Альбрант Евгений Олегович
учитель информатики
МБОУ «СОШ №1»
г. Абакан, Республика Хакасия

МЕТОД ПРОЕКТОВ КАК СРЕДСТВО ПОВЫШЕНИЯ МОТИВАЦИИ ШКОЛЬНИКОВ К ИЗУЧЕНИЮ ПРОГРАММИРОВАНИЯ

***Аннотация**: в данной статье рассматривается проблема мотивации к изучению программирования у школьников, которая снижается по мере накопления большого теоретического материала. Автором обосновывается мысль о необходимости учителю создать благоприятные условия, повышающие заинтересованность учеников в изучении программирования. Высокоэффективным мотивирующим фактором может стать метод проектов, в процессе которого у школьников появляется понимание возможностей практического применения изучаемого материала по программированию.*

***Ключевые слова**: графика, анимация, программирование, проект.*

Одним из самых важных этапов информационного воспитания школьников является изучение основ программирования. И причиной тому является не столько возрастающая популярность профессии программиста или тенденции развития цивилизации, где когда-нибудь всё за человека будут делать машины, роботы, а человеку необходимо лишь уметь грамотно управлять этим процессом. Причиной для развития алгоритмического мышления является необходимость научить ребёнка структурировать знания, находить информацию, разбивать сложные задачи на более простые, выстраивать последовательную структуру своей работы. Это далеко не полный перечень умений, формируемых благодаря занятиям программированием.

Но понимание важности изучения данного раздела информатики порой доступно лишь учителю. Ученик же, в силу своей подчас низкой усидчивости, не-

желания запоминать сложные теоретические выкладки, не всегда способен воспринимать сложные алгоритмические конструкции, вникая в тонкости их использования в различных ситуациях.

Первое знакомство с программированием для школьника начинается задолго до запуска PascalABC или какой-то иной среды программирования. Сначала он услышит рассказ учителя об исполнителях, способных к выполнению алгоритмов – последовательностей действий, приводящих к достижению определённого результата. Затем попробует описать в словесной форме, что человеку, например, необходимо сделать, чтобы приготовить яичницу на завтрак. Формальный исполнитель – менее понятное выражение, но и оно рано или поздно поддаётся пониманию, когда мы вместе с учеником представим робота, выполняющего наши указания.

Первой возможностью попробовать силу программирования для школьника становится среда «КуМир», где появляется возможность на практике отработать умение «запрограммировать» робота на выполнение поставленной задачи. Каждый учитель видел горящие глаза ребёнка, когда ему удаётся-таки выполнить какой-нибудь сложный манёвр. Кто-то сделает это элегантно, в одно или два действия, а кто-то довольно неуклюже, так и не осознав всех плюсов использования циклов или условных операторов.

Начало изучения языка программирования Паскаль – всегда непростая пора для юных программистов. Огромное количество информации одних школьников пугает и отталкивает, заставляет думать, что программирование – не такая уж интересная область, как это казалось в младших классах. Другие, несмотря на своё усердие и достаточно хорошее понимание основ (структура программы, разработка линейных и разветвляющихся конструкций, использование переменных и оператора присваивания и др.), довольно скоро сдают позиции: особенно это становится заметно на темах «Циклы», «Массивы», «Процедуры и функции». И это, на мой взгляд, самая подходящая пора немного отойти от намеченного плана и подойти к программированию на языке Паскаль с другой стороны, более легкой и игровой – графике и анимации.

Школьникам предлагается с помощью среды программирования PascalABC что-нибудь нарисовать на экране. Конечно, необходимо подключить специальную библиотеку для работы с графикой и потратить некоторое время на изучение основных команд, необходимых для рисования хотя бы самых простых фигур: линии, прямоугольника, окружности, дуги. Ещё несколько вспомогательных команд (установка цвета пера или стиля закраски) – и мы готовы к воплощению своего первого шедевра. Здесь ученикам можно дать полную свободу действий и позволить нарисовать с помощью изученных команд всё, что у них получится за время, отведённое на уроке: от смайлика до автомобиля или домика в лесу.

Достаточно тяжелый труд – десятками команд отрисовать что-то очень простое. Но эти действия необходимы, ведь в дальнейшем нам придётся усложнить задачу: мы попробуем создать компьютерную анимацию.

Для начала необходимо преобразовать объект, который мы хотим привести в движение, таким образом, чтобы он имел какую-то опорную точку с координатами X и Y. Рассмотрим на примере рисования окружности. Заменим *circle(150,100,30)* на *circle(x,y,30)*, предварительно описав переменные x и y и присвоив им первоначальные значения:

```
X:=150; Y:=100;
```

Как же заставить закрашенную окружность (в задумке ребёнка – солнышко) перемещаться по экрану? Попробуем это проанализировать вместе с детьми.

Любая анимация, связанная с перемещением объекта, – это набор однотипных действий, который можно описать схематично следующим образом:

Задание начальных координат для объекта – Рисование объекта – Пауза – Стирание объекта – Изменение координат – снова Рисование объекта (и повторение всего набора действий необходимое количество раз)

Разберём каждый шаг по отдельности.

1. *Задать начальные координаты* положения объекта – самое простое. Ученику необходимо лишь ориентироваться в графическом экране, понимая, что точка (0,0) находится в левом верхнем углу экрана.

```
x:=150; y:=100;
```

2. *Рисование объекта*. Чтобы отрисовать необходимый объект, заданный координатами X,Y, необходимо выполнить, как минимум, две команды: установить цвет пера (или заливки) и выполнить рисование самого объекта. Если объект сложный, состоящий из множества линий, окружностей и других графических примитивов, то правильнее было бы вынести это действие в отдельную процедуру. Если же объект простой, то усложнять программу нет необходимости, а значит получаем примерно следующий набор строк:

```
SetBrushColor(clYellow); //установка цвета закраски объекта;
Circle(x,y,30);//рисуем жёлтое солнце (без лучей).
```

3. *Пауза* необходима для того, чтобы глаз успел разглядеть объект в данной позиции. Если пренебречь паузой, анимация может пройти для нас просто незаметно, потому что современные компьютеры способны в считанные доли секунды сотни раз нарисовать объект и сменить его положение на экране. Для создания задержки используется команда *sleep (n),* где n – количество миллисекунд, в течение которых программа не будет переходить к следующему шагу нашего алгоритма:

```
Sleep(100);
```

4. *Стирание объекта.* Это действие можно выполнить разными способами. Если на экране кроме описываемого объекта ничего нет, то можно воспользоваться процедурой стирания всего экрана – *clearwindow*. Но, скорей всего, мы создаём анимацию для части рисунка, а значит не хотим стирать экран полностью. Самый простой вариант стирания только требуемого объекта – «рисование поверх старого», что означает, что мы должны отрисовать тот же объект с теми же координатами, но другим цветом – цветом фона (например, белым). А значит нам необходимы снова два действия, как и на шаге 2:

```
SetBrushColor(clWhite); //установка цвета фона для стирания
Circle(x,y,30);//рисуем белым цветом солнце поверх старого изображения
```

5. *Изменение координат* необходимо, чтобы объект мог после стирания появиться уже на новом месте, что и создаст иллюзию движения. Представим, что

солнце должно двигаться не только вправо, но и немного вниз. А значит изменение координат будет выглядеть следующим образом:

```
x:=x+5; y:=y+1;//по горизонтали движение происходит чуть быстрее
```

После этого шага можно снова возвращаться к шагу 2. Но описанный нами набор из нескольких действий создаст лишь одну «итерацию» движения. Мы же хотим, чтобы движение продолжалось некоторое продолжительное время. Необходимо включить в программу цикл. Вид цикла может быть выбран учеником самостоятельно. Если мы точно рассчитали, что для красивой анимации достаточно выполнить 150 шагов, то действия 2–5 необходимо встроить внутрь следующей конструкции:

```
For i:=1 to 150 do
Begin {описанные выше действия} End;
```

Если же мы решим, что циклическое повторение необходимо выполнять, пока соблюдается некоторое условие, то более подходящей конструкцией будет

```
While (условие) do
Begin {описанные выше действия} End;
```

Очень часто нам необходимо реализовать бесконечный цикл. Самое элегантное решение данной задачи – использовать цикл с постусловием в следующем виде:

```
Repeat
{описанные выше действия}
Until false;
```

Конечно, до готового к запуску алгоритма ещё далеко: нужно правильно составить структуру программы, безошибочно описать переменные и выполнить все разобранные шаги. Тестирование всегда выявляет разного рода ошибки, но в нашем случае это вовсе не плохо, ведь у учеников появляется возможность критического оценивания собственного труда, анализа допущенных ошибок. Кто-то понимает, что более правильным было бы внутри цикла сначала стирать объект, а потом менять координаты и рисовать снова, кто-то догадается, что скорость

движения можно не задавать фиксированно, а получать случайным образом в некотором диапазоне.

Приведу пример работающей программы на языке Pascal с учётом необходимых доработок:

```
Program Sun;
Uses graphABC;
Var x,y:integer;
Begin
Setpencolor(clWhite);
X:=150; Y:=100;
While (x<640) do
Begin
setBrushColor(clWhite);
Circle(x,y,30);
X:=X+5;
Y:=Y+1;
setBrushColor(clYellow);
Circle(x,y,30);
Sleep(10);
End;
End.
```

Большая часть учеников пробует «передвигать» более сложный объект, чем круг. Кто-то рисует корабль, который перемещается над синей гладью моря, а кто-то – самолет, парящий в небе от одного края экрана до другого. Каждый за время проведения занятий с графикой в Pascal создаёт свой индивидуальный проект, сложность которого ограничена лишь способностями ученика.

К графике мы возвращаемся ещё не раз. Она позволяет лучше продемонстрировать работу с массивами на примере рисования движущегося звёздного

неба или падающих снежинок. С помощью графики легче воспринимаются операторы *case* и *readkey*, которые можно использовать для отслеживания нажатия клавиш и реакции на них игрового персонажа.

А немного поднаторев, мои ученики самостоятельно разрабатывают свои первые компьютерные игры, создавая героя, который может перемещаться по экрану, собирая сокровища и обходя препятствия.

Очень важно создать для своих учеников правильную мотивацию, заставляющую их не просто любить свой предмет, но и понимать, где полученные знания могут им пригодиться. Я не готов утверждать, что все заинтересовавшиеся программированием ученики в дальнейшем посвятят себя этой отрасли. Программирование – слишком обширная тема, и вряд ли в школьном курсе информатики возможно хоть на 10% затронуть все имеющиеся возможности этого увлекательного занятия. Кто-то выбирает для себя «Олимпиадное программирование», нацеливаясь на решение сложных задач, которые зачастую не имеют практического применения. Кому-то в будущем становится интересно программирование графики для игр с использованием всех возможностей современных видеокарт. Кого-то интересует сфера робототехники, где основы программирования также играют немаловажную роль. Для меня же, как учителя, важным является то, что через формирование атмосферы созидания мне удалось на уроках информатики, связанных с программированием, побудить своих учеников не просто с большей отдачей изучать новые конструкции и команды, но и пробовать самостоятельно применять полученные знания для реализации собственных проектов.