Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный
университет имени П.М. Машерова»
Кафедра информатики и информационных технологий

Т.Г. Алейникова, О.П. Оганджанян

ЗАДАЧНИК ПО ПРОГРАММИРОВАНИЮ В SCRATCH

Витебск
ВГУ имени П.М. Машерова
2018

УДК 004.415.25(076.5)
ББК 32.973.3я73
А45

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 6 от 27.06.2018 г.

Авторы: доцент кафедры информатики и информационных технологий ВГУ имени П.М. Машерова, кандидат физико-математических наук **Т.Г. Алейникова;** старший преподаватель кафедры информатики и информационных технологий ВГУ имени П.М. Машерова **О.П. Оганджанян**

Р е ц е н з е н т :
методист отдела по поддержке и развитию педагогических инициатив в работе с одаренными детьми ГУДОВ «ВОИРО» *О.В. Лозинская*

Алейникова, Т.Г.

А45 Задачник по программированию в Scratch / Т.Г. Алейникова, О.П. Оганджанян. – Витебск : ВГУ имени П.М. Машерова, 2018. – 44 с.

В учебном издании содержатся задачи различного уровня сложности, ориентированные на решение в среде программирования Scratch. Задачи сгруппированы по темам и охватывают основные возможности Scratch. Приведены решения, сформулированы дополнительные задания для самостоятельной работы. Особое внимание уделяется решениям олимпиадных задач.

Задачник адресован студентам педагогических специальностей, а также может использоваться учителями и школьниками на занятиях по Scratch-программированию.

УДК 004.415.25(076.5)
ББК 32.973.3я73

СОДЕРЖАНИЕ

ТЕМА 1. Движение и взаимодействие

Справочный материал: используются команды блока Управление: цикл всегда, ждать, стоп все; команды блока Движение: идти, перейти в, повернуться к, стиль вращения, если на краю, оттолкнуться; команды блока События: когда щелкнут по флажку, когда клавиша пробел нажата, передать сообщение, когда я получу сообщение; команды блока Внешность: установить размер, следующий костюм, думать, показаться и спрятаться.

ЗАДАЧА 1. «Мальчик с собакой»

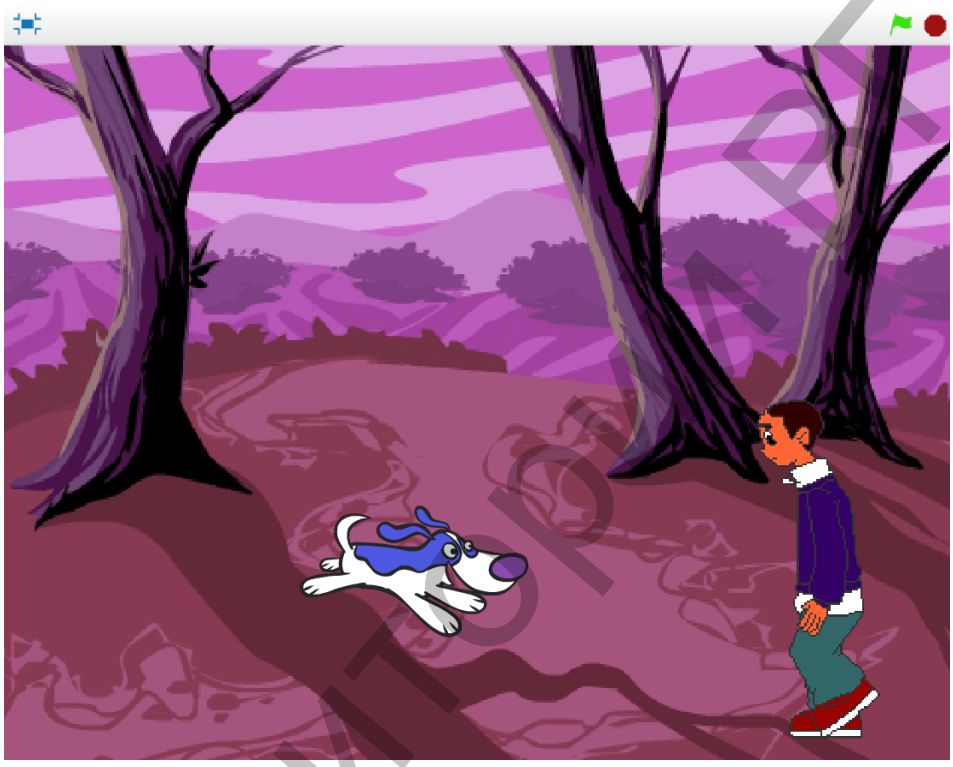

Рис. 1. Результат выполнения программы "Мальчик с собакой"

Сцена имеет два фона, которые выбираются из категории "На открытом воздухе" библиотеки. Смена фона происходит через определенное время или при нажатии клавиши. Мальчик прогуливается по сцене, отталкивается от края, после небольшой паузы думает: "Где моя собака?" и передает сообщение: "Потерялась собака". Собака показывается на сцене после того, как получит это сообщение и бежит навстречу мальчику (рис. 1). Когда нажимается клавиша пробел, тогда все остановится.

РЕШЕНИЕ.

При решении задачи предполагается параллельное и последовательное выполнение скриптов и взаимодействие спрайтов между собой через сообщения. На рис. 2 предлагается параллельное выполнение двух скриптов мальчика, когда произойдет событие "щелчок по флажку", на рис. 3 — параллельное выполнение двух скриптов собаки, когда произойдут события "щелчок по флажку", "нажатие клавиши пробел" и последовательное выполнение третьего скрипта обработки сообщения. На рис. 4 изображен скрипт сцены.

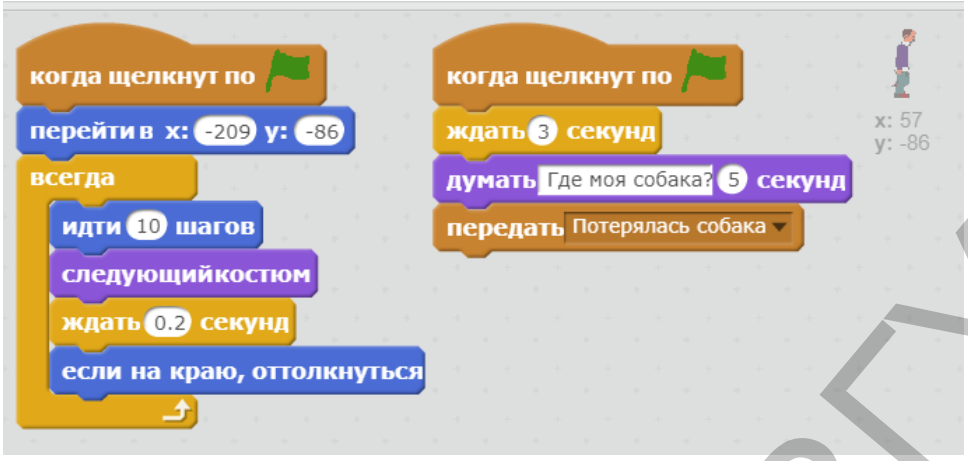

Рис. 2. Скрипты мальчика

Рис. 3. Скрипты собаки

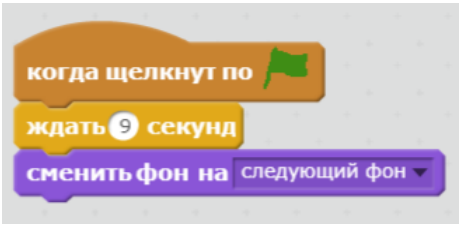

Рис. 4. Скрипт сцены

Команды в одном скрипте выполняются последовательно, а команды в разных скриптах с одинаковыми шапочками (когда произойдет событие "щелчок по флажку") выполняются параллельно (одновременно).

ТЕМА 2. Ветвления и циклы

Справочный материал: используются команды блока Управление: цикл всегда, конечный цикл, ждать, ветвление если-то; команды блока Движение: идти, перейти в, повернуться к, повернуться по часовой стрелке на число градусов, стиль вращения, если на краю, оттолкнуться; команды блока Операторы: больше, выдать случайное от до; команды блока Сенсоры: расстояние до, цвет касается; команды блока События: когда щелкнут по флажку, передать сообщение, когда я получу сообщение; команды блока Внешность: установить размер, сменить костюм на, показаться и спрятаться; команды блока Звук: играть звук.

ЗАДАЧА 2. «Голодная рыба»

Рис. 5. Результат выполнения программы "Голодная рыба"

Hungry fish перемещается с открытым ртом по сцене, следуя за указателем мыши на расстоянии 10 шагов. Рыбки Little fish, Little fish2, Little fish3 с небольшим интервалом появляются в левой части сцены и двигаются слева направо, поворачиваются случайным образом, отталкиваются от края. Если рот Hungry fish касается тела Little fish, то Hungry fish ее съедает: издает звук «chomp» и открывает-закрывает рот, а Little fish прячется и после паузы появляется в случайной точке левой части сцены (рис. 5).

РЕШЕНИЕ.

При решении задачи предполагается использование конечного цикла и датчика случайных чисел. На рис. 6 предлагается последовательное выполнение двух скриптов Hungry fish, на рис. 7 — скрипт Little fish. Скрипт создается для одной рыбки, а затем дублируется спрайт Little fish 2 раза вместе со скриптом (Little fish2, Little fish3). Сцена имеет фон underwater2, который выбирается из категории "На открытом воздухе" библиотеки.

Для того, чтобы Hungry fish следовала за указателем мыши по условию задачи на расстоянии 10 шагов используется команда "повернуться к указателю мышки", если расстояние до указателя мышки больше 10. Чтобы создать эффект открытия и закрытия рта у Hungry fish выполняется переход на вкладку Костюмы, дублируется костюм fish–open и переименовывается в fish–closed. В векторном режиме меняется форма рта. В скрипт Hungry fish добавляется конечный цикл "Повторить 2" раза.

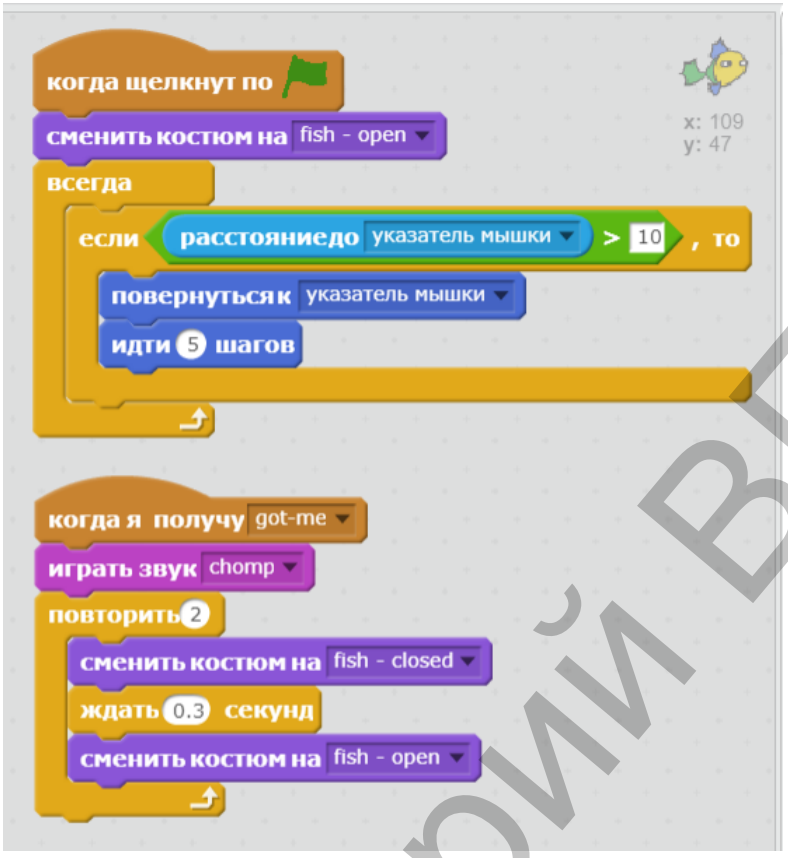

Рис.6. Скрипты спрайта Hungry fish

Для того, чтобы Little fish двигалась слева направо и поворачивались случайным образом выполняется команда движения вместе с оператором датчика случайных чисел: "повернуть по часовой стрелке на выдать случайное от -20 до 20 градусов". Аналогичным образом для появления Little fish после паузы в случайной точке левой части сцены — "перейти в x: -200 y: выдать случайное от -200 до 200".

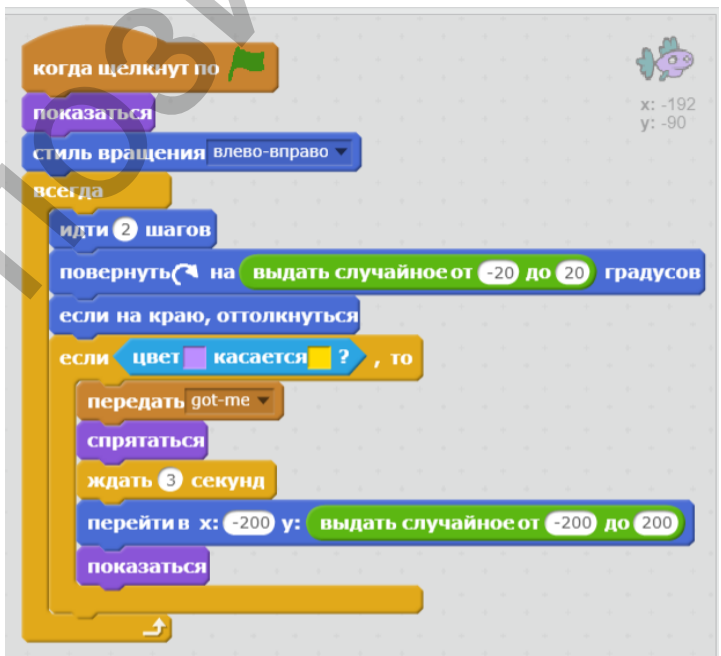

Рис. 7. Скрипт спрайта Little fish

ЗАДАНИЕ. Измените скрипты мальчика и собаки в ***Задаче 1:*** Если мальчик касается края, то меняется фон сцены, а мальчик появляется с противоположной стороны — эффект движения на меняющемся фоне. Собака при приближении к мальчику на расстояние меньше 100 шагов радостно лает и все останавливаются.

ЗАДАЧА 3. «Кто первый?»

Расстояние от старта до финиша 250 шагов. Все участники соревнования: динозавр, попугай, лиса и собака первоначально расположены на линии старта. У каждого спрайта выполняется свой скрипт, когда произойдет событие "когда клавиша пробел нажата". При достижении финиша победитель говорит: "Я – первый!" и все участники останавливаются. Кто (динозавр, попугай, лиса или собака) быстрее достигнет финиша?

РЕШЕНИЕ.

Для решения задачи рисуются линия Старта и линия Финиша на расстоянии 250 точек между ними. Например, как на рис. 8.

Рис. 8. Исходное положение спрайтов на линии Старта

Располагаются спрайты на линии Старта (рис. 9).

Рис. 9. Скрипт расположения спрайтов на линии Старта

Рис. 10. Основной скрипт задачи "Кто первый?"

Пишется скрипт для каждого участника соревнования (рис. 10). Дописываются скрипты с одинаковыми шапочками (когда произойдет событие "клавиша пробел нажата") в соответствии с условием задачи.

ЗАДАНИЕ. Измените скрипты спрайтов в ***Задаче 3:*** введите случайные данные в скорости движения участников забега.

Тема 3. Координаты. Перо. Графические эффекты

Справочный материал: используются команды блока Управление: конечный цикл, ждать, ветвление если-то; команды блока Движение: повернуться к, повернуть в направлении, установить x в, установить y в, перейти в x: y:, плыть секунд в точку x: y:; команды блока Операторы: меньше; команды блока Сенсоры: расстояние до; команды блока События: когда щелкнут по флажку, когда спрайт нажат, передать сообщение, когда я получу сообщение; команды блока Внешность: показаться и спрятаться, сменить костюм на, изменить эффект на, говорить в течении секунд; команды блока Перо: очистить, опустить перо, поднять перо, установить цвет для пера, установить размер пера.

ЗАДАЧА 4. «Обведи рисунок»

На сцене изображен конверт (рис. 11). Координаты отмеченных точек: (-150;-100), (150;-100), (150;30), (-150;30), (0;100). Обведите рисунок, не поднимая пера и не проводя дважды по одной линии.

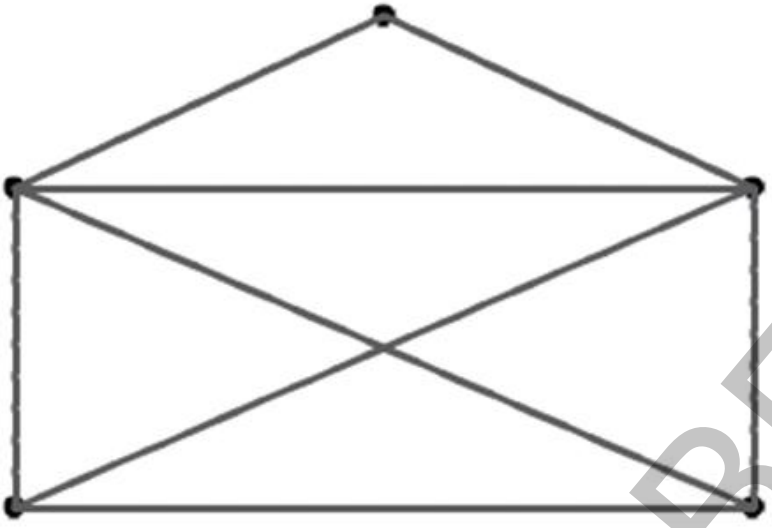

Рис. 11. Фон сцены с изображением конверта

РЕШЕНИЕ.

Для решения задачи подготавливается фон сцены с изображением конверта (рис. 11). Выбирается спрайт Pencil из категории "Предметы" библиотеки. Можно снять флажок Показать в Инфо спрайта. Устанавливается начальное положение карандаша, размер и цвет пера (рис. 12). Продумывается алгоритм перехода из точки в точку с заданными координатами (рис. 13).

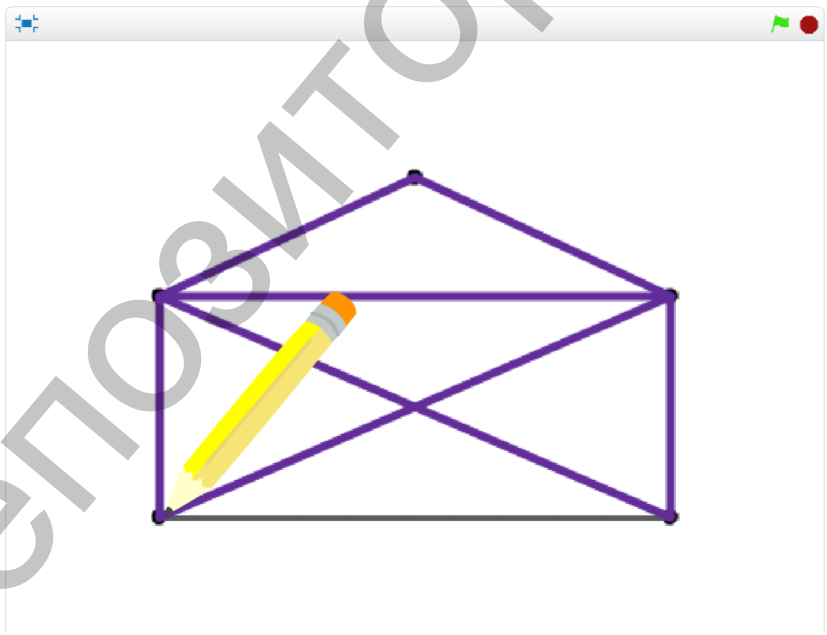

Рис. 12. Результат выполнения программы "Обведи рисунок"

Рис. 13. Скрипт изображения конверта

ЗАДАЧА 5. «День рождения»

Голубой вертолет плывет по сцене и приземляется около волшебника. Оставляет коробку с подарком и улетает (рис. 14). Волшебник говорит фразу: «Щелкни по коробке мышкой, чтобы она открылась!» (рис. 15). Вы щелкаете мышкой по коробке, и она превращается в торт со свечами. Через 2 секунды торт становится коробкой, а через 1 секунду коробка исчезает.

Рис. 14 Результат выполнения программы "День рождения"

Рис. 15. Действия Волшебника

РЕШЕНИЕ.

Для решения задачи загружается фон сцены party из категории "В помещении" библиотеки спрайтов. Загружается спрайт Helicopter из категории "Транспорт". На вкладке Костюмы в растровом режиме отображается вертолет слева-направо. Пишется скрипт для Helicopter (рис. 16).

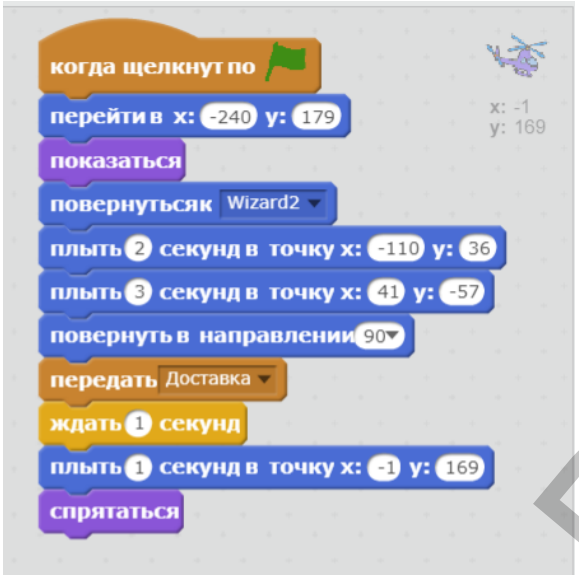

Рис. 16 Скрипт спрайта Helicopter

Загружается спрайт Wizard2 из категории "Фантастика". Размещается Волшебник на сцене в точку (x:167 y:-11). Пишется скрипт для Wizard2 (рис. 17).

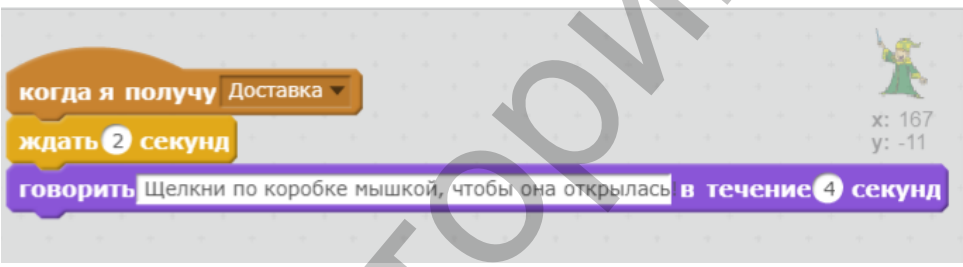

Рис. 17. Скрипт спрайта Wizard2

Загружается спрайт Gift из категории Предметы. Размещается на сцене в точке (74,-63). На вкладке Костюмы переименовывается костюм gift-a в Коробка, добавляется новый костюм cake-a кнопкой Выбрать костюм из библиотеки (рис. 18). Пишется скрипт для Gift (рис. 19). Для того, чтобы выполнить условие задачи: через 2 секунды торт становится коробкой, а через 1 секунду коробка исчезает, применяются эффект "Призрак" и смена костюма.

Рис. 18. Выбор костюма для спрайта Gift

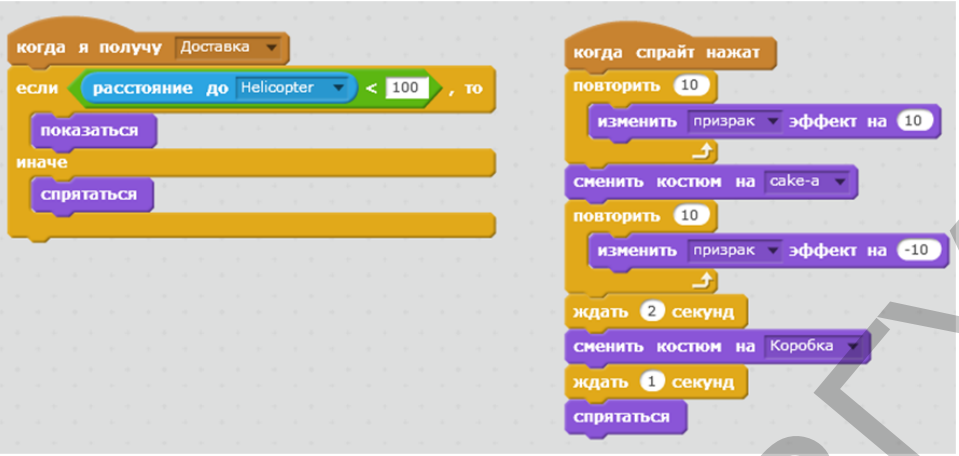

Рис. 19. Скрипт спрайта Gift

ТЕМА 4. Процедуры

Справочный материал: используются команды блоков Управление; Движение; Операторы; Сенсоры; Внешность; События; Перо; Другие блоки: создать блок (процедуры).

ЗАДАЧА 6. «Кот прыгает и ползает»

Кот перемещается по сцене, при приближении к первому препятствию перепрыгивает его, а под следующим — проползает (рис. 20).

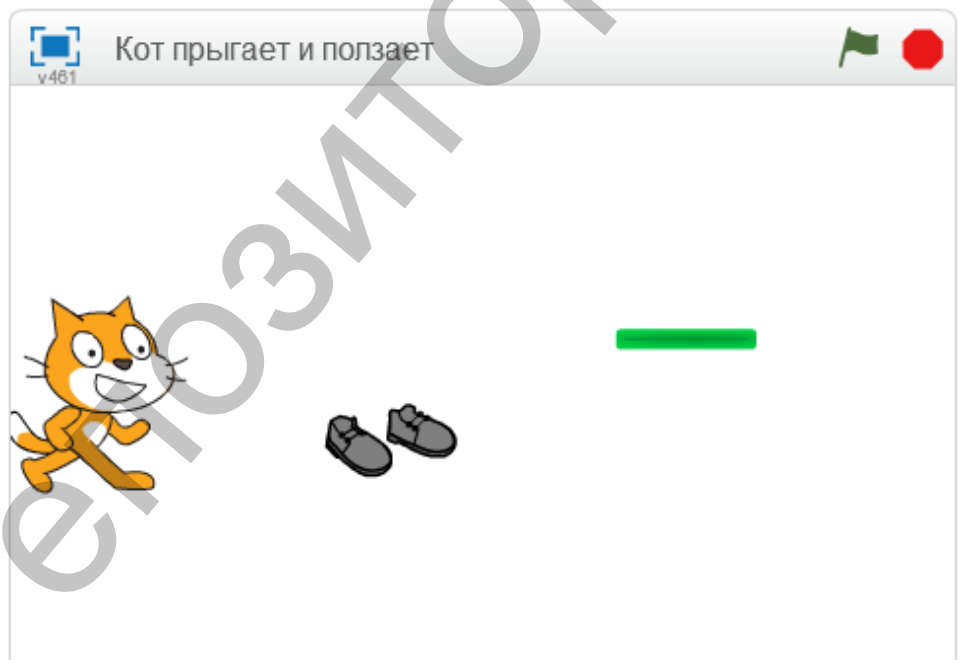

Рис. 20. Исходное положение спрайтов задачи 6

РЕШЕНИЕ.

Сначала выбираются из библиотеки необходимые спрайты (переименованы в «кот», «обувь», «планка») и размещаются на сцене. Устанавливается в свойствах спрайта кот стиль вращения влево-вправо. В палитре Другие блоки создаются три новые процедуры (Другие блоки): стартовать, прыгать, ползать. Основной скрипт кота (рис. 21) включает цикл и проверку условий для определения расстояния до препятствий (обувь и планка).

Рис. 21. Основной скрипт кота

Процедура ***стартовать*** включает команды, которые необходимо выполнить для того, чтобы спрайт занял исходное положение (рис. 22). Здесь можно также установить начальные свойства (стиль вращения, показать-спрятать и т.п.)

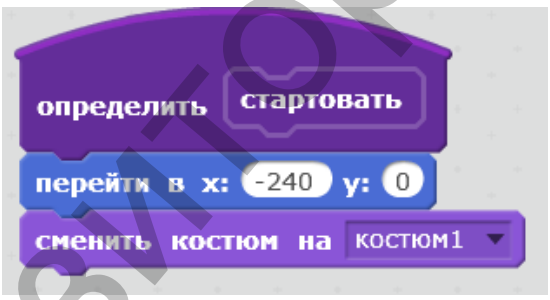

Рис. 22. Скрипт процедуры ***стартовать***

Команды процедур ***прыгать*** и ***ползать*** изображены на рис. 23. Прыжок достигается изменением координат спрайта, пауза необходима для того, чтобы пользователь успевал заметить перемещение. Чтобы ползание выглядело более естественным, добавим костюм cat2 из библиотеки костюмов.

Этот пример показывает, как можно расширить список имеющихся команд в палитре блоков, придумывая свои собственные. Кроме того, поэтапное проектирование алгоритма решения позволяет разделить сложную задачу на более простые подзадачи и реализовывать их поочередно с помощью такого мощного инструмента как процедуры.

ЗАДАНИЕ. Выберите из библиотеки спрайтов своего героя и научите выполнять какие-нибудь действия, применяя процедуры (Другие блоки).

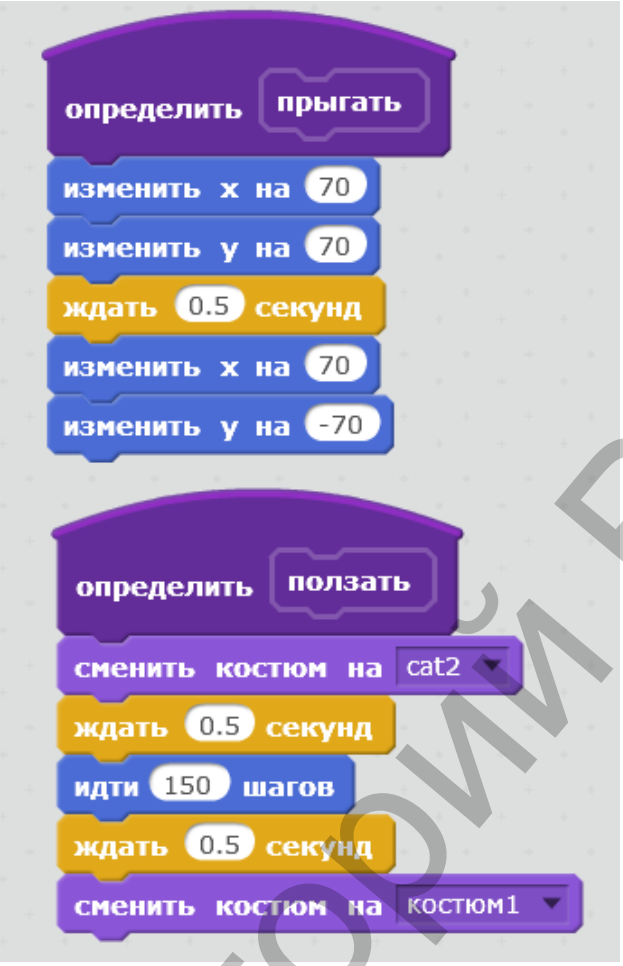

Рис. 23. Скрипт процедур ***прыгать***, ***ползать***

ЗАДАЧА 7. «Вращающиеся квадраты»

Используя процедуры с параметрами, нарисуйте вращающиеся квадраты (рис. 24).

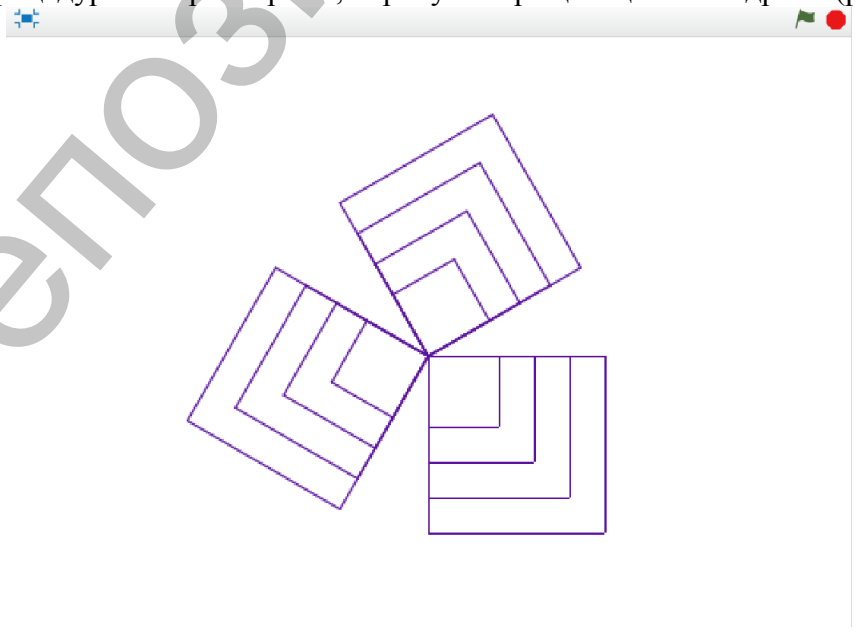

Рис. 24. Результат выполнения программы "Вращающиеся квадраты"

РЕШЕНИЕ.
Для решения задачи используются две процедуры с параметром и одна процедура без параметра. В процедуре ***Квадрат*** параметр "сторона" принимает значения: 40, 60, 80 и 100. Вызов процедуры осуществляется в процедуре ***Квадраты***, в которой все значения передаются параметру последовательно (рис. 25).

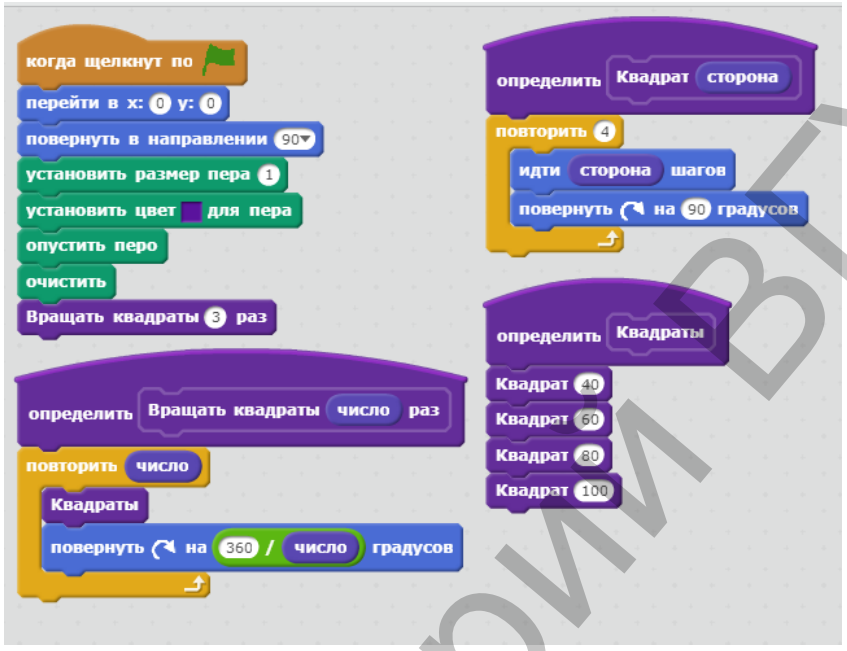

Рис. 25. Основной скрипт задачи "Вращающиеся квадраты"

В процедуре ***Вращать квадраты*** значение параметра "число" используется как делитель в операторе деления команды "повернуть по часовой стрелке на *360/число* градусов" и как переменная конечного цикла.
ЗАДАНИЕ. Используя процедуры с параметрами и без, нарисуйте вращающийся флажок (рис. 26), многоугольники (рис. 27), узор из квадратов (рис. 28).

Рис. 26. Результат выполнения программы "Вращающийся флажок"

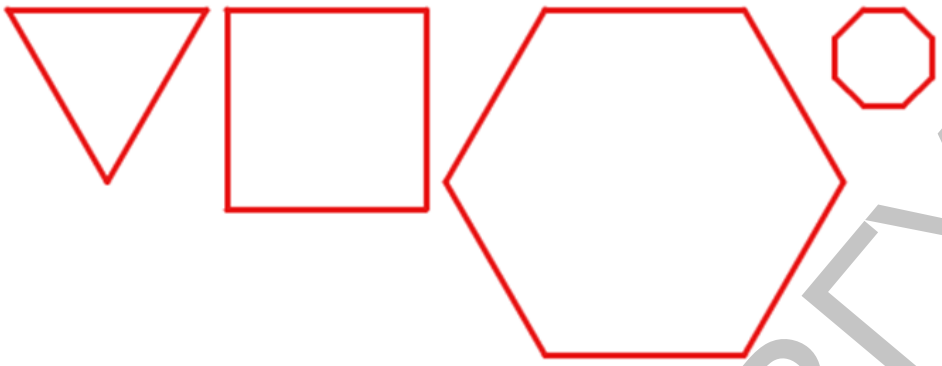

Рис. 27 Результат выполнения программы "Многоугольники"

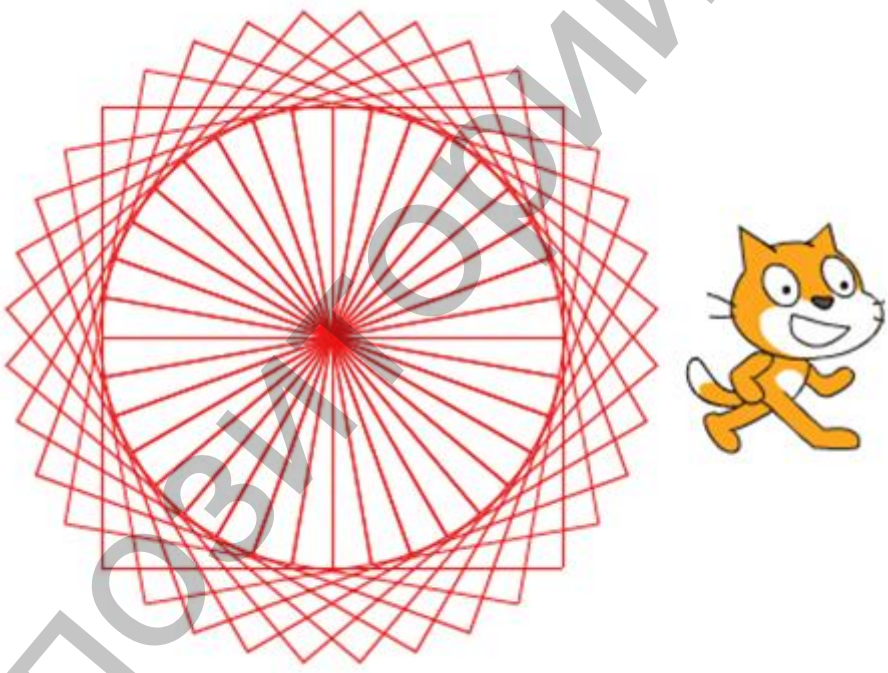

Рис. 28. Результат выполнения программы "Узор из квадратов"

ЗАДАЧА 8. «Дом»

Получите изображение дома (рис. 29). Обеспечьте возможность масштабирования изображения.

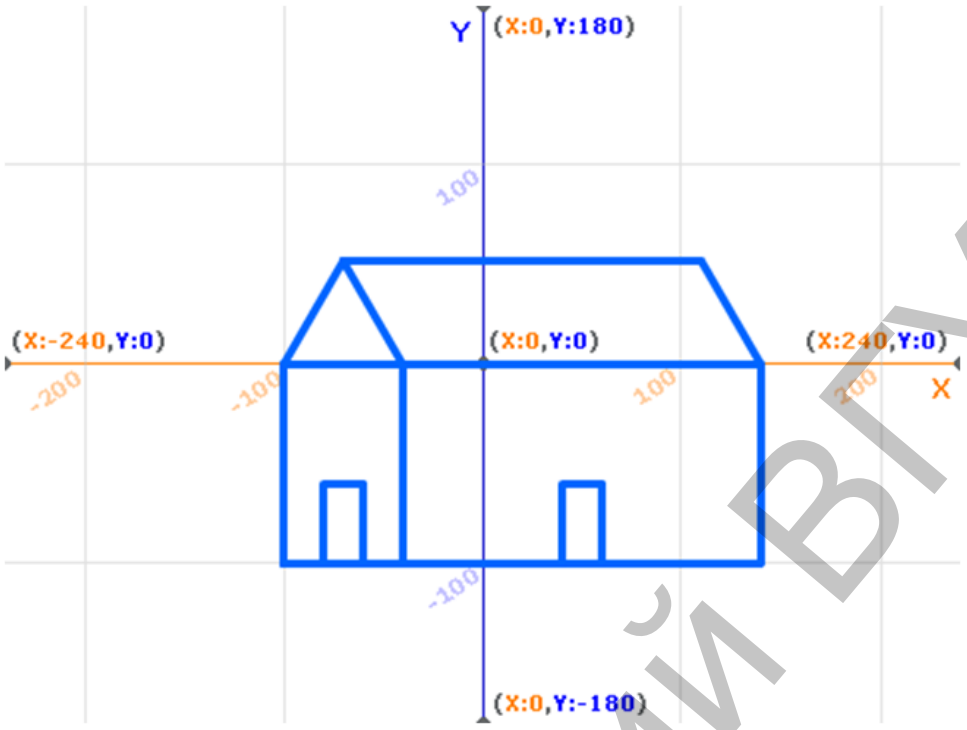

Рис. 29. Результат выполнения программы «Дом»

РЕШЕНИЕ.

Этот проект подробно рассмотрен в [1, с. 106] как пример программирования сверху вниз. Задача разделена на подзадачи, которые реализованы с помощью процедур с параметрами.

Определим исходный масштаб 20 = 1 единице для получения изображения, представленного на рис. 29. Рисование осуществляется поэтапно, в соответствии со следующим планом (рис. 30).

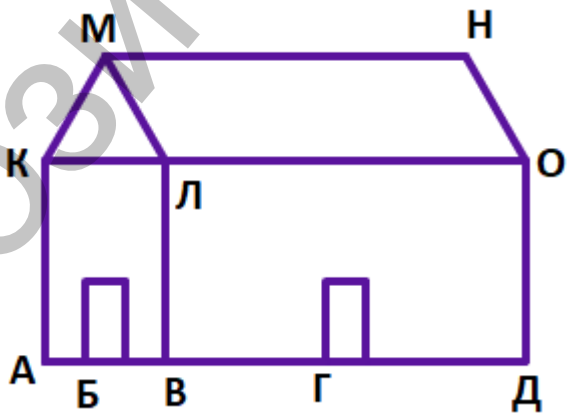

Рис. 30. План изображения дома

Сторона **АВЛК** (размер 5х3 единиц). Закачиваем в точке **А**.

Перемещаемся на 1 единицу по горизонтали в точку **Б**. Рисуем дверь (размер 2х1 единиц).

Перемещаемся на 2 единицы по горизонтали в точку **В**. Рисуем сторону **ВДОЛ** (размер 5х9 единиц). Заканчиваем в точке **В**.

Перемещаемся на 4 единицы по горизонтали в точку **Г**. Рисуем вторую дверь.

Возвращаемся в точку **А** (назад на 7 единиц), переходим в точку **К** (вверх на 5 единиц).

Крыша состоит из двух фигур: треугольника и параллелограмма. Рисуем треугольник **КЛМ**. Переходим в точку **О** (12 единиц по горизонтали). Рисуем параллелограмм **ОНМЛ** (размер 3х9 единиц).

Для начала прячется изображение рисующего спрайта, используя меню свойств (кнопка **i**). Основной скрипт программы «Дом» изображен на рис. 31.

Рис. 31. Основной скрипт программы «Дом»

Реализация плана построения выполняется в процедуре ***Дом*** с параметром ***масштаб*** – (рис. 32.).

Рис. 32. Скрипт процедуры ***Дом***

В этой процедуре используется несколько вложенных процедур: ***Сторона 1***, ***Дверь***, ***Сторона 2*** (код этих процедур приведен на рис. 33.

Рис. 33. Скрипт процедур ***Сторона 1***, ***Дверь***, ***Сторона 2***

Процедура ***Крыша*** также использует вложенную процедуру – ***Треугольник*** (рис. 33).

Рис. 34. Скрипт процедур ***Крыша*** и ***Треугольник***

Изменяя масштаб, можно получать изображения домов разного размера. Эта программа демонстрирует, какие возможности заложены в процедурном стиле программирования.

ТЕМА 5. Клоны. Счетчики

Справочный материал: используются команды блоков Управление: создать клон себя самого, когда я начинаю как клон, повторять пока не, ждать до; команды блоков Движение; Сенсоры; Операторы; Внешность; События; Данные: Создать переменную – жизни; Другие блоки.

ЗАДАЧА 9. «Шарики»

Сверху вниз падают разноцветные шарики. Кот, двигаясь под управлением клавиш-стрелок, поражает их специальным оружием, увеличивая количество жизней (рис. 35). Шары, касаясь Кота, уменьшают количество жизней. Игра прекращается, если жизней не осталось.

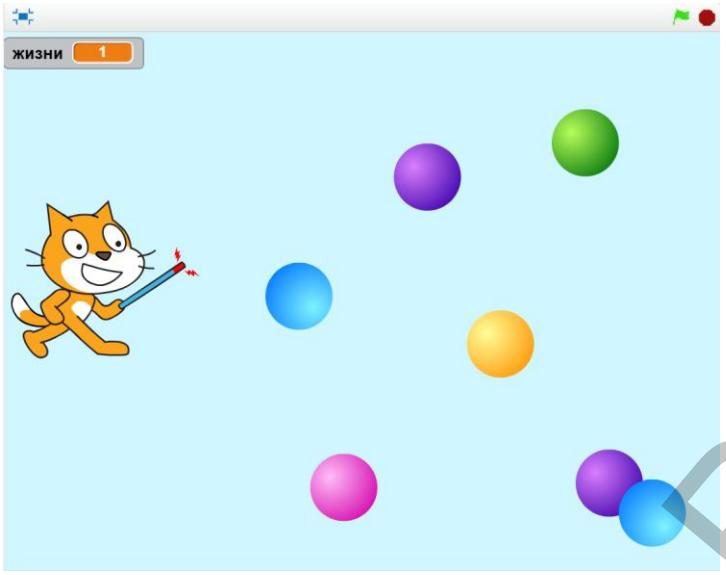

Рис. 35. Результат выполнения программы "Шарики"

РЕШЕНИЕ.

В качестве оружия для поражения шариков используется предмет из библиотеки Magicwand, который добавляется в костюмы (не к спрайтам!), а затем "перетаскивается" в костюм Кота (рис. 36). Используя инструмент заливки, верхняя часть волшебной палочки окрашивается в красный цвет. Используется один костюм Кота, анимация движения ногами в данном проекте не программируется.

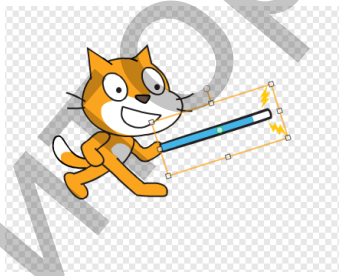

Рис. 36. Костюм Кота с волшебной палочкой

Скрипт для движения Кота может быть с независимой обработкой нажатия клавиш (рис. 37) или с использованием процедуры ***двигаться*** в основном алгоритме (рис. 38).

Рис. 37. Скрипт движения Кота (вариант1)

Рис. 38. Скрипт движения Кота (вариант2)

В процедуру ***двигаться*** можно добавить не только параметр ***направление***, но и еще один — ***скорость***.

Разноцветные шарики появляются в случайном месте сцены и движутся сверху вниз. Чтобы не создавать несколько спрайтов, используется клонирование. В костюмах библиотечного спрайта Ball есть несколько вариантов цвета. При создании нового клона меняется костюм. Оригинал спрайта спрятан. Скрипт спрайта Ball приведен на рис. 39.

Шарики остаются на экране, отталкиваясь от края, пока их не коснется Кот красным кончиком волшебной палочки (условие цикла пока не). В случае, когда Кот попадает по шарику у него увеличивается количество жизней (переменная жизни), если шарик касается Кота (желтого цвета), то передается сообщение ***поражен*** и количество жизней уменьшается (рис. 40).

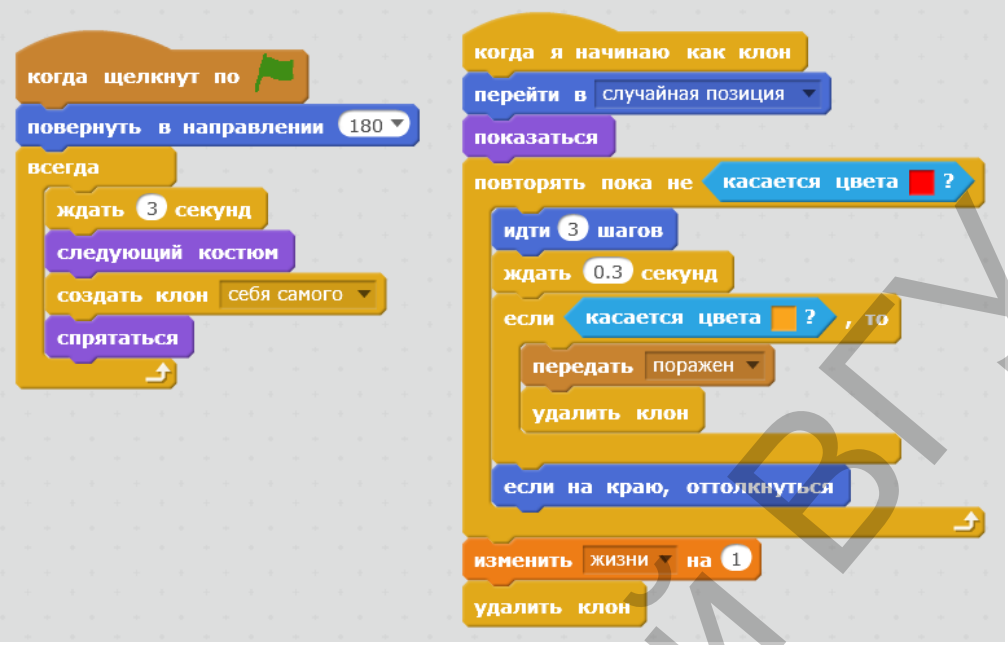

Рис. 39. Скрипт спрайта Balls

Рис. 40. Скрипт Кота

На старте проекта значение переменной жизни устанавливается 3. Завершается игра, когда жизни заканчиваются, для этого используется цикл ***ждать до***. Завершающий цикл обеспечивает постепенное исчезновение Кота за счет применения эффекта призрак.

ЗАДАНИЕ. Создайте второй уровень игры. Усложните прохождение этого уровня каким-либо образом (увеличением скорости движения шариков, дополнительными препятствиями и т.п.).

ТЕМА 6. Списки. Строки

Справочный материал: используются команды блоков Управление, Движение, Сенсоры, Операторы, Внешность, События, Данные: Создать переменную — Вопрос, Игрок, Создать список, Другие блоки.

ЗАДАЧА 10. «Викторина "Как я знаю Скретч?"»

Идея викторины взята из официального сайта проекта Scratch [2]. Кот Скретч задает вопросы о программе Скретч, обращаясь к Игроку по введенному им имени (рис. 41). Ведется подсчет вопросов. Смена фона происходит на каждом вопросе (желтый цвет) (рис. 42) и ответе игрока (зеленый и красный цвет фона) (рис. 43, рис. 44). Вопросы викторины группируются в список. Варианты ответов предлагаются спрайтами Гобо1, Гобо2, всегда, если. Реакция программы на правильность ответа Игрока реализуется спрайтами Кнопка1 и Кнопка2 (рис. 45).

Рис. 41. Интерактивное взаимодействие с Игроком

Рис. 42. Первый вопрос викторины

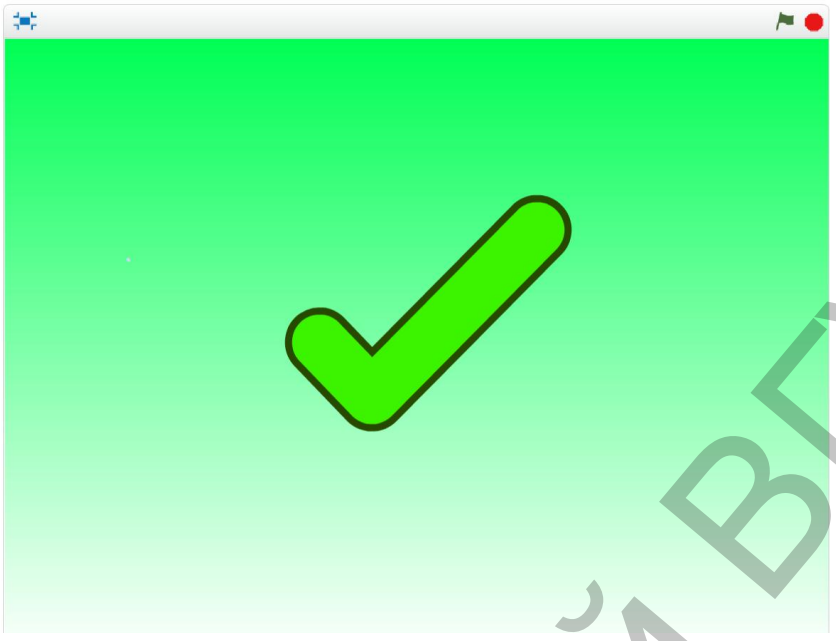

Рис. 43. Правильный ответ Игрока

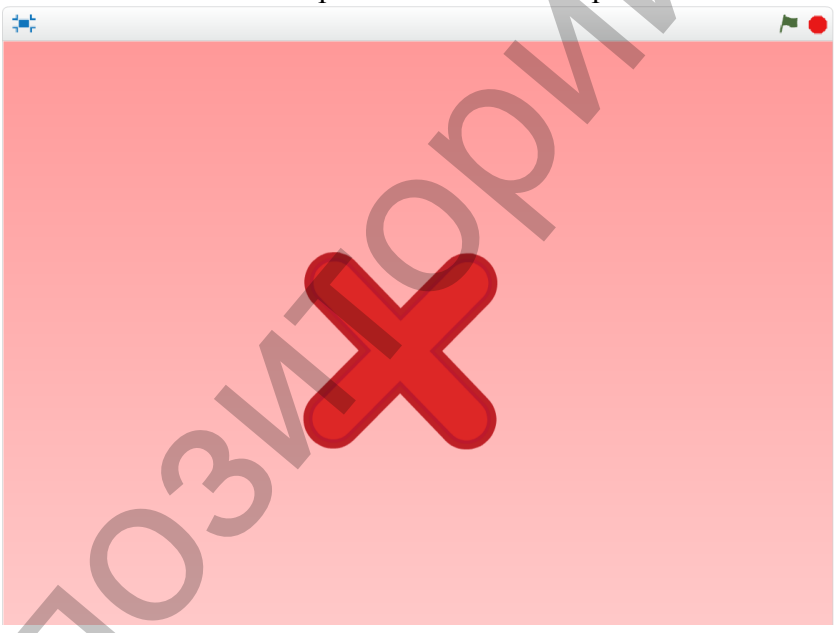

Рис. 44. Неправильный ответ игрока

Рис. 45. Спрайты викторины

РЕШЕНИЕ.

Список – это тип, позволяющий группировать данные. Набор нескольких величин можно назвать одним именем и обращаться к его элементам по порядковому номеру. Решение задачи предполагает формирование списка в скрипте кота в процедуре ***подготовить вопросы*** (рис 46), а удаление содержимого списка в скрипте спрайта Кнопка1 (рис. 47).

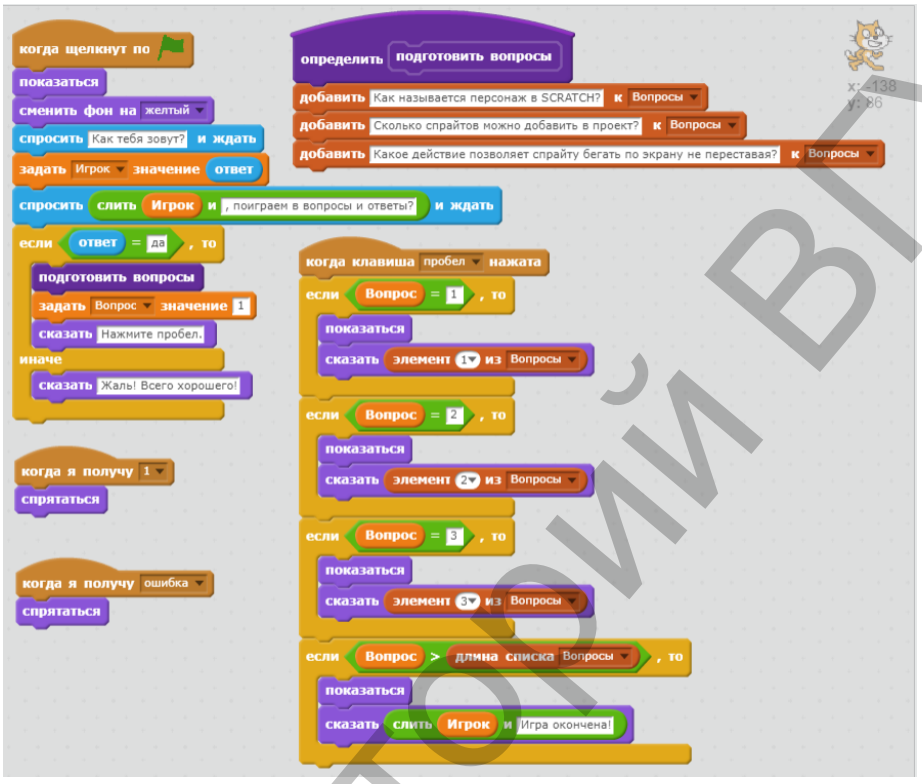

Рис. 46. Скрипт Кота Скретч

Переменная Игрок содержит строковый тип данных, так как пользователь в качестве ответа на вопрос "Как тебя зовут?" может ввести не только имя, но и фамилию. Переменная Вопрос относится к целочисленному типу. Значение переменной Игрок используется командой слить, а значение переменной Вопрос — как счетчик.

Рис. 47. Скрипт спрайта Кнопка1

Скрипт спрайта Кнопка2 (рис. 48). Гобо1 и Гобо2 участвуют в первых двух вопросах викторины. Гобо1 на 1 вопрос предлагает правильный ответ, на второй — неправильный, а Гобо2 – наоборот, поэтому скрипт спрайта Гобо1 (рис. 49) можно скопировать и отредактировать для Гобо2 (рис. 50).

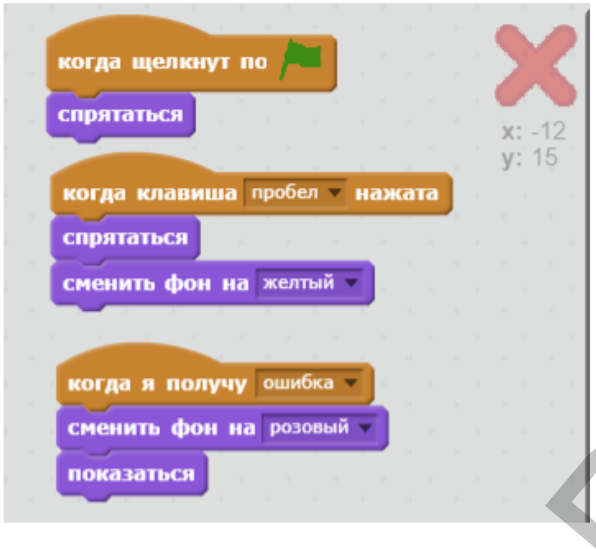

Рис. 48. Скрипт спрайта Кнопка2

Рис. 49. Скрипт спрайта Гобо1

Рис. 50. Скрипт спрайта Гобо2

Спрайты всегда и если участвуют в третьем вопросе викторины. Всегда является правильным ответом, а если – неправильным, поэтому скрипт спрайта всегда (рис. 51) также можно скопировать и отредактировать для если (рис. 52).

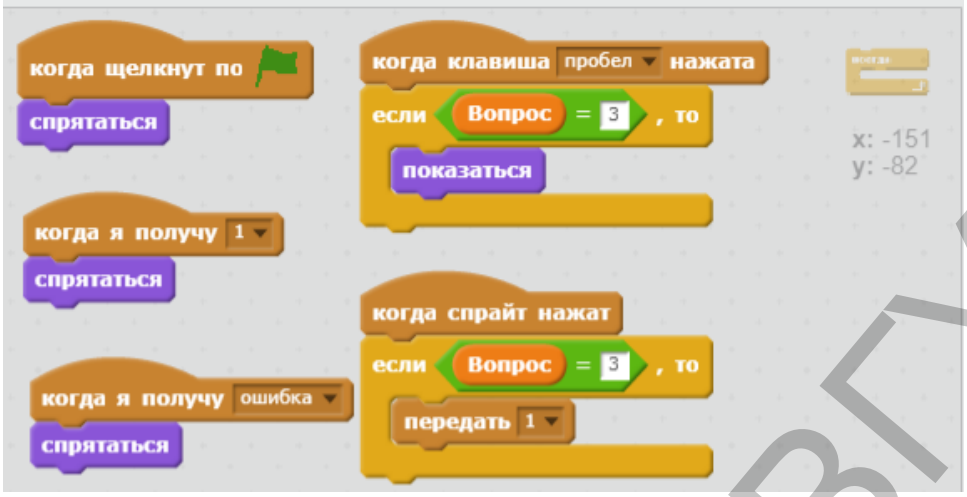

Рис. 51. Скрипт спрайта всегда

Рис. 52. Скрипт спрайта если

ЗАДАНИЕ. В ***Задаче 4*** координаты точек сохраните в списках (рис. 53). Измените скрипт карандаша таким образом, чтобы в решении задачи использовать списки (рис. 54).

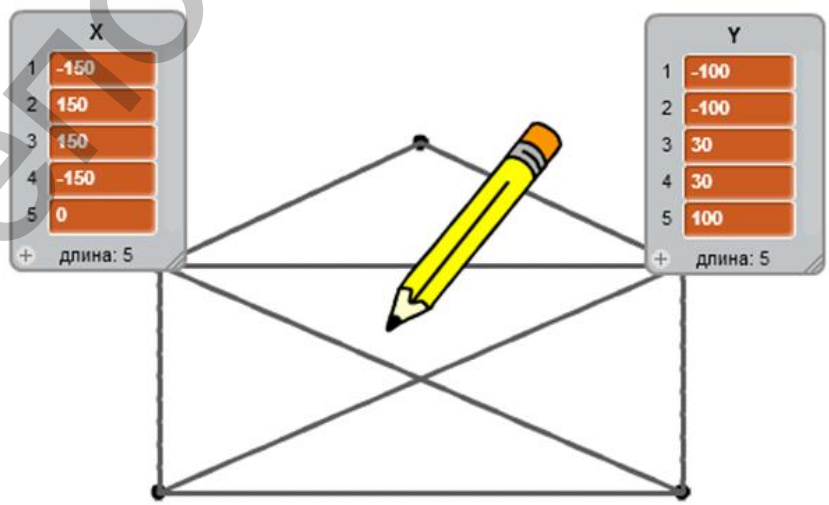

Рис. 53. Координаты точек в списках

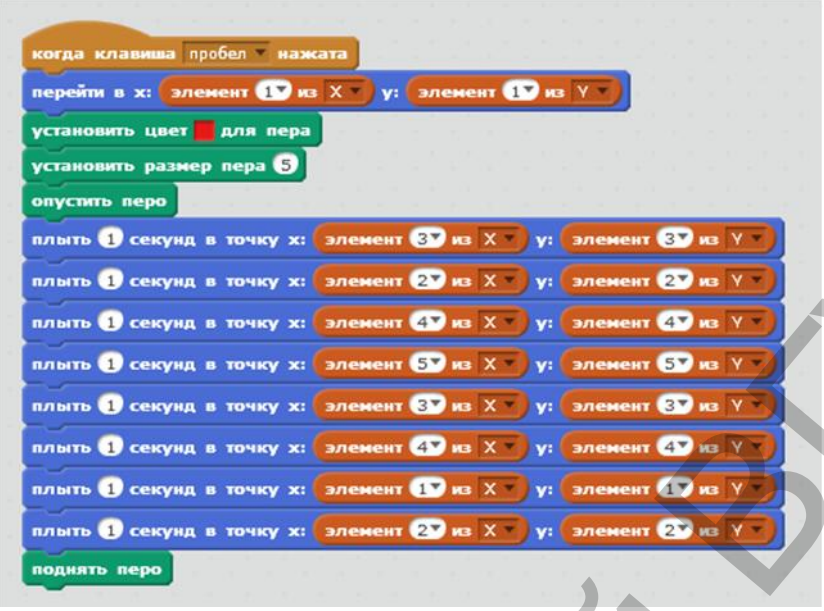

Рис. 54. Скрипт изображения конверта с использованием списков

Тема 7. Олимпиадные задачи

Справочный материал: используются команды блоков Управление, Движение, Сенсоры, Операторы, Внешность, События, Данные, Другие блоки.

ЗАДАЧА 11. «Лестница»

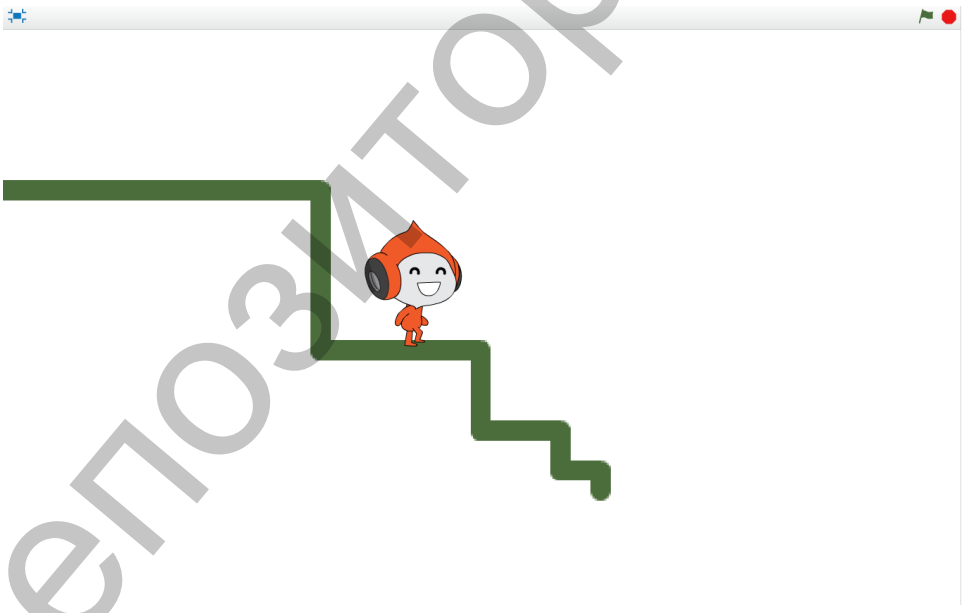

Рис. 55. Результат выполнения программы "Лестница"

Сцена имеет три фона с изображением лестниц разного размера. Ширина верхней ступеньки лестницы задана и равна *n*, высота каждой ступени в два раза меньше ее ширины, ширина следующей ступени равна высоте предыдущей. Ступеньки заканчиваются, когда ширина очередной ступени становиться меньше 20. При запуске проекта фон выбирается случайно. Помоги Пико спуститься по ступенькам лестниц с *n*=160 (фон1), *n*=200 (фон2), *n*=120 (фон3). После спуска Пико отходит от лестницы на 40 шагов и говорит "Ух!" (рис. 55).

РЕШЕНИЕ.

Для подготовки фоновых рисунков с лестницами заданных размеров можно использовать скрипт изображения ступеней (рис. 56). В результате выполнения проекта на сцене рисуется лестница красного цвета с начальной ступенькой длины 200. Сохранить полученное изображение можно, воспользовавшись в контекстном меню сцены функцией "save picture of stage". Изменяя начальные данные (цвет и размер), получим еще два изображения для фонов проекта "Лестница".

Рис. 56. Скрипт изображения ступеней

Решение задачи предполагает случайный выбор одной из трех лестниц, по которой должен двигаться Пико. Для этого используется переменная *RandNumber*. Ее значение определяется в процедуре ***стартовать***, в зависимости от него выбирается фон и задается значение переменной *n*. Вложенные команды если-иначе исключают повторные проверки условия (рис. 57).

Движение Пико обеспечивается тремя командами ***повторить*** и процедурой ***шагать***. Количество повторений циклов вычисляется делением переменной n на 4 и 8 (т.к. в процедуре ***шагать*** движение происходит на 4 шага). В случае, если результат деления не целый, происходит его автоматическое округление.

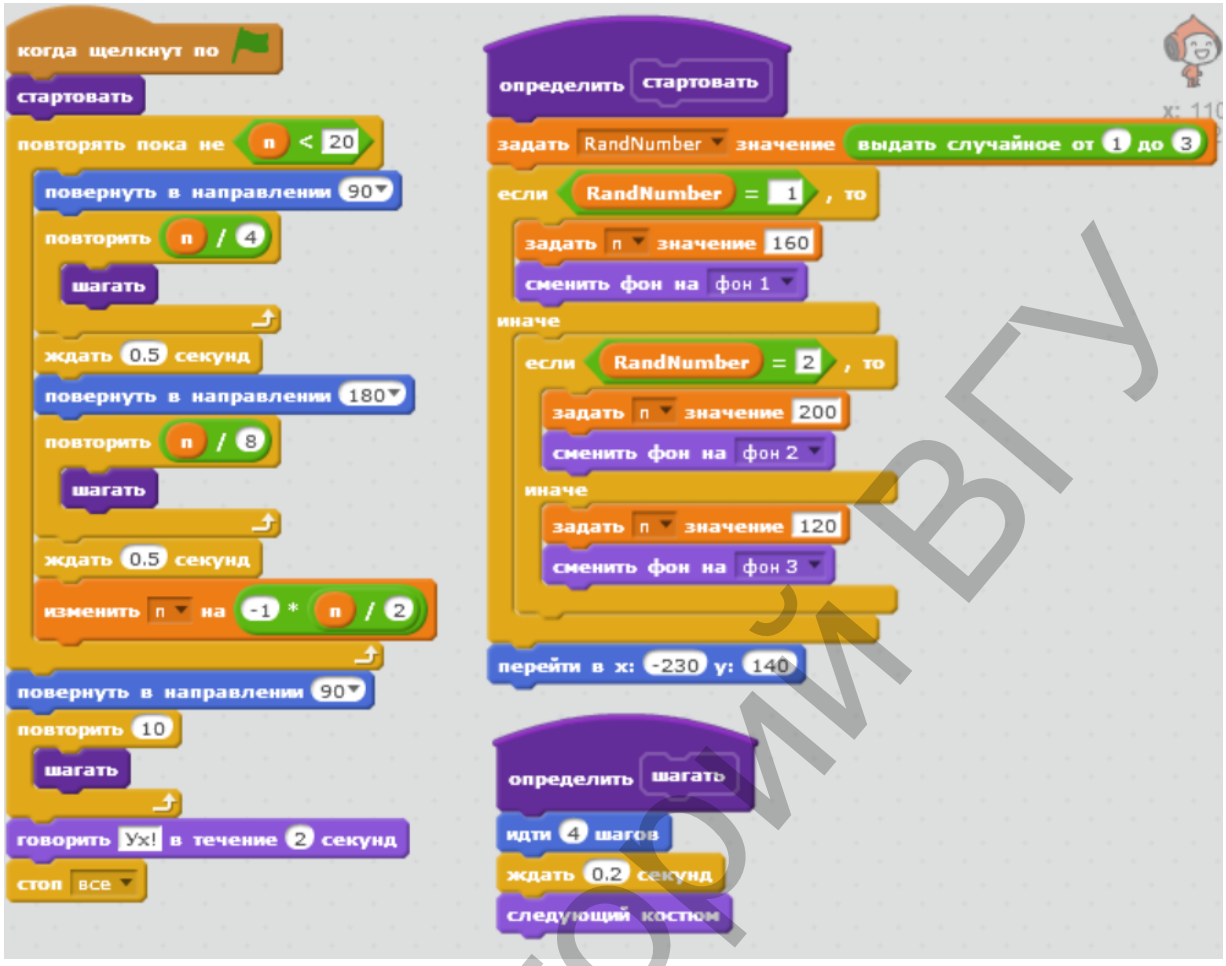

Рис. 57. Скрипт Пико

ЗАДАНИЕ. Модифицируйте процедуру ***шагать*** таким образом, чтобы в ней учитывалось направление движения.

ЗАДАЧА 12. «Буфет»

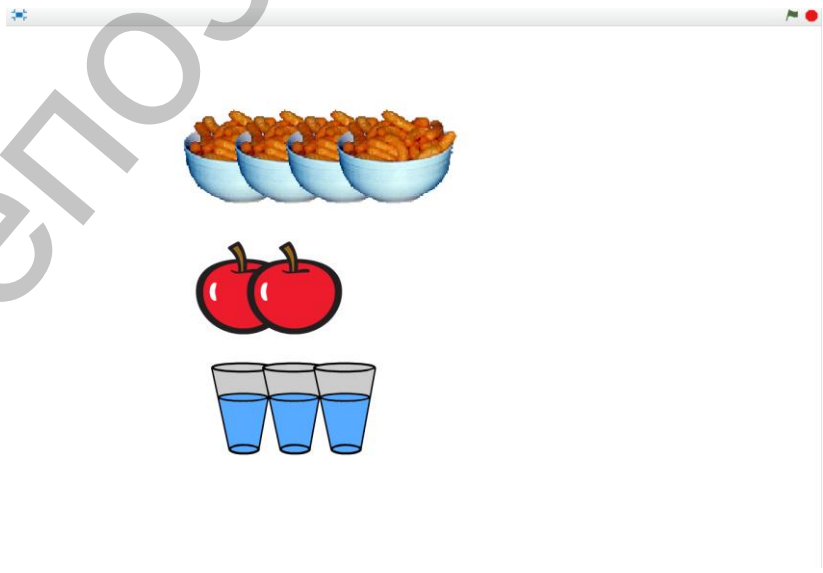

Рис. 58. Результат выполнения программы "Буфет"

В буфете имеется три вида блюд. Количество блюд каждого вида задается случайно. Пользователь выбирает блюдо с помощью мышки, нажимая на его изображение, после чего оно исчезает с экрана. Если какой-либо вид еды закачивается, происходит его пополнение (случайным количеством). Щелкать можно произвольное число раз по любому объекту (рис. 58).

РЕШЕНИЕ.

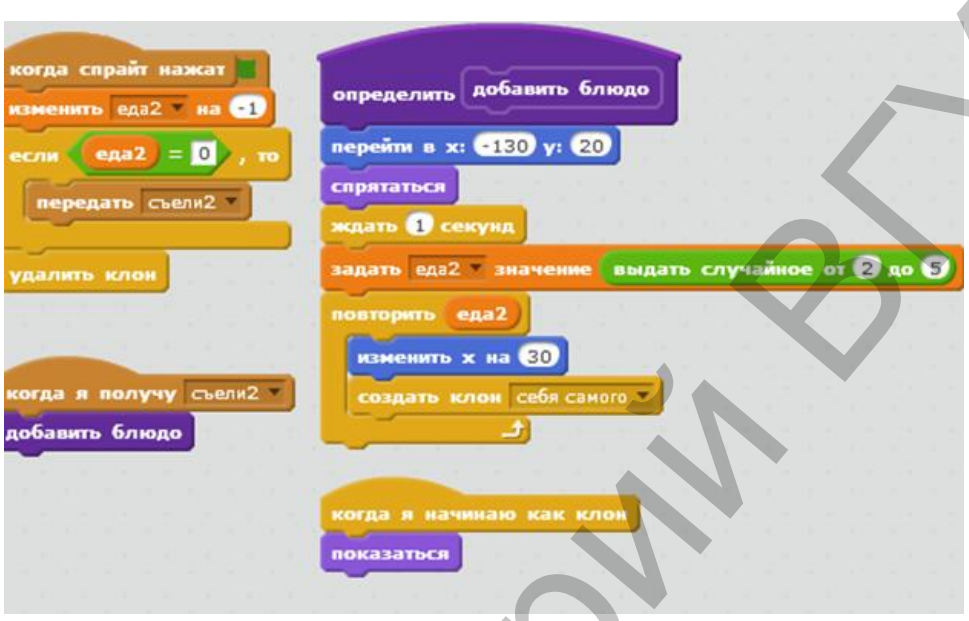

Рис. 59. Скрипт одного из видов еды

Для каждого блюда создается переменная *едаN*, начальное значение которой определяется случайным образом. Эта переменная управляет количеством видимых пользователем изображений блюда – клонов, оригинал спрятан. Пополнение блюд реализовано с помощью сообщений. Это позволяет при старте проекта добавить блюда, передав сообщения в скрипте сцены.

ЗАДАЧА 13. «Банкомат»

Рис. 60. Результат выполнения программы "Банкомат"

В банкомате имеется достаточное количество купюр номиналами 50, 20, 10 и 5 (рис. 60). Пользователь по запросу вводит сумму (не больше 500) и получает ее минимальным количеством купюр. Изображение купюр появляется на сцене, пользователю выдается сообщение. Предусмотрите обработку некорректного ввода запрашиваемой суммы.

РЕШЕНИЕ.

Начнем с создания списка с номиналами купюр (рис. 61).

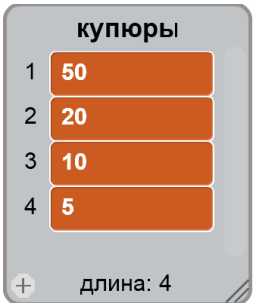

Рис. 61. Список с номиналами купюр

Обратите внимание на то, что значения в списке расположены в порядке убывания. Это позволяет упростить дальнейшее решение задачи.

Чтобы разменять заданную пользователем сумму (переменная *сумма*) наименьшим количеством купюр, используется известный в программировании "жадный алгоритм". Введем переменную *i* – номер купюры в списке. Количество купюр (переменная *n*) определяется как результат целочисленного деления оставшейся суммы на значение номинала.

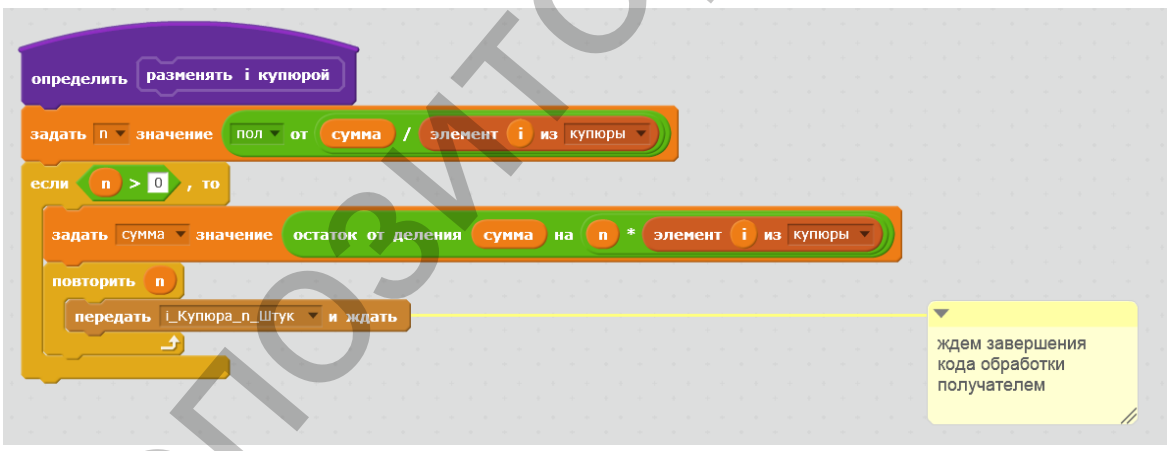

Рис. 62. Процедура размена *i*-ой купюрой из списка

В процедуре размена (рис. 62) для вычисления *n* используется встроенная функция ***пол*** (неудачный русский перевод англ. ***floor***), которая возвращает наибольшее целое, не превосходящее аргумент. Если *i*-я купюра нужна для размена ($n>0$), то передается сообщение для создания необходимого количества её изображений и вычисляется оставшаяся для размена сумма. Сообщение получает спрайт с костюмами – номиналами купюр. Скрипт обработки сообщения приведен на рис. 63.

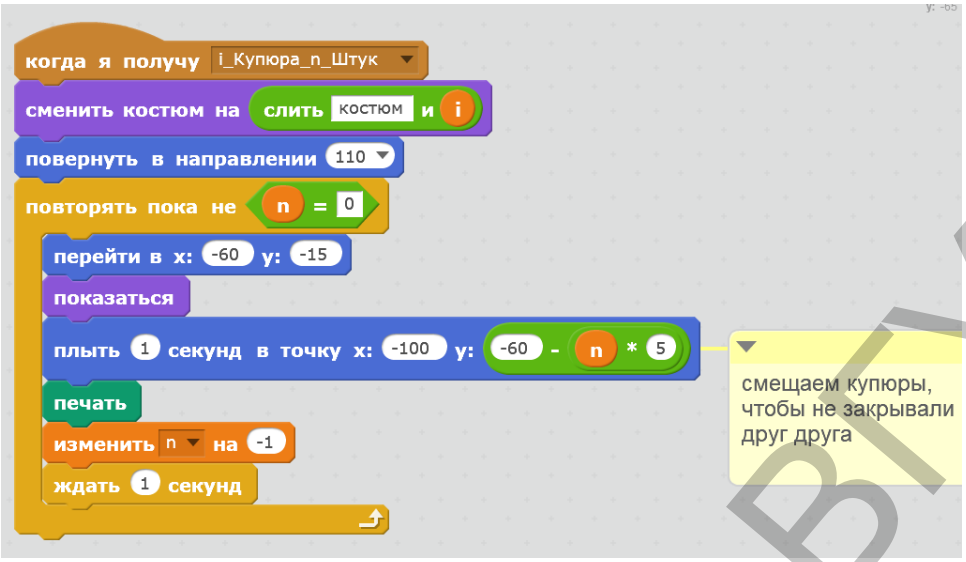

Рис. 63. Обработчик сообщения для выдачи купюр

Костюмы спрайта соответствуют номерам купюр в списке номиналов. Это позволяет использовать в имени костюма порядковый номер *i* (используется функция соединения строк ***слить***). Поворот и перемещение спрайта подобраны таким образом, чтобы печаталась пачка из *n* купюр.
Основной скрипт спрайта Банкомат приведен на рис. 64.

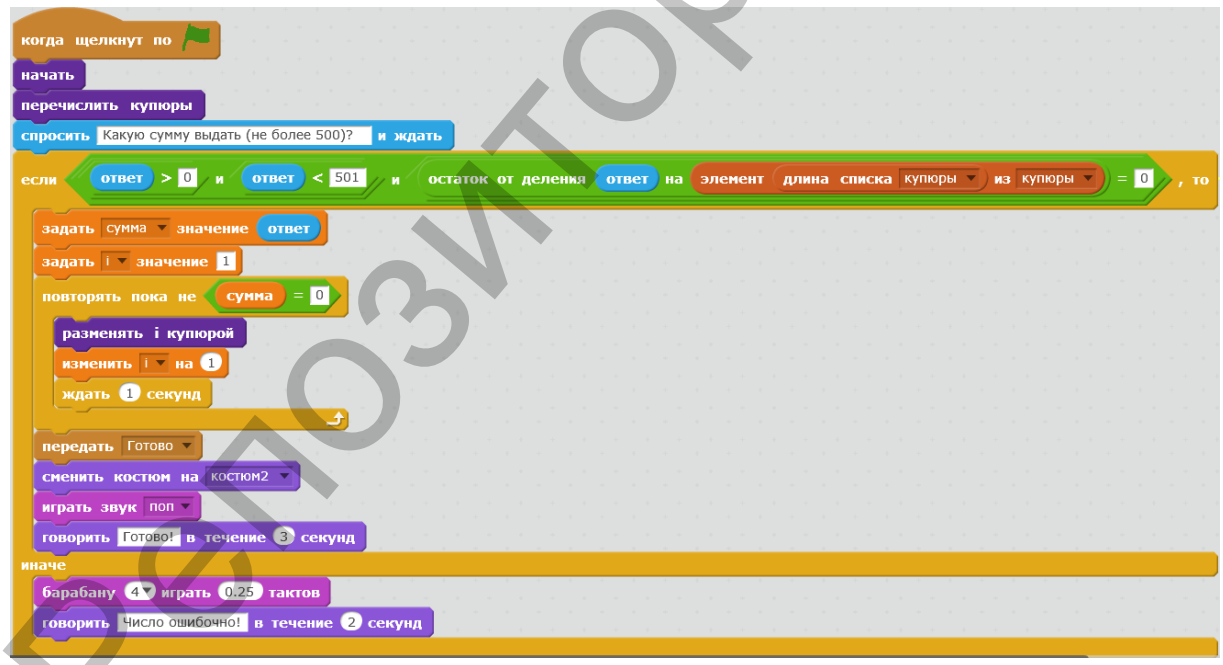

Рис. 64. Основной скрипт спрайта Банкомат

Конструкция если-то-иначе позволяет проверить правильность введенных пользователем данных. Сообщение ***Готово*** передается для того, чтобы спрятать спрайт с купюрами и сменить костюм у банкомата.
ЗАДАНИЕ. Разработайте процедуры ***начать*** и ***перечислить купюры*** (для информирования пользователя об имеющихся номиналах), а также обработчики сообщения ***Готово***.

ЗАДАЧА 14. «Звонки»

Рис. 65. Результат выполнения программы "Звонки"

Несколько друзей обмениваются звонками случайным образом (звонки самому себе не допустимы). Каждый звонок сопровождается изображением телефона и надписью о том, кто кому звонит. Вызываемый абонент говорит: «Алло!». Новый звонок может происходить до завершения предыдущего. Звонки прекращаются при нажатии клавиши «пробел».

РЕШЕНИЕ.

На сцене размещаются спрайты друзей, их имена сохраняются в списке (рис. 66).

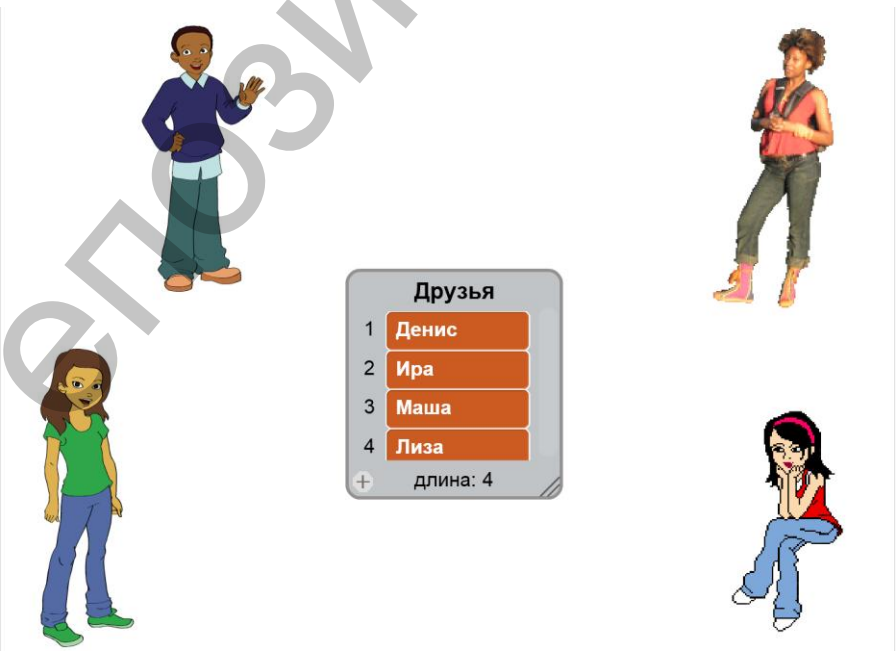

Рис. 66. Спрайты друзей и их имена в списке

Переменная *n* для всех спрайтов соответствует порядковому номеру друга в списке. Для спрайта Телефон используется изображение из Интернета с двумя костюмами: Off и On (звонок). Основной скрипт спрайта (рис. 67) обеспечивает звонок одного друга другому из списка через определенное время (1 сек).

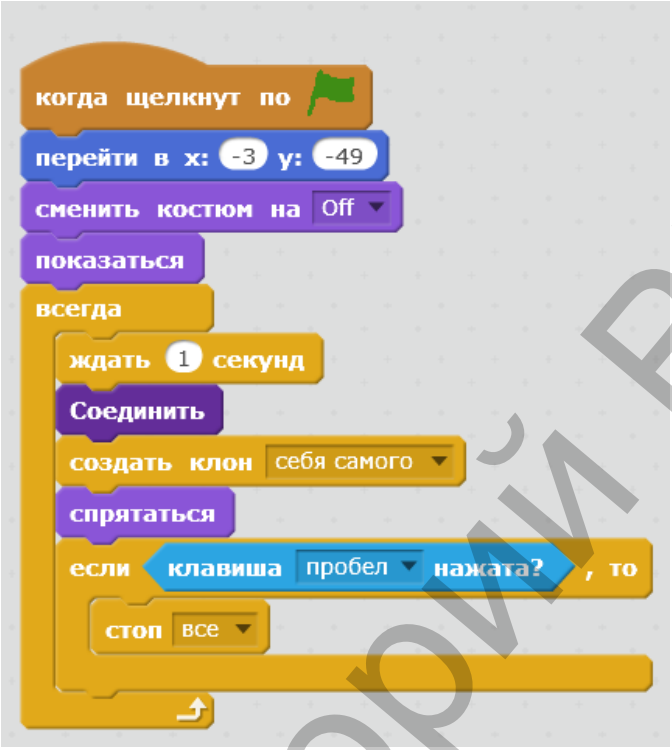

Рис. 67. Основной скрипт спрайта Телефон

До описания процедуры ***Соединить*** (рис. 68) создаются две переменные только для этого спрайта: *nBx* (номер вызываемого) *nИсх* (номер звонящего). Область видимости этих переменных позволит каждому клону иметь свои значения, не зависящие от оригинала и других клонов. Для подбора случайным образом не совпадающих номеров собеседников (звонок самому себе исключен по условию) используется цикл пока не. В условии выполнения цикла – двойное отрицание, т.е. тело цикла выполняется до тех пор, пока номера совпадают.

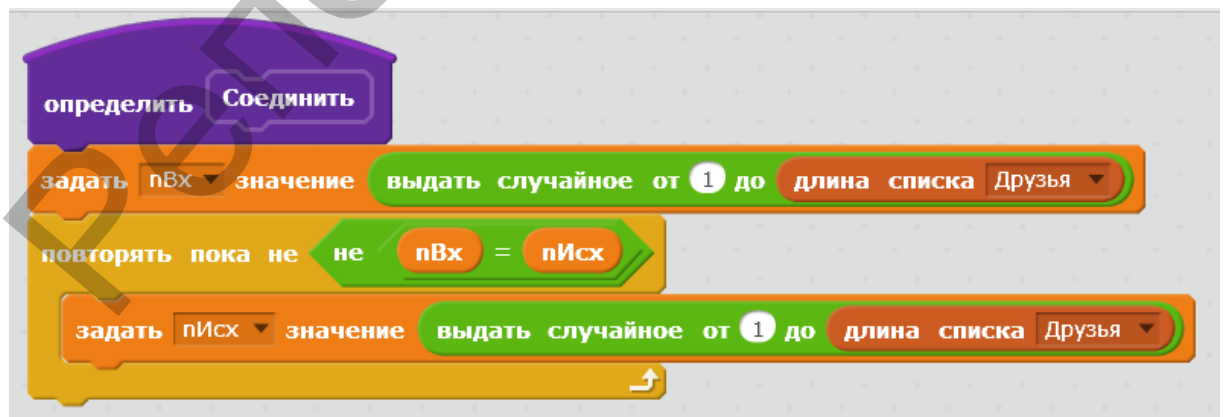

Рис. 68. Скрипт процедуры ***Соединить***

Все изображения звонков на сцене – клоны (оригинал спрятан). Скрипт клона приведен на рис. 69.

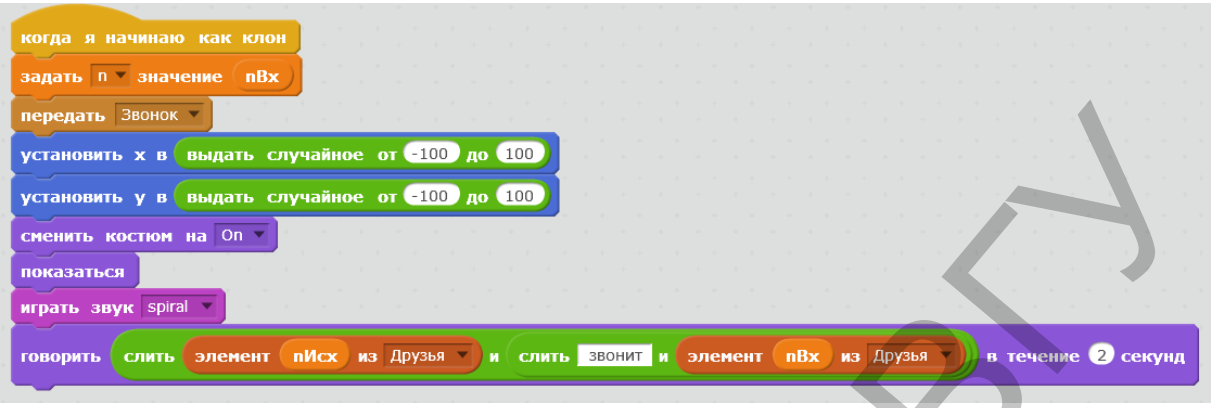

Рис. 69. Скрипт клона Телефон

Случайное размещение телефона на сцене позволяет отобразить несколько соединений одновременно. Для того, чтобы вызываемый абонент ответил "Алло!" через переменную *n* передаем другим спрайтам номер входящего, т.к. переменная *nBx* им не видна. Взаимодействие осуществляется посредством передачи сообщения ***Звонок***. Чтобы отобразить фразу кто кому звонит используется имена в списке Друзья и используется функция слияния строк.

ЗАДАНИЕ. Разработайте обработчики сообщения ***Звонок***.

ЗАДАЧА 15. «АБВ»

Рис. 70. Диалог с пользователем в программе "АБВ"

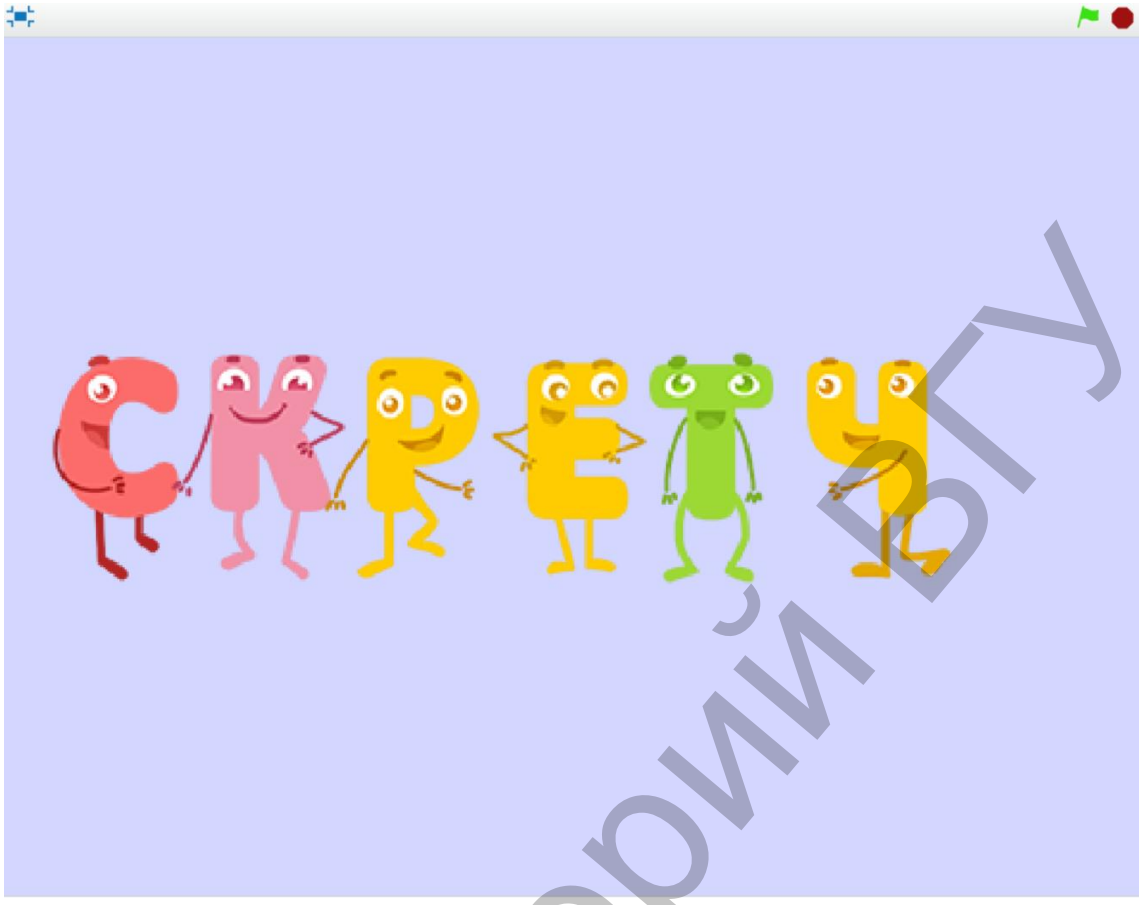

Рис. 71. Бегущая строка в программе "АБВ"

Введенный с клавиатуры пользователем текст отображается в виде бегущей по сцене строки. Для отображения символов используются костюмы (русские буквы и пробел, иные символы отображаются пробелом) из коллекции образовательного проекта [3]. Если во введенном тексте содержится слово ***скретч***, то после бегущей строки на сцене появляется поздравление с днем рождения Скретч.

РЕШЕНИЕ.

Для создания спрайта Алфавит загружаются костюмы (33 буквы русского алфавита и пробел) в алфавитном порядке. Для отображения букв используются клоны спрайта с соответствующими костюмами. Основной алгоритм реализован в скрипте спрайта Кота (рис. 72).

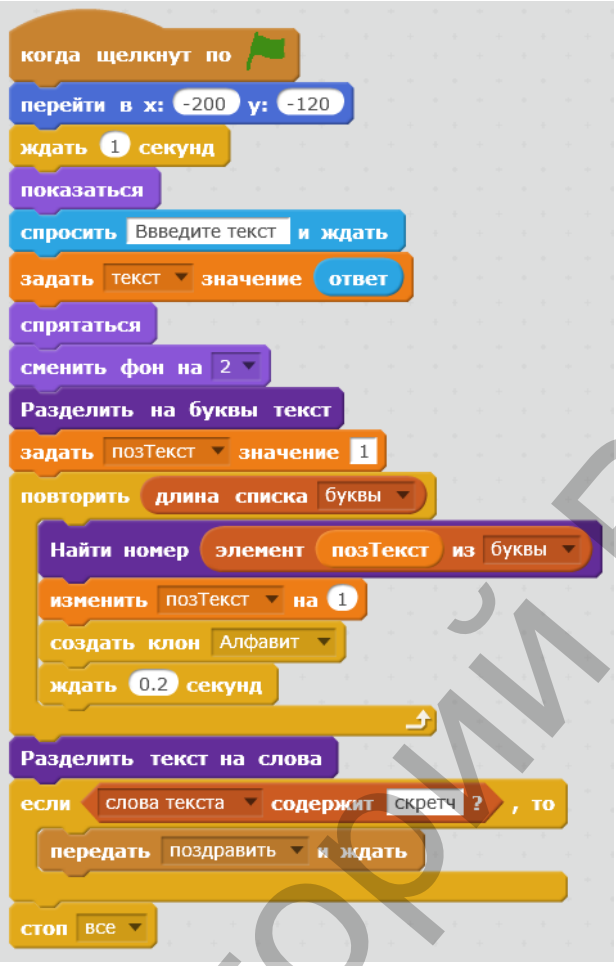

Рис. 72. Скрипт спрайта Кота

Для решения вводится несколько переменных: *текст* – для хранения заданной пользователем строки, *позТекст* – номер текущей буквы в тексте. Для поиска буквы в алфавите составляется список (*буквы*), содержащий буквы введенного пользователем текста, а также список *алфавит* с буквами в алфавитном порядке и пробелом. В таком случае номер костюма спрайта Алфавит будет совпадать с номером буквы в списке *алфавит* и сохраняться в переменной *номер*.

В отдельных процедурах реализуется разделение текста на буквы и поиск позиции буквы в алфавите. Разделение текста на слова необходимо для реализации второй части задания – проверки, есть ли в тексте слово "скретч".

Скрипты процедур приведены на рис. 73, 74.

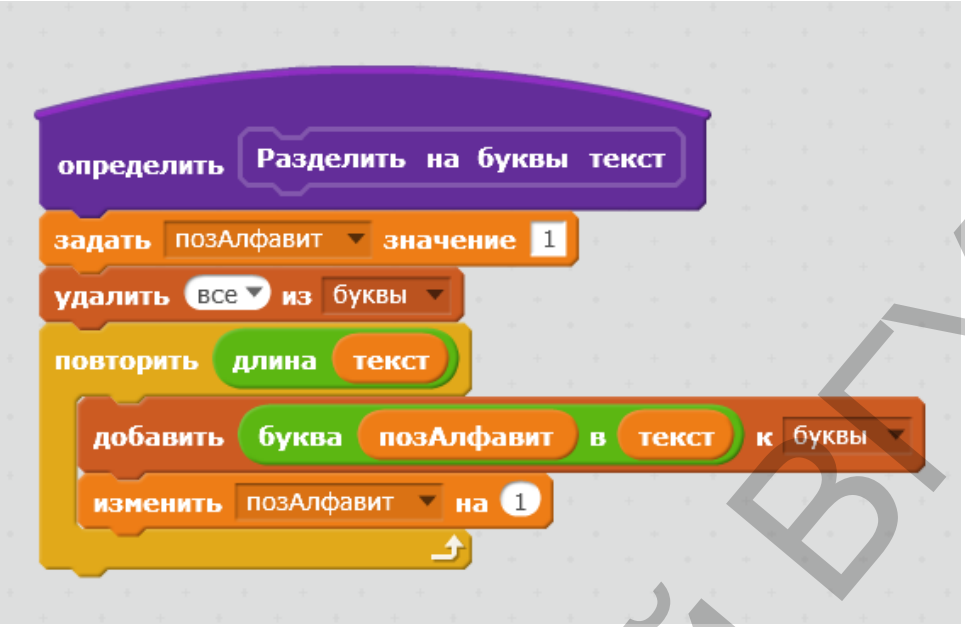

Рис. 73. Скрипт процедуры ***Разделить на буквы текст***

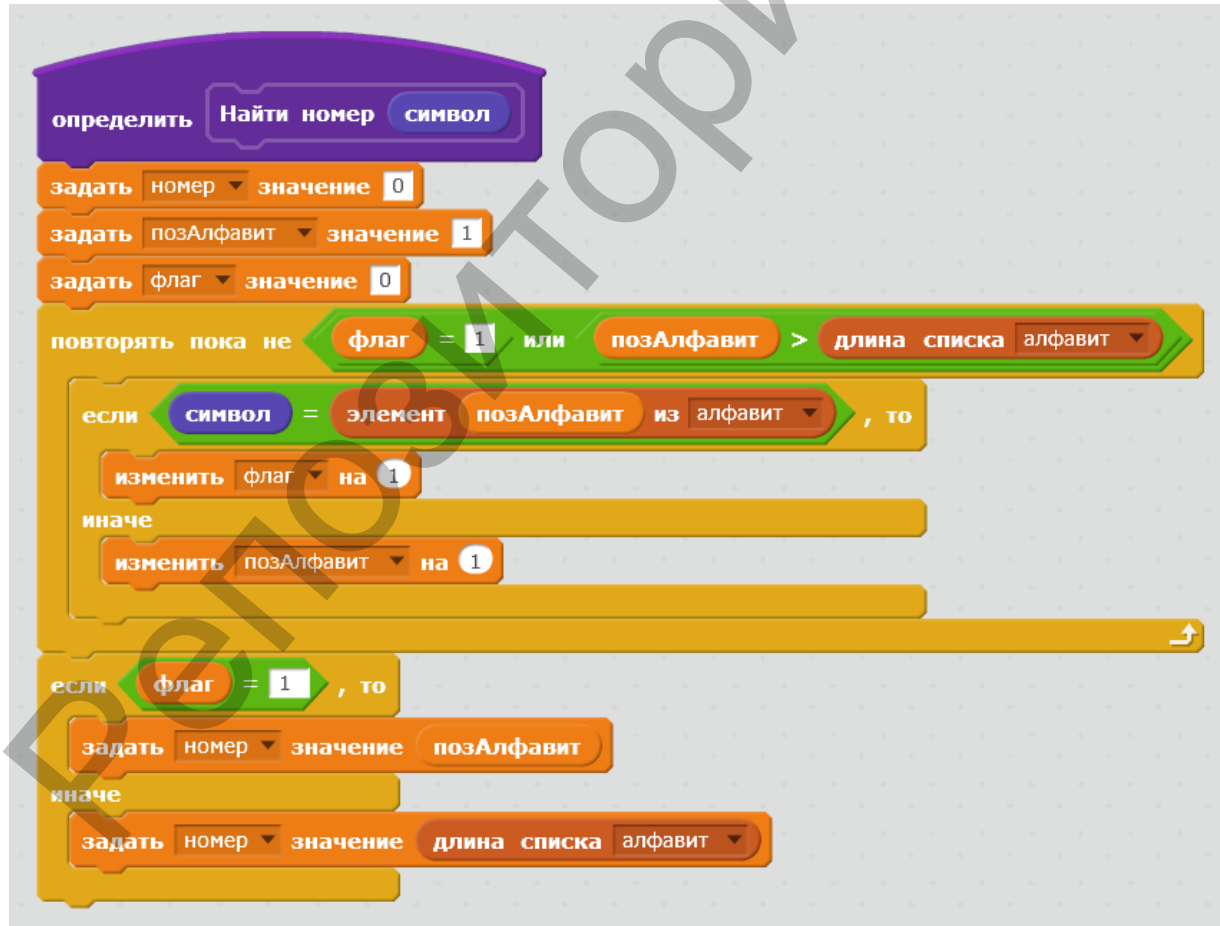

Рис. 74. Скрипт процедуры ***Найти номер***

Переменная *флаг* используется для выхода из цикла, когда нужная буква найдена. Если пользователь вводит символы, отличные от русских букв, то значение переменной *флаг* останется равным 0, а *номер* буквы станет равным длине списка алфавит (позиция пробела). Таким образом, такие символы заменяются пробелом.

На рис. 73 приведится скрипт, который позволяет разделить текст на слова. Словами считаем любую последовательность символов, которая заканчивается пробелом. Для однообразия добавляем к заданной пользователем строке пробел в конце, так как маловероятно, что это сделает вводящий текст пользователь. В начале цикла подстрока пустая (необходимо убедиться, что в рамке нет пробела). Далее склеиваются символы из текста в подстроку, пока не встречается пробел. *Подстрока* добавляется к списку слов, а увеличение переменной *позТекст* позволяет пропустить пробел и перейти к следующему слову. Этот алгоритм рассчитан на текст, в котором слова разделены ровно одним пробелом.

Рис. 75. Скрипт процедуры ***Разделить текст на слова***

Для отображения найденного символа используем скрипт спрайта Алфавит (рис. 76).

Рис. 76. Скрипт спрайта Алфавит

Переменная *номерКлона* создана только для этого спрайта, это позволяет одновременно отображать клоны с разным значением номера и соответственно костюма.

ЗАДАНИЕ 1. Разработайте обработчики сообщения ***поздравить*** для спрайтов и сцены.

ЗАДАНИЕ 2. Преобразуйте алгоритм решения задачи таким образом, чтобы не использовать списки.

ЗАДАНИЕ 3. Модифицируйте скрипт для текста, в котором слова могут быть разделены несколькими пробелами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Маржи, М. Scratch для детей. Самоучитель по программированию. – М.: Манн, Иванов и Фербер, 2017. – 288 с.
2. Официальный сайт проекта Scratch. – Режим доступа: https://scratch.mit.edu/. – Дата доступа: 20.08.2018.
3. Сообщество Scratch в Республике Беларусь. – Режим доступа: https://scratch.by. – Дата доступа: 20.08.2018.

Учебное издание

АЛЕЙНИКОВА Татьяна Григорьевна
ОГАНДЖАНЯН Ольга Петровна

ЗАДАЧНИК ПО ПРОГРАММИРОВАНИЮ В SCRATCH

Технический редактор *Г.В. Разбоева*
Компьютерный дизайн *Л.Р. Жигунова*

Подписано в печать 01.10.2018. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 2,56. Уч.-изд. л. 2,21. Тираж 50 экз. Заказ 128.

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя,
изготовителя, распространителя печатных изданий
№ 1/255 от 31.03.2014 г.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.