

Концепции программирования, поддерживаемые Scratch

В процессе создания с помощью Scratch интерактивных историй, игр и анимаций, начинающий может изучить основные методы и концепции, а также получить навыки программирования приложений.

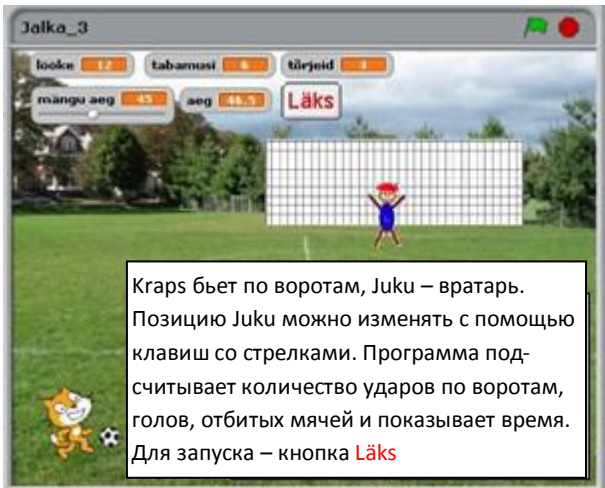

Необходимые знания и умения разрешения проблем и дизайна проектов


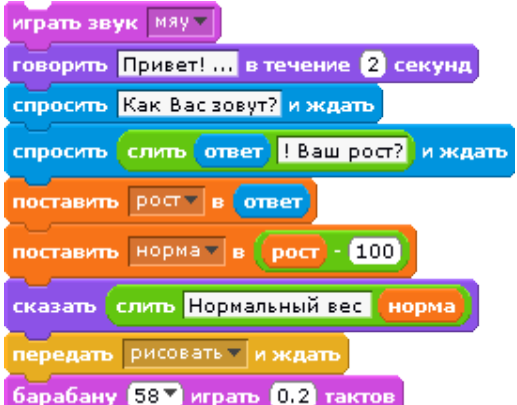
- логическое и алгоритмическое мышление;
- системный подход к решению проблем;
- развитие идей, начиная от исходной концепции и до конечного результата создания проекта;
- навыки и опыт создания интерфейса пользователя;
- умение отладки и тестирования результата;
- развитие настойчивости и умения концентрироваться.

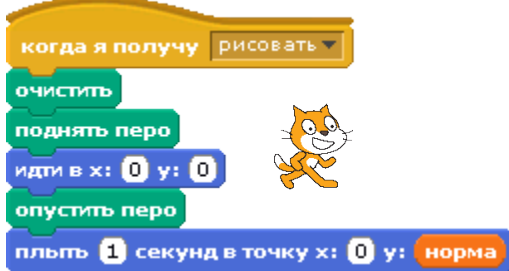

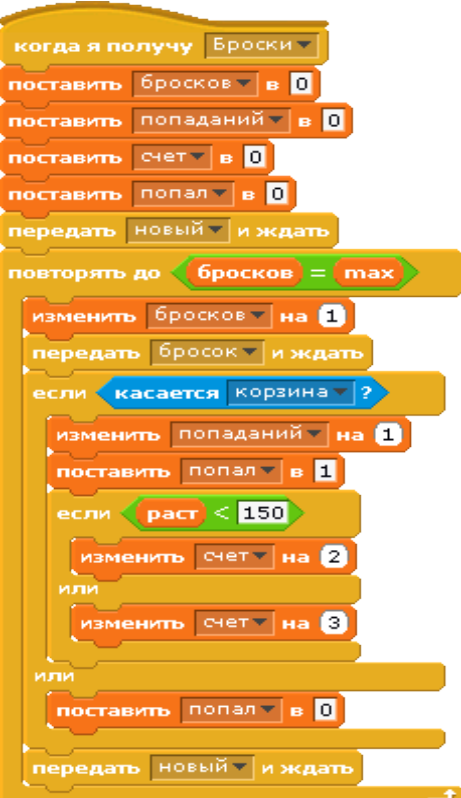
Фундаментальные идеи о компьютерах и программировании

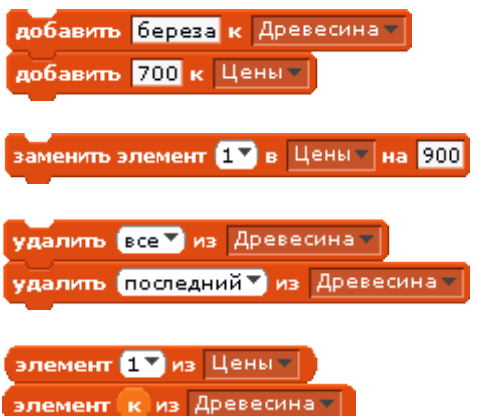
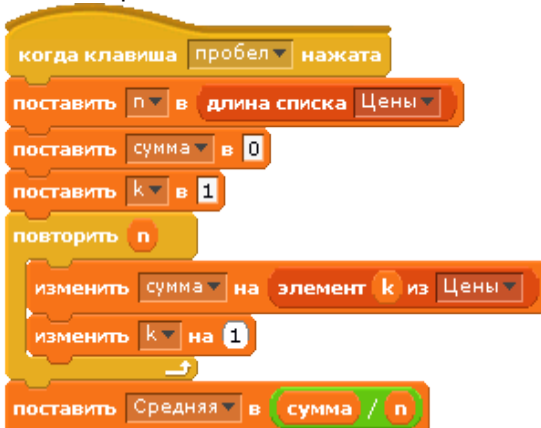
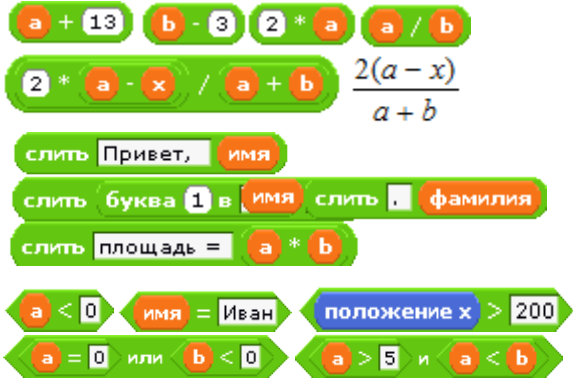
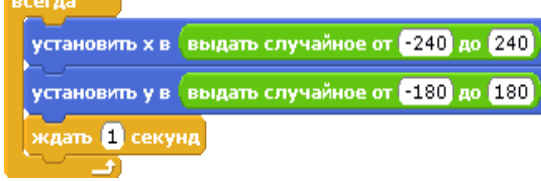
- программа однозначно задает компьютеру его действия шаг за шагом;
- составление программ требует не специальных знаний, а тщательной и ясной продуманности


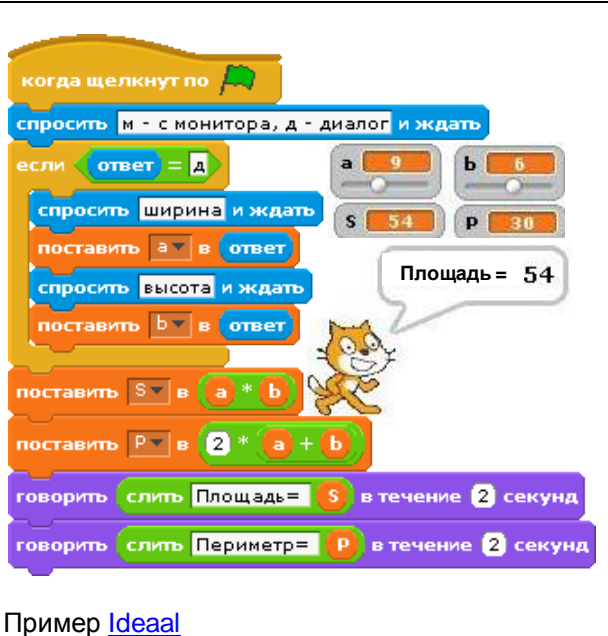
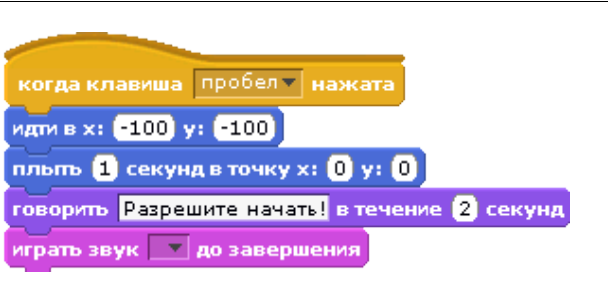
Основные концепции и понятия создания приложений и программирования


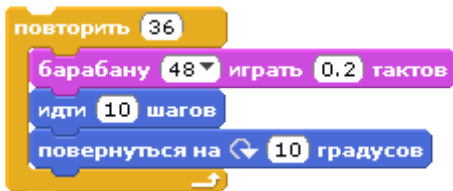
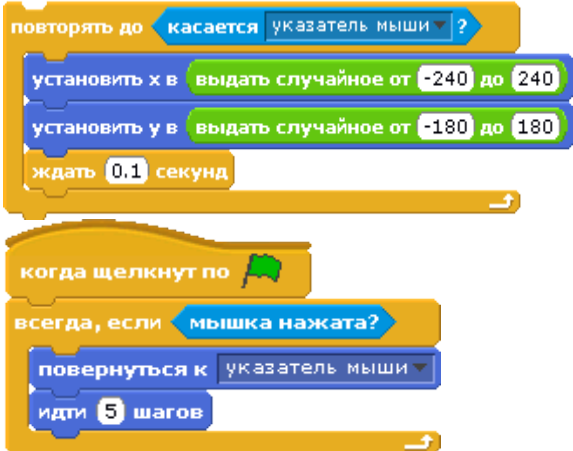
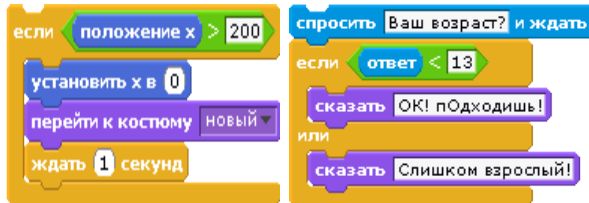
Концепт	Пояснения	Пример
Интерфейс пользователя	Независимо от системы и языка программирования в процесс создания приложения входит дизайн и реализация интерфейса пользователя. Интерфейс пользователя содержит средства, с помощью которых пользователь может общаться с программой: задавать требуемые команды и видеть результаты их выполнения, изменять исходные данные и т.п. В Scratch интерфейс создается на сцене . Его элементами могут быть различные фоны, активные и пассивные спрайты, командные кнопки, клавиши, мониторы переменных и т.п. В других системах для этого обычно используются формы и диалоговые окна.	 <p>Пример</p>
Программа. Команды и блоки. Программные единицы: процедуры и скрипты	Программа – это последовательность команд (операторов), которая определяет, какие действия должен выполнять компьютер с данными и объектами и обеспечивает работу интерфейса . В каждом языке имеется ограниченный набор команд, для представления которых заданы определенные правила. В Scratch команды (операторы) представляют собой графические блоки , разделенные на группы: Движение , Управление и т.д. Блок однозначно определяет синтаксис и возникновение синтаксических ошибок практически невозможно.	 <p>Пример</p>

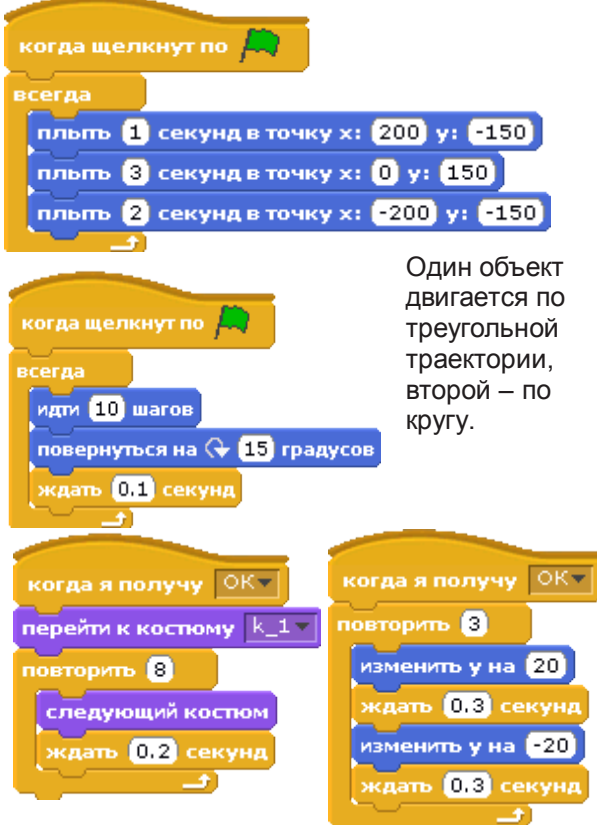

	<p>Программа может содержать несколько единиц разного типа: процедуры, функции и т.п.</p> <p>В Scratch программные единицы называются скриптами. Каждый скрипт связан с одним спрайтом и определяет его действия. У одного спрайта может быть несколько скриптов. Скрипт может запустить (обратиться, вызвать) другие скрипты, принадлежащие тому же или другим спрайтам.</p> <p>Для обращения используются блоки передать сообщение и ждать или передать сообщение</p>	<p>Выше приведены два основных скрипта из четырех, используемых в примере.</p> <p>Приведенные два скрипта связаны со спрайтом Krapc. Juku и Piri каждый связан с одним скриптом, аналогичным второму скрипту для Krapc.</p> <p>Щелчок по зеленому флажку запускает первый скрипт для Krapc. После приветствия данный скрипт с помощью команды передать Старт и ждать разом запускает скрипты для Juku и Piri. Эти скрипты начинаются с блоков Когда я получу Старт и выполняются до конца. Дождавшись окончания работы скриптов Juku и Piri, скрипт Krapc продолжает выполнение следующих команд. Krapc передвигается в центр сцены и запускается второй скрипт для этого спрайта. После выполнения последнего продолжается выполнение первого скрипта, которое передвигает Krapc в заданную точку сцены.</p>
<p>Объекты (спрайты). Свойства объектов, методы и события</p>	<p>Центральное место в Scratch занимают графические объекты, называемые спрайтами (sprite) и их костюмы. Объектом также является сцена и ее фоны.</p> <p>Хотя Scratch формально не является объектно-ориентированной системой, зачастую целесообразно рассматривать ее с помощью объектно-ориентированного подхода.</p> <p>С каждым объектом связан определенный набор свойств: имя, размер, позиция на сцене (x-y), цвет и т.д., а также методы, с помощью которых задаются действия с объектами данного типа: изменение позиции, размера и цвета, поворот и т.д. Блоки команд по существу соответствуют методам.</p> <p>Объект может реагировать на события: щелчок по клавише мыши, нажатие на заданную клавишу, соприкосновение с другим объектом.</p>	<p>Щелчок по спрайту Krapc запускает программу (реакция на событие). Пример</p>  <p>Блоки скрипта изменяют разные свойства объекта (спрайта): направление, позицию, размер, цвет, завихрение.</p>
<p>Виды данных</p>	<p>В Scratch можно использовать символные, графические и аудиоданные.</p> <p>Символьные данные: числа и тексты или строки. Их можно использовать во многих блоках (командах) как постоянные, переменные и элементы списков. Значения можно находить (вычислять) с помощью выражений и функций.</p> <p>Графические данные могут быть представлены двумя вариантами: спрайты и фоны сцены – импортируются или создаются с помощью редактора рисования. С</p>	

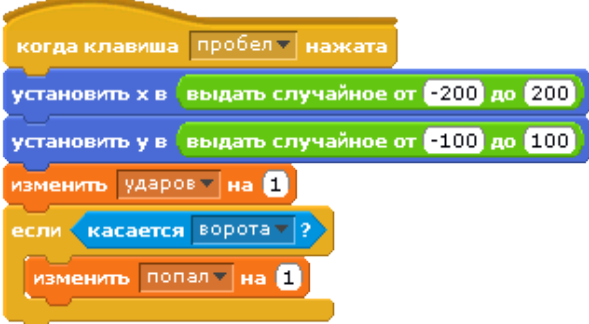
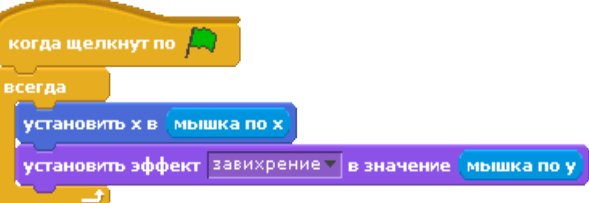

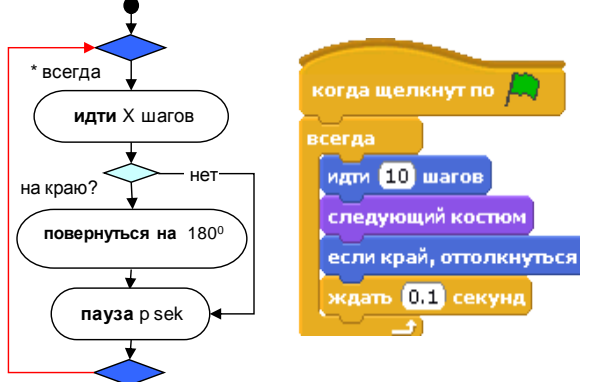
	<p>помощью команд можно задавать с ними различные действия. Рисунки, созданные с помощью команд Пера. Аудиоданные. Используются различные средства для создания звуков (блоки играть звук..., барабану играть... и т.п.), импортировать и записывать клипы: музыка, речь и т.п.</p>	<p>Скрипты демонстрируют использование данных разного вида. Символьные данные (числа и строки) используются как константы и переменные. Пример</p>  <p>В примере графические данные представляют собой спрайт Krapc и линия, рисуемая пером при движении спрайта. Звуковые данные создаются с помощью команд играть звук... и барабану играть...</p>
<p>Организация данных</p>	<p>С точки зрения организации данных во всех языках программирования различаются: скалярные данные: константы и переменные структурные данные: таблицы, массивы, списки и т.п. Значения констант задаются непосредственно в программе. В приведенном выше примере скрипта используется несколько числовых и текстовых констант. Это конкретные числа: 2, 100, 58, 0.2, 0 и тексты: "Привет! Я Krapc!", "Как Вас зовут?" и т.п., расположенные в полях блоков. Переменные представляются в программе с помощью имен, их значения в программе не видны. Обычно значения появляются в ходе выполнения программы. Например, значение переменной <i>рост</i> вводится командой спросить, значение переменной <i>норма</i> вычисляется по формуле $рост - 100$. Таблицы, массивы, списки представляют собой наборы данных с определенной структурой. В Scratch используются только списки – частный случай массивов</p>	
<p>Переменные и команда присваивания</p>	<p>Переменная – это ячейка памяти, в которую программа может записывать значения: числа и строки и использовать (считывать) их позднее для вычисления новых значений. В Scratch переменные можно создавать и использовать с помощью блоков в группе Переменные. При создании переменной задается имя, которое используется в командах для ссылки на ее текущее значение. Кроме того, при создании задается, является ли переменная доступной всем спрайтам (глобальная переменная) или только одному конкретному спрайту (локальная переменная). Для присваивания и изменений значений переменных используются блоки поставить переменная = выражение и изменить переменная на выражение. Значение переменной можно отобразить на сцене с помощью монитора. Значение можно изменять и вручную с помощью  слайдера (рычажка).</p>	<p>Программа имитирует броски в баскетболе</p>  <p>Примеры: Korvpall и Ideaal</p>

<p>Списки (массивы)</p>	<p>Список в Scratch более менее соответствует используемому в других языках одномерному динамическому массиву. Подобный массив представляет собой упорядоченный набор ячеек памяти (элементов). На элементы можно ссылаться с помощью имени массива и индексов. В Scratch можно создавать и использовать списки с помощью блоков группы Переменные. С помощью блоков этой группы можно добавлять элементы в конец списка или в середину, заменять, удалять и ссылаться на элементы.</p> 	 <p>Скрипт для нахождения средней цены компьютеров</p>  <p>Пример</p>
<p>Выражения, операции и функции</p>	<p>С помощью выражения можно задать правило вычисления значения. Для создания выражений в Scratch используются блоки группы Операторы. Выражение состоит из операндов и операций. Операндами могут быть: константы, переменные, функции, элементы списков. В зависимости от операций выражения можно разделить на следующие группы:</p> <ul style="list-style-type: none"> числовые выражения: +, -, *, / текстовые выражения: слить, буква в (выделить) сравнения: <, =, > логические выражения: и, или, не <p>В выражениях можно использовать различные функции: abs (абсолютная величина), sqrt (корень квадратный), sin, cos, asin, log, ln и т.д. Некоторые функции представлены отдельными блоками: mod (остаток от деления двух чисел), округлить и выдать случайное...</p> <p>NB! Для задания порядка выполнения операций в выражении в Scratch нельзя использовать скобки.</p>	 <p>поставить c в $a + 13 / \log b$</p> <p>поставить c в $\frac{2(a-x)}{a+b}$</p> <p>поставить c в $\frac{2(a-x)}{a+b}$</p> <p>поставить c в $c = \frac{a+13}{\log b }$ с округляется до двух позиций после запятой</p> <p>всегда</p>  <p>Пример Ideaal</p>

<p>Рисование, черчение</p>	<p>В большинстве языков программирования имеются средства, которые дают возможность создавать рисунки во время выполнения программы. В Scratch основные средства для этого расположены в группах Перо и Движение.</p> <p>Скрипт ниже рисует прямоугольный треугольник, длины катетов a и b которого задаются с помощью случайных чисел. Прямой угол помещается в центр сцены в точку (0,0). Переменная m задает масштаб: количество точек в одной единице длины.</p> 	
<p>Ввод и вывод символьных данных</p>	<p>При решении задач часто необходимо задать программе исходные данные и отобразить результаты. В первом случае говорят о вводе (считывании) данных, во втором – о выводе данных (печати, отображении).</p> <p>В Scratch для ввода данных можно использовать мониторы данных с рычажком (только для чисел) и блок Спросить, который дает возможность вводить числа и тексты в режиме диалога. Результаты переменных и/или списков можно выводить с помощью мониторов или блока Говорить</p>	
<p>Управление процессами</p>	<p>Очень важным при решении задач является задание порядка выполнения действий. Именно в этом заключается искусство программирования. В общем случае требуемые действия и порядок их выполнения не зависят от используемого языка, а зависят от сути задачи или алгоритма ее решения.</p> <p>Можно выделить четыре типа процессов: последовательность или последовательный процесс, повторение или циклический процесс, выбор или разветвляющийся процесс и параллельный процесс.</p> <p>В языках программирования имеются средства для описания процессов различного типа.</p>	<p>Пример Ideaal</p>
<p>Последовательность или последовательный процесс</p>	<p>При создании программы следует учитывать, что действия, задаваемые блоками, выполняются в определенном порядке. Простейшим случаем является последовательный процесс, при котором блоки выполняются подряд сверху вниз.</p>	

<p>Повторение или циклический процесс</p>	<p>Для описания повторений в Scratch, как и в других языках программирования, имеется несколько блоков (операторов): Всегда, Повторить n раз, Повторять до условия, Всегда если условие. Блоки, расположенные внутри блока Всегда, практически выполняются бесконечно. Работу скрипта можно прервать с помощью красной кнопки. Внутри блока повторения может быть один или несколько блоков Если условие, которое при выполнении условия с помощью соответствующих блоков прерывает работу скрипта (блок Остановить скрипт) или работу всей программы (Остановить все). При выполнении команды повторить n внутренние блоки выполняются n раз. Количество повторений – число n может быть задано константой, переменной или выражением</p> <p>При выполнении команды Повторять до условия внутренние блоки повторяются (выполняются) до тех пор, пока условие станет истинным..</p> <p>При выполнении команды всегда если условие внутренние блоки повторяются (выполняются) до тех пор, пока условие истинно.</p>	 <p>Первый блок задает бесконечное движение спрайта вперед и назад по сцене. Прервать это можно с помощью красной кнопки. При выполнении второго блока спрайт движется слева направо. Движение (повторение) заканчивается, когда значение координаты x станет больше 200.</p>  <p>Объект (например, кот) "барабанит" и делает круг по сцене (360°). Пример</p> 
<p>Выбор или разветвляющийся процесс</p>	<p>С помощью блоков если условие и если условие или можно описывать разветвляющиеся процессы. В операторе если ... (выбор из одного), если условие истинно, выполняются внутренние блоки, в противном случае они пропускаются. В операторе если условие или (выбор из двух), если условие истинно, то выполняются блоки первой ветви, в противном случае – блоки второй ветви</p>	 <p>В первом блоке проверяется условие, является ли значение координаты x больше 200. Если да, выполняются внутренние блоки. Фрагмент второго скрипта отображает сообщение, зависящее от введенного возраста. Пример Ideaal</p>

<p>Параллельные процессы</p>	<p>Два или несколько скриптов (процессов) можно выполнять одновременно, т.е. параллельно. Параллельное выполнение можно задавать несколькими способами. Например все скрипты, первый блок в которых это блок с зеленым флажком, запускаются одновременно и выполняются параллельно, если щелкнуть по зеленому флажку. Параллельно выполняются также скрипты, начинающиеся блоком Когда я получу сообщение, в которых принимается одинаковое сообщение.</p>	 <p>Один объект движется по треугольной траектории, второй – по кругу.</p> <p>Когда приходит сообщение "OK", один объект делает три прыжка, второй – 8 раз меняет костюм, например танцует. Проект Tantsud</p>
<p>События</p>	<p>Объекты могут реагировать на определенные события: нажатие на какую-либо клавишу, щелчок мышью по объекту, прикосновение к другому объекту или краю сцены и т.п. В скриптах можно предусмотреть реакции на определенные события. Основным средством для обработки события являются так называемые блоки заголовков: Когда клавиша ... нажата и Когда щелкнут по спрайту. Внутри скриптов часто используется блок касается {спрайт край курсор}? Если данный спрайт касается другого спрайта, края сцены или указателя мыши (курсора), то блок возвращает значение истина. В принципе нажатие на зеленый флажок и запуск скриптов с помощью блоков передать и когда я получу сообщение также являются событиями.</p>	 <p>Крaps поворачивается и движется направо</p> <p>Крaps поворачивается и движется налево</p> <p>При щелчке по зеленому флажку переменным присваивается значение 0 и мяч помещается в исходное положение.</p>

		 <p>При нажатии на клавишу пробел с помощью случайных чисел изменяется положение мяча и увеличивается на единицу количество ударов. Если мяч касается ворот, то значение переменной <i>попал</i> увеличивается на 1.</p>
<p>Взаимодействие</p>	<p>На объекты можно повлиять в реальном времени, передвигая мышь, с помощью звука и т.п., используя блоки мышка по x, мышка по y, громкость и другие блоки группы Сенсоры.</p>	 <p>После щелчка по зеленому флажку, объект двигается вместе с курсором мыши в горизонтальном направлении и изменяет размер завихрения в соответствии со значением <i>y</i>.</p>
<p>Взаимодействие между спрайтами и скриптами</p>	<p>Если программа состоит из нескольких скриптов, то часто возникает необходимость в координировании и синхронизации их работы. Один спрайт может обратиться к другим, те в свою очередь к следующим и т.д. Для организации работы скриптов используются блоки:</p> <ul style="list-style-type: none"> передать сообщение; передать сообщение и ждать; когда я получу сообщение. <p>Представленная рядом программа состоит из четырех скриптов. Скрипт с зеленым флажком является главным. С него начинается работа программы и он запускает остальные. Если пользователь вводит букву "д", то запускается скрипт Читай, и после окончания им работы запускается скрипт Вычисли. Если введена отличная от д буква, то сразу запускается скрипт Вычисли.</p>	 <p>Не дожидаясь окончания работы скрипта Вычисли, запускается также скрипт Начерти (здесь не показан, аналогичен скрипту в пункте Черчение). Пример</p>
<p>Алгоритмизация</p>	<p>Алгоритм определяет, какие действия и в каком порядке следует выполнять для решения данной задачи или выполнения данной работы. Требуемые действия и порядок их выполнения в общем случае не зависят от конкретного языка, а зависят от самой задачи или модели. Действия в алгоритме представляются в более общем виде. В последнее время для представления алгоритмов часто используется язык моделирования UML.</p>	

	Скрипты Scratch можно рассматривать как один из способов представления алгоритмов.	Примеры: Jalka , Ruutvxrrand , Maks_Vek
Моделирование	<p>Средства Scratch можно использовать для объяснения и иллюстрации принципов объектно-ориентированного моделирования.</p> <p>Спрайты Scratch принадлежат одному универсальному классу – Спрайт. Спрайты имеют определенные наборы свойств: имя, позиция, размер,... и методов: идти(), повернуться(), изменитьX(), ...</p> <p>Последние представляются с помощью команд или блоков. Составленные из команд (основных методов) скрипты можно рассматривать как подклассы пользователя. Связанные со спрайтами костюмы, перо, переменные, списки также можно рассматривать как подклассы..</p> <p>Сцена рассматривается как основной класс, класс Спрайт является его подклассом: все спрайты располагаются на сцене.</p> <p>Класс Сцена можно рассматривать представителем системы или проекта.</p>	<pre> classDiagram class Фон { имя номер } class Аудиоклип { имя номер играть() } class Костюм { имя номер } class Перо { размер цвет вверх() вниз() очистить() } class Сцена_система { размеры цвет изменить_фон() изменить_цвет() } class Спрайт { имя позиция: X, Y направление размер цвет видимость идти() повернуться() поменять_костюм() изменить_цвет() сказать() играть_звук() } class Переменная { имя значение тип создать() удалить() поставить() изменить() } class Список { имя длина создать() удалить() добавить() изменить() } class Класс_пользователя { } Фон "1..*" -- "0..*" Сцена_система Аудиоклип "0..*" -- "0..*" Сцена_система Аудиоклип "0..*" -- "0..*" Спрайт Костюм "1..*" -- "0..*" Спрайт Перо "1" -- "0..*" Спрайт Сцена_система "0..*" -- "0..*" Спрайт Сцена_система "0..*" -- "0..*" Переменная Спрайт "0..*" -- "0..*" Переменная Спрайт "0..*" -- "0..*" Список Спрайт "0..*" -- "0..*" Класс_пользователя </pre> <p>Пример. Проект Tantsud</p>

Концепции программирования, которые в данный момент Scratch не поддерживает: функции; использование параметров и аргументов; рекурсия; определение своих классов объектов; запросы; обработка ошибок; ввод/вывод файлов.