герой примет направление `point in direction(90)` и сделает один шаг.

[1] Общая идея взята из материалов дистанционной обучающей олимпиады по информатике DOOI2008 http://www.botik.ru/ICCC/NewPage/ICCCpageRus/Projects/DOOI08/

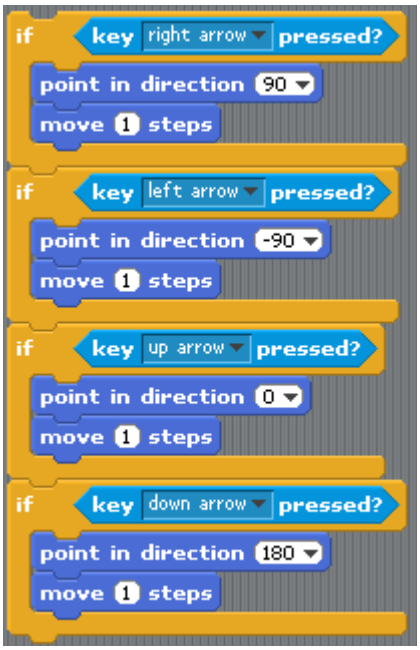




Для оформления движения в другую сторону, скопируем блок. Для этого кликнем по желтому блоку правой клавишей мышки, и в выпадающем меню выберем пункт `dublicat` (**дублировать**).

Новый блок расположим ниже и исправим направление, изменив реакцию на нажатие стрелки «влево»:

Аналогично поступим для создания блоков «вверх» и «вниз»:

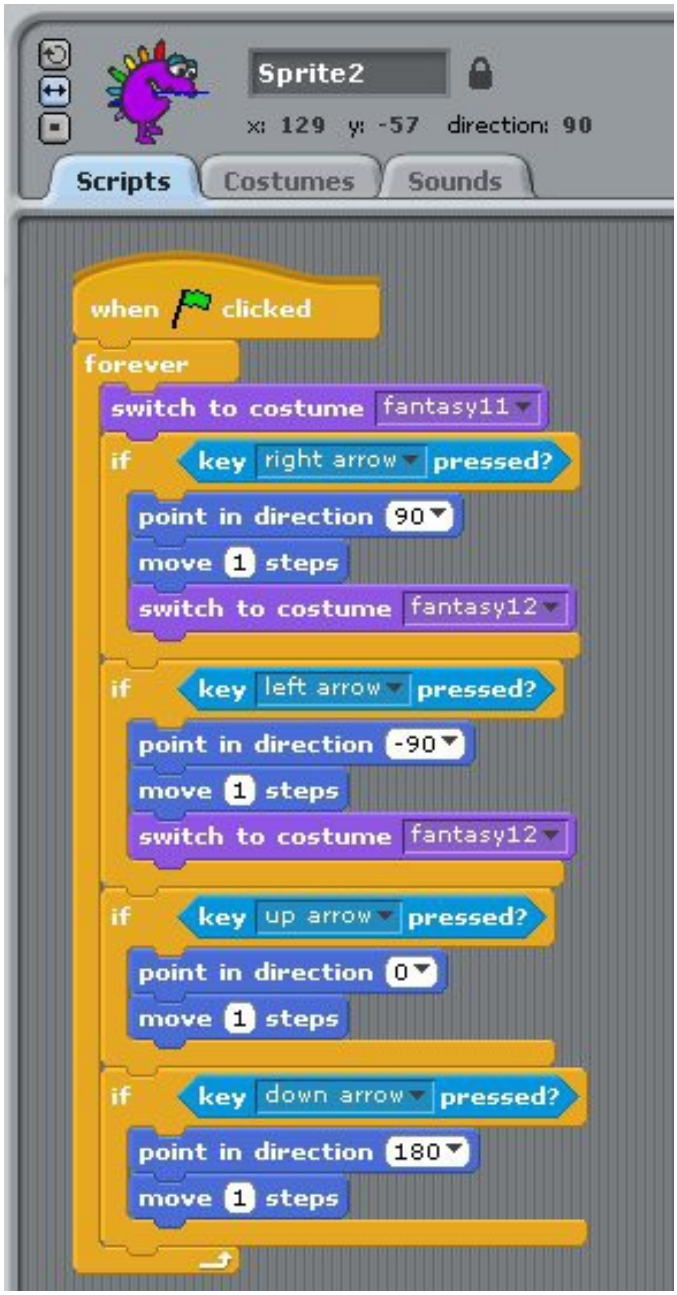


Теперь, если мы подключим команду запуска программы по зеленому флажку, и протестируем программу, то увидим, что при нажатии на любую из клавиш выбора направления, наш исполнитель слегка, совсем чуть-чуть сдвигается в строну выбранного направления. Чтобы движение было хорошо заметно и регулируемо, эти четыре проверки условия должны выполняться много-много раз. Поэтому всю подготовленную программу заключим в бесконечный цикл `forever`.

Таким образом, мы научили нашего сластену реагировать на нажатие управляющих клавиш на клавиатуре.

Для создания анимации используем хорошо знакомую нам смену «костюма» `switch to costume()`. Сначала наш гурман одет в костюм fantasy11, и пока нажата клавиша движения вправо или влево одевается в костюм fantasy12, при отпускании клавиши, костюм fantasy11 восстанавливается, т.е. при нажимании и отпускании стрелок наблюдаем эффект анимации.

Не забываем переключить эффект вращения объекта на эффект отражения.

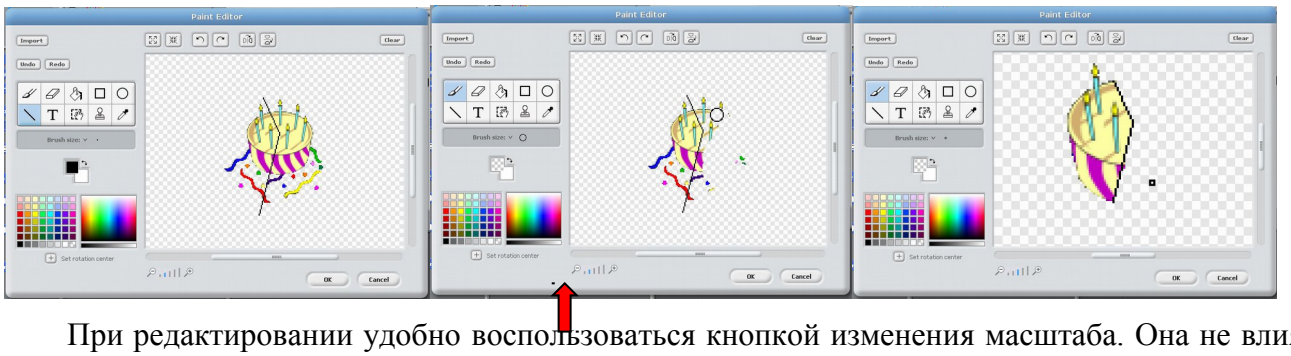


Аналогичным образом создаем и программируем управление еще одним героем, причем создать костюмы для героев можно даже самостоятельно, например, нарисовать вкусный тортик на ножках, который будет убегать, чтобы его не съели. Естественно, вместо управляющих стрелок для второго объекта нужно использовать управление с помощью буквенных клавиш, подобно тому, как мы изменяли способ запуска программы.

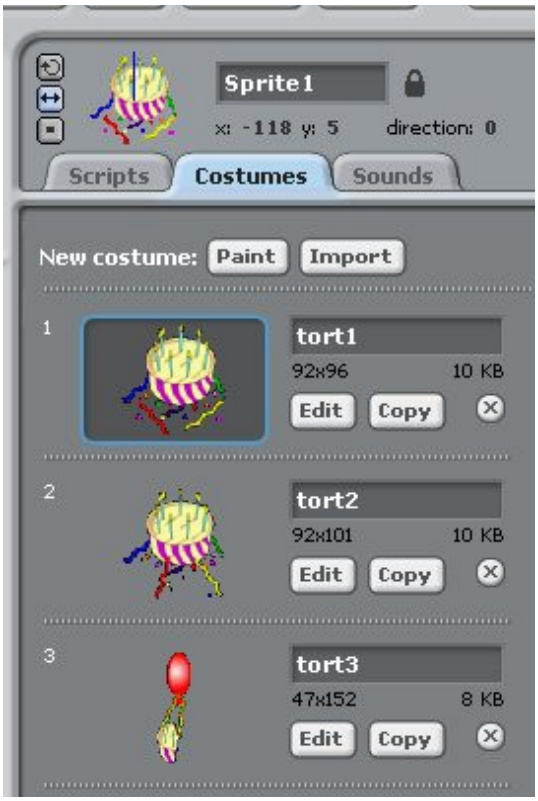
Добавление нового объекта

Осталось научить нашего сластену кушать: вставим новый объект тортик.

Скопируем и отредактируем костюм объекта в нескольких вариантах, один для появления, другой – кусочки, остающиеся после откусывания:

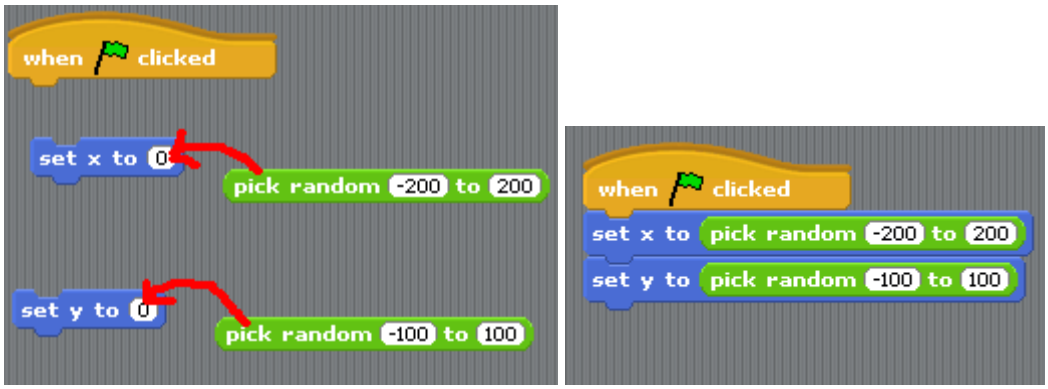


При редактировании удобно воспользоваться кнопкой изменения масштаба. Она не влияет на конечный размер картинки, а только увеличивает или уменьшает изображение при редактировании.



Использование генератора случайных чисел

Запрограммируем вкусный объект таким образом, чтобы он появился в каком-то неопределенном месте, пусть компьютер сам придумает, где поставить этот объект. Зададим компьютеру только область, в которой компьютер будет придумывать числа – эти придуманные числа и будут координатами размещения нашего тортика. Пусть по горизонтали разброс в придумывании будет от -200 до +200, а по вертикали от -100 до +100. В синем ящике выберем команду установки координат `set x to()` и `set y to()`, а вот координаты пусть компьютер определяет сам случайным образом с помощью команды из светло-зеленого ящика: `pick random () to ()` выбирающей случайное число в указанном диапазоне.

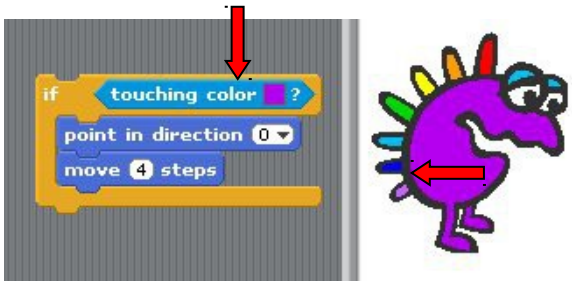


Теперь при каждом новом запуске программы, тортик всякий раз будет стоять в новом месте.

Программирование взаимодействия объектов

Сейчас, наша задача – заставить тортик подниматься вверх, когда его поймает сластена.

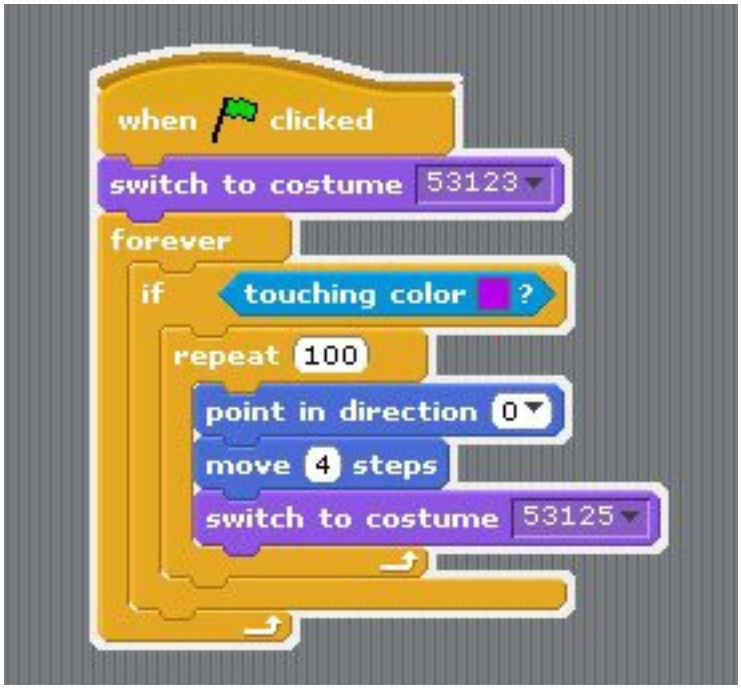
Если сластена касается торта, **тогда** пусть выполнится программа взлета вверх кусочка на шарике. Прикоснулся гурман к торту или нет, проверим следующей командой `touching color`. В данном случае проверяем прикосновение не к спрайту, как в программе с печальной рыбой-солнцем, а соприкосновение с определенным цветом. Выбор цвета выполняется так: сначала кликнуть на квадратик области цвета в блоке, затем кликнуть по персонажу, в месте наиболее характерного цвета:



Когда сластена догоняет тортик, он откусывает кусочек, остаток на воздушном шарике улетает. Затем, вновь можно догонять вкусный объект, пока остаются кусочки. В конце концов, тортик заканчивается и игра прекращается.

Напишем программу взлета тортика. Взлет происходит при выполнении условия – **если** тортика коснулся объект фиолетового цвета (т.е. условие выполняется), **тогда** выполнить перемещение вверх – небольшому кусочку торта на воздушном шарике принять направление `point in direction(0)` и повторять шаги в данном направлении.

Пусть у нас даже сто раз повторятся две такие команды: выбрать направление вверх и сделать 4 шага. Но, проверять выполнение данного условия необходимо много-много раз, поэтому условие проверки соприкосновения с фиолетовым цветом заключим в бесконечный цикл:

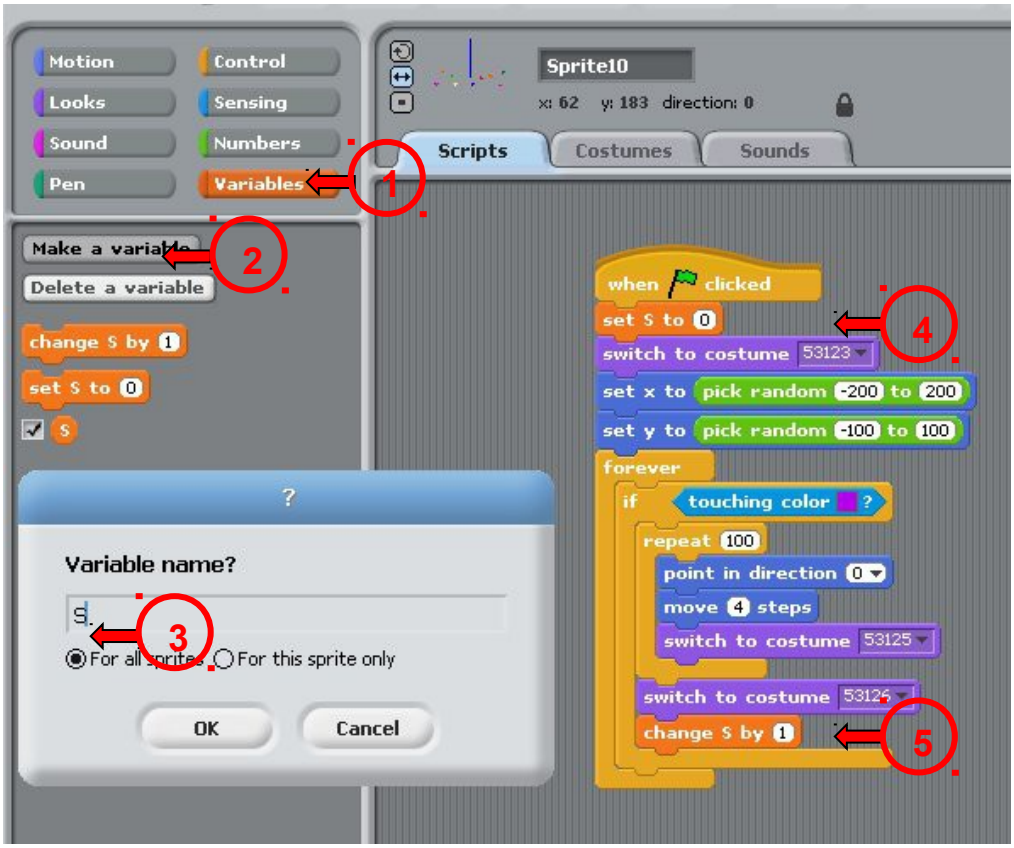


Программирование подсчета очков:

Обычно в играх используют подсчет количества очков, поэтому нам необходима переменная для хранения очков, значение которой будет увеличиваться всякий раз после того, как тортик поднялся вверх.

Введем переменную S, в которой будут накапливаться набранные очки, а начальное значение переменной установим ноль. Если очки на рабочем поле не отображаются, то следует отметить галочку напротив имени переменной.

1. Перейдем в красный ящик **Variables**
2. В ящике **Variables** выберем команду `Make a variable`,
3. В открывшемся окне зададим имя переменной S,
4. Добавим к программе команду установления начального значения переменной `set S to (0);`
5. Добавим изменения значения переменной после перемещения торта вверх: `change S by (1).`

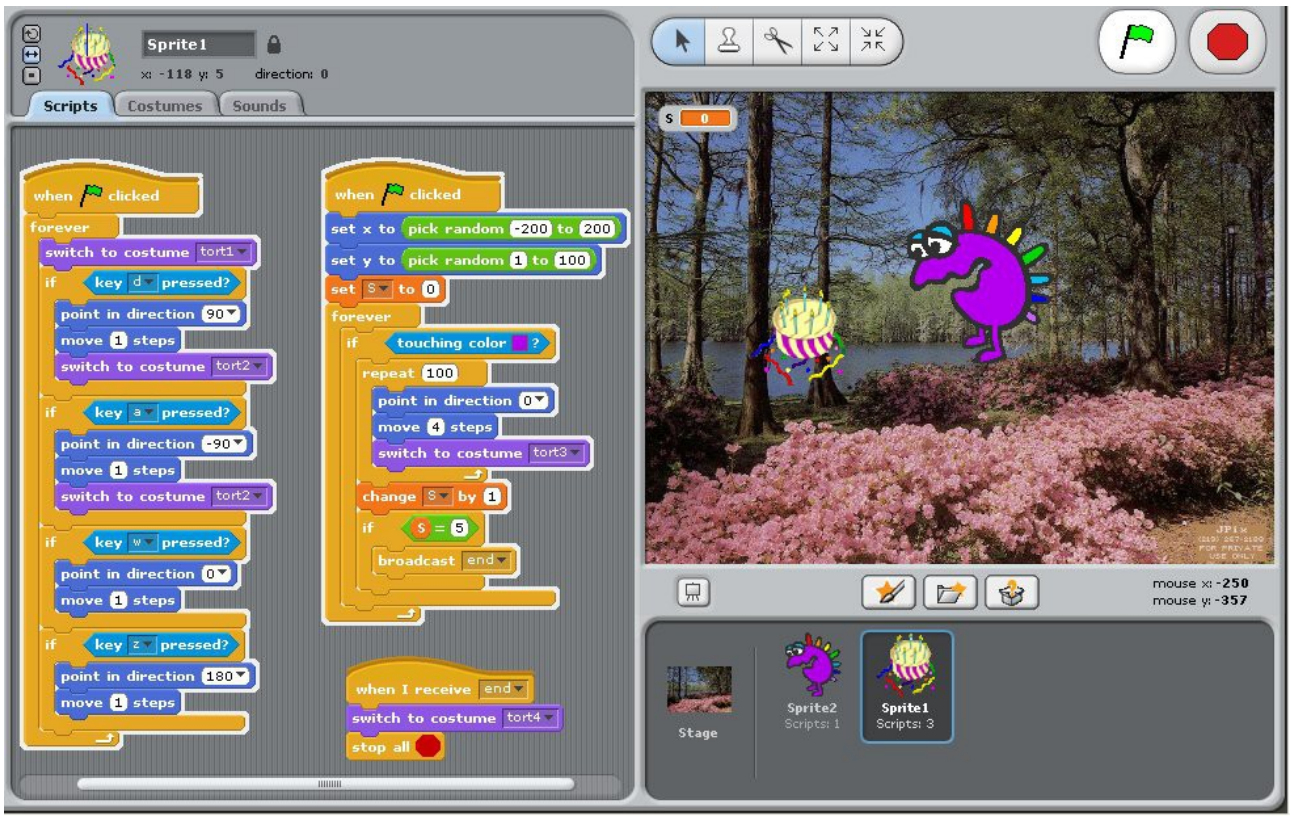


Как только жадный сластена укусит торт в пятый раз – тортик кончится и игра прекратится. Т.е. после достижения переменной значения 5 (S=5), тортик может исчезнуть, при помощи команды `hide`, а может просто изменить, свой вид, надев например новый костюм, состоящий из разноцветных точек.

Выберем такой вариант: когда переменная S принимает значение, равное пяти, используем команду `broadcast` **. передать сообщение.** Создадим сообщение **end,** которое передадим нашим объектам: тортик наденет новый костюм (разноцветные точки) и остановит выполнение программы, а уж какой вид примет наевшийся сластена можно спроектировать самим.

Запустим программу, и протестируем на работоспособность.

Программа для объекта «торт» может иметь следующий вид:



Дополнительное задание:

Попробуйте сделать реакцию на окончание игры, например, монстрик скажет что-то, попрощается, изменит внешний вид, заиграет музыка.