**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. | Kraps бьет по воротам, Juku – вратарь. Позицию Juku можно изменять с помощью клавиш со стрелками. Программа под­считывает количество ударов по воротам, голов, отбитых мячей и показывает время. Для запуска – кнопка Läks  [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VJalka_K.html) |
| 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. | [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VProge_Skeem.html) |
|  | Программа может содержать несколь­ко единиц разного типа: **процедуры**, **функции** и т.п.  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.  Для обращения используются блоки  **передать *сообщение* и ждать** или **передать *сообщение*** | Выше приведены два основных скрипта из четырех, используемых в примере. Приведенные два скрипта связаны со спрай­том Kraps. Juku и Pipi каждый связан с одним скрип­том, аналогичным второму скрипту для Kraps.  Щелчок по зеленому флажку запускает первый скрипт для Kraps. После приветствия данный скрипт с помощью команды  **передать *Старт* и ждать** разом запускает скрипты для Juku и Pipi. Эти скрипты начинаются с блоков **Когда я получу *Старт*** и выполняются до конца. Дождавшись окончания работы скриптов Juku и Pipi, скрипт Krapsа продолжает выполнение следующих команд. Kraps передвигается в центр сцены и запускается второй скрипт для этого спрайта. После выполнения последнего продолжается выполнение первого скрипта, которое передвигает Krapsа в заданную точку сцены. |
| Объекты  (спрайты).  Свойства объектов,  методы и события | Центральное место в Scratch занимают **графические объекты**, называемые **спрайтами** (sprite) и их костюмы. Объектом также является **сцена** и ее фоны.  Хотя Scratch формально не является объектно-ориентированной системой, зачастую целесообразно рассматри­вать ее с помощью объектно-ориен­тированного подхода.  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.  Объект может реагировать на **события**: щелчок по клавише мыши, нажатие на заданную клавишу, соприкосновение с другим объектом.  . | Щелчок по спрайту Kraps запускает программу (реакция на событие). [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VObjektid.html)  **Vene_Scr_pilt3.gif**  Блоки скрипта изменяют разные свойства объекта (спрайта): направление, позицию, размер, цвет, завихрение. |
| Виды данных | В Scratch можно использовать сим­вольные, графические и аудио­данные.  **Символьные данные**: числа и тексты или стринги. Их можно использовать во многих блоках (командах) как постоянные, переменные и элементы списков. Значения можно находить (вычислять) с помощью выражений и функций.  **Графические данные** могут быть представлены двумя вариантами:  **Спрайты** и **фоны сцены** – импортируются или создаются с помощью редактора рисования. С 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. | **Pilt_vene9-indeks2.gif**  Скрипты демонстрируют использование данных разного вида. Символьные данные (числа и стринги) используются как константы и переменные. [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VAndmeliigid.html)    В примере графические данные представляют спрайт Kraps и линия, рисуемая пером при движении спрайта.  Звуковые данные создаются с помощью команд **играть звук...** и **барабану играть...** |
| Организация данных | 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 Kraps!”, ”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 *рост* вводится командой **спросить**, значение переменной *норма* вычисляется по формуле *рост – 100*.  Таблицы, массивы, списки представляют собой наборы данных с определенной структурой. В Scratch используются только **списки –** частный случай массивов | |
| Переменные и команда присваивания | Переменная – это ячейка памяти, в которую программа может записывать значения: числа и стринги 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)**. | Программа имитирует броски в баскетболе  Vene_pilt-korvpall1.gif  Примеры**:** [Korvpall](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VKorvpall.html) и [Ideaal](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VIdeaal.html) |
| Списки  (массивы) | 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 **Переменые**.  С помощью блоков этой группы можно добавлять элементы в конец списка или в середину, заменять, удалять и ссылаться на элементы.  **Vene_pilt10_loend.gif**  **Vene_pilt10_loend.gif**  **Vene_pilt10_loend.gif**  **Vene_pilt10_loend.gif** | Скрипт для нахождения средней цены компьютеров  **Vene_loendid_skript.gif**  [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VLoendid.html) |
| Выражения,  операции и  функции | 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:   * числовые выражения: +, -, \*, / * текстовые выражения: **слить**,  **буква в** (выделить) * сравнения: **<**, **=**, **>** * логические выражения: **и**, **или, не**   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…**  **NB!** Для задания порядка выполнения операций в выражении в Scratch нельзя использо­вать скобки. | **Vene_pilt11_muut.gif**  Vene_pilt11_muut.gifVene_pilt11_muut.gifVene_pilt11_muut.gif  Vene_pilt11_muut.gifVene_pilt11_muut.gif  **Vene_pilt11_muut.gif**    **Vene_pilt11_muut.gif**  Пример [Ideaal](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VIdeaal.html) |
| Рисование,  черчение | 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**.  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. | **Vene_pilt12-joonis.gif**  Демо [Rist\_Ring](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VJoonestaminel.html) |
| Ввод и вывод символьных данных | 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).  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** | Пример [Ideaal](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VIdeaal.html) |
| Управление процессами | 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.  Можно выделить четыре типа процессов: **последовательность** или последовательный процесс, **повторение** или циклический процесс, **выбор** или разветвляющийся процесс и **параллельный** процесс.  В языках программирования имеются средства для описания процессов различного типа. | |
| Последова­тельность или последова­тельный процесс | 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. | **Vene_pilt15.gif** |
| Повторение или циклический процесс | 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  При выполнении команды  **Повторять до *условия***  внутренние блоки повторяются (выполняются) до тех пор, пока условие станет истинным..  При выполнении команды  **всегда если *условие*** внутренние блоки повторяют­ся (выполняются) до тех пор, пока условие истинно. | 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.  Vene_pilt15.gif  Объект (например, кот) ”барабанит” и делает круг по сцене (360o). [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VKordused_1.html)  Vene_pilt14.gif  Vene_pilt14.gif |
| Выбор или разветвляю­щийся процесс | 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 | 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](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VIdeaal.html) |
| Параллель­ные процессы | 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.*** | Vene_pilt17.gif  Vene_pilt17.gifОдин объект двигается по треугольной траектории, второй – по кругу.    Когда приходит сообщение ”ОК” , один объект делает три прыжка, второй – 8 раз меняет костюм, например танцует. Проект [Tantsud](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VTantsud.html) |
| События | 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. | Vene_pilt19.gif  При щелчке по зеленому флажку переменным присваивается значение 0 и мяч помещается в исходное положение.  Vene_pilt19.gif  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 *попал* увеличивается на 1. |
| Взаимодейст­вие | 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**. | Vene_pilt20.gif  После щелчка по зеленому флажку, объект двигает­ся вместе с курсором мыши в горизонтальном направлении и изменяет размер завихрения в соответствии со значением y. |
| Взаимодейст­вие между спрайтами и скриптами | 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:  **передать** *сообщение;*  **передать** *сообщение* **и** **ждать**;  **когда я получу** *сообщение.*  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***. | Не дожидаясь окончания работы скрипта ***Вычисли***, запускается также скрипт ***Начерти***  (здесь не показан, аналогичен скрипту в пункте  Черчение). [Пример](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VKoostoo.html) |
| Алгоритмиза­ция | 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.  Скрипты Scratch можно рассматривать как один из способов представления алгоритмов. | Vene_pilt20.gif  Примеры: [Jalka](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VJalka_K.html), [Ruutvхrrand](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VRuutvorrand_0.html), [Maks\_Vek](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VMaks_Vektoris.html) |
| Моделирова­ние | Средства Scratch можно использовать для объяснения и иллюстрации принципов объектно-ориентированного моделирования.  Спрайты Scratch принадлежат одному универсальному **классу** – **Спрайт**. Спрайты имеют определенные наборы **свойств**: имя, позиция, размер,… и  **методов**: идти(), повернуться(),  изменитьX(), ...  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..  **Сцена** рассматривается как основной класс, класс **Спрайт** является его подклассом: все спрайты располага­ются на сцене.  Класс **Сцена** можно рассматривать представителем системы или проекта. | Пример. Проект [Tantsud](http://www.tud.ttu.ee/~vilip/Scratch/Vene_Opik/VTantsud.html) |

**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.