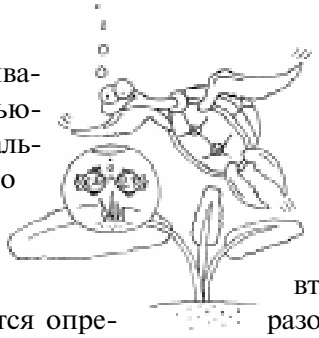
Кузнецова Ирина Николаевна

МОДЕЛИРОВАНИЕ В СРЕДЕ ЛОГО
ЗАНЯТИЕ 5. СОЗДАНИЕ МИКРОМИРОВ



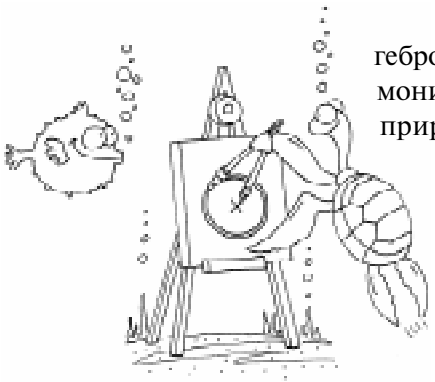
Микромиром называют виртуальную (компьютерную) модель части реального или воображаемого мира. При моделировании ставят различные цели, в зависимости от цели микромир населяется определенными объектами.

Среда Лого Миры предназначена именно для создания микромиров. Роли различных объектов в них играют черепашки, событиями являются щелчки мыши, события обрабатываются процедурами (методами).

В [1] описаны основные этапы моделирования:

- постановка задачи,
- разработка модели
- компьютерный эксперимент
- анализ результатов моделирования

ЗАДАНИЕ 1.
Природа и математика



Можно ли «алгеброй описать гармонию» прекрасных природных форм? Цветок, лист травы, очертания дерева, облака – как математически описать эти контуры?

При изучении форм растений и других живых организмов используются разные математические средства.

Одно из таких направлений называется фрактальной геометрией.

Фракталы позволяют описывать замысловатые изгибы облаков и множество других деталей ландшафта. Фрактальная геометрия – это эффективный путь для рисования реалистических природных объектов на экране компьютера.

Ландшафтные дизайнеры начинают с основной формы и потом повторяют ее снова и снова. Таким же образом создаются пейзажи для фантастических фильмов.

В 80-х годах XX века Бенуа Мандельброт обнаружил, что фрактальное описание поверхности излома металла позволяет измерить прочность этого металла.

Эволюция различных экосистем может быть описана и прогнозирована с использованием фракталов. Например, Г. Гастинг использовал фракталы для моделирования динамики развития болот. Фракталы помогают изучить и определить распространение кислотных дождей и других популяций.

Фракталы применяются для описания объектов астрономии, метеорологии, экономики, экологии, при изучении галактических кластеров.

Фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому.

Б.Мандельброт
(http://home.ural.ru/~shabun/fractals/fractals.htm).

Фрактал можно описать так: имеется последовательность отрезков заданной длины, расположенных определенным образом (см. рисунок 1).

Эта последовательность называется *правило*. На каждом из таких отрезков стро-

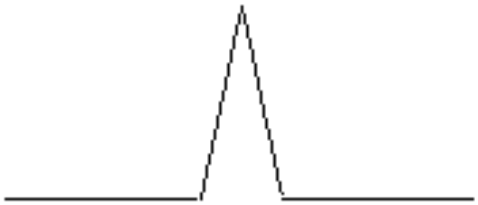


Рисунок 1.

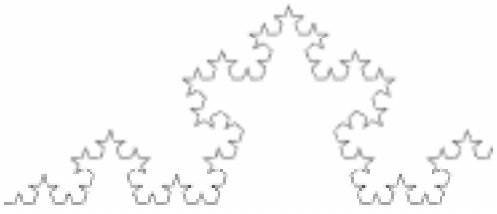


Рисунок 2.

ится правило из отрезков уменьшенной длины. Затем на каждом новом отрезке снова строится правило из отрезков уменьшенной длины. Так продолжается либо указанное число раз (уровней) либо до выполнения заданного условия (см. рисунок 2).

Удобно строить фрактал, как траекторию пути черепашки, выполняющей приписанный ей алгоритм. Очевидно, что при этом используется рекурсивный алгоритм (см. Занятие 3).

Существует несколько программ для построения фракталов, например программа *Frac Tree* и другие (http://fractals.nsu.ru/fractals.chat.ru/index.htm).

В среде Лого также давно уже известны опыты построения и использования фракталов. Например, в коллекцию системы *MicroworldPro* включен проект Fractal.mw2, демонстрирующий разные фракталы. Один из ранних опытов – создание пейзажей в среде *LogoWriter* в гимназии № 470 Санкт-Петербурга. На рисунке 3 представлена ночная дорога, построенная Алексеем Гороховым, учеником 7 класса в 1997 году.

На этой картине с помощью фракталов нарисован контур гор и деревьев.

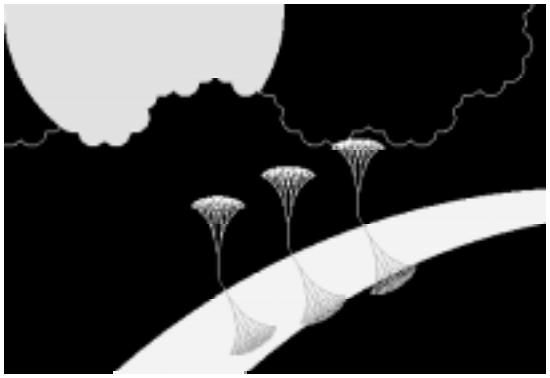


Рисунок 3.

Рассмотрим сначала программу рисования дерева. Дерево состоит из ствола

и веток. При этом ствол и ветки – это линейные отрезки, расположенные под определенным углом друг к другу. Каждый отрезок меньше предыдущего. Если длина отрезка меньше заданного числа, рисование заканчивается.

```
to tree "n
if :n < 2.5[stop] ; минимальный отрезок меньше
                  ; 2,5 шага черепашки
; правило рисования одного участка:
fd :n rt 10
tree (0.75 * :n )
lt 20 tree (0.75 * :n)
rt 10 bk :n
end
```

Правило рисования одного участка следующее:

Рисуется отрезок длинной `:n` шагов; поворот направо на 10 градусов

Рисуется отрезок длиной `0.75 * :n` ; поворот налево на 20 градусов

Рисуется отрезок длиной `0.75 * :n` ; поворот направо на 10 градусов

Возврат в начальную точку.

В ходе рекурсии вместо каждого отрезка рисуется новая ветка.

Для рисования контура гор используется так называемый фрактал Коха.

Первый участок описан в процедуре **rule**.

```
to rule "s "a    ; s – длина отрезка,
                 ; a – угол поворота
fdf :s lt :a ; вперед на :s, налево на :a
fdf :s rt 2 * :a ; вперед на :s,
                  ; направо на 2 * :a
fdf :s lt :a ; вперед на :s, налево на :a
fdf :s ; ; вперед на :s
end
```

В процедуре **koch** описано правило построения следующих участков, а именно – вместо каждого отрезка строится описная в **rule** последовательность отрезков уменьшенной длины (**0.75 * :s**):

```
to koch "s "a
if or colorunder = 4.4 :s < 12[rule :s :a stop]
koch :s * 0.75 :a lt :a
koch :s * 0.75 :a rt 2 * :a
koch :s * 0.75 :a lt :a
koch :s * 0.75 :a
end
```

Обратим внимание на процедуру **fdf**. Это описание команды «вперед» внутри ограды цвета с номером 5.

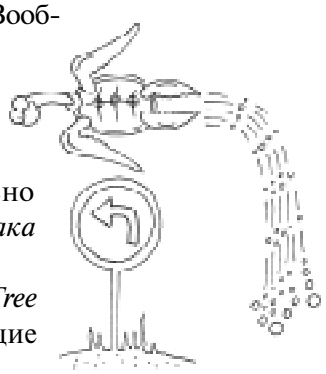
```
to fdf "s
if :s < 2 [stop]
pu fd 1
ifelse colorunder = 4.4 [stop ]
[if :s < 2[stop] bk 1 pd fd 1 fdf :s - 1]
end
```

Ниже приводится программа рисования ограды, внутри которой создается рисунок.

```
to ogr
setc 5 fill pu fd 165 rt 90
pd setc 9
fd 300 rt 90 fd (300 + 30) rt 90
fd (300 * 2) rt 90 fd (300 + 30) rt 90
fd 300 pu rt 90 fd 50
pd setc 9 fill pu setc 1 home pd
end
```

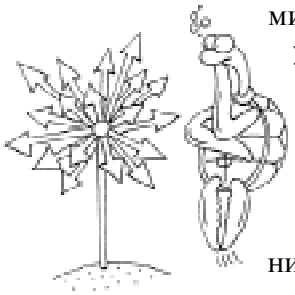
ЗАДАНИЕ 2.
Описание построения фрактала из программы FracTree (1997 год)

В профессиональном приложении *FracTree* графическое изображение также рисуется черепахой. Воображаемая черепаха читает строку знаков и изменяет свое положение или направление согласно смыслу текущего *знака* (команде).



Черепаха *FracTree* понимает следующие знаки:

«F»: Переместиться вперед на один шаг (текущего размера) и рисовать линию.
«f»: Переместиться вперед на один шаг без рисования линии.
«+»: Поворот по часовой стрелке на угол *delta*.
«-»: Поворот против часовой стрелки на угол *delta*.
«|»: Поворот на 180 градусов.
«[»: Запомнить состояние черепахи в стеке; это создает точку ответвления.
«]»: Восстановить состояние черепахи из стека; это возвращает черепаху назад к началу ответвления.
«*»: Увеличение размера шага на 10 %.
«/»: Уменьшение размера шага на 10 %.
«,»: Изменение размера шага случайным образом.



В этих командах упоминается угол *delta* – угол, на который поворачивается Черепаха. Для получения значения угла *delta* число 360 делится на число направлений *(direction)*:

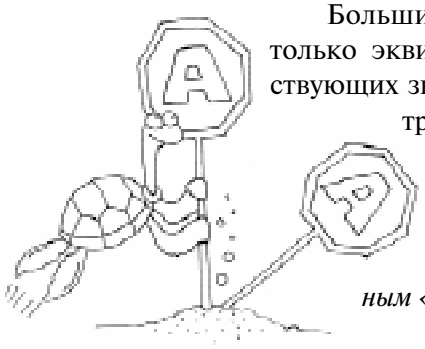
360/direction.

Число *direction* задается параметром при выборе вида фрактала.

В правилах могут использоваться и другие знаки, которые описаны с помощью уже известных.

В дополнение к этим командам, поведение черепахи определяется двумя параметрами: размером шага и углом поворота. Размер шага автоматически отрегулирован, так, чтобы результирующее изображение заполнило экран. При этом вычисляется размер шага во время интерпретации последовательности команд без фактического рисования результирующего образа на экране.

В процессе обработки строки каждый *знак* во входной строке заменяется другим в соответствии с заданным правилом. Правила содержатся в *таблице правил*. Таблица правил содержит *заменяющую* строку для каждого *знака*.



Большинство строк – только эквивалент соответствующих знаков, задающих тривиальные правила подобно следующему:

«A» становится заменённым «A» и т. д.

Другие строки указывают на нетривиальные правила подобно следующему:

«F» заменяется на *«F-F ++»*.

Начальная строка называется *аксиомой*. Для кривой Koch это – только *«F»*, но она может быть более сложной для других фракталов. Аксиома – это нулевая стадия рекурсивного процесса. *N*-ая стадия получена заменой всех составляющих согласно правилам. Это рекурсивный процесс, который производит все более сложные строки. Полученные строки затем интерпретируются черепахой, которая рисует фрактал.

Приводим первые стадии кривой Koch как пример:

Кривая Koch – классический пример фрактала.

Аксиома – *«F»*.

Единственное правило замены – «$F \rightarrow$ *F-F ++ F-F*»,
число направлений (*direction*) равно 6, то есть каждый поворот происходит на угол 360/6 = 60 градусов.

Рассмотрим переписывание строки по стадиям:

Стадия 0: *«F»* (аксиома)

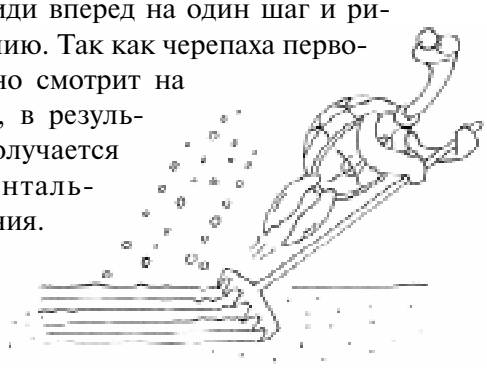
Стадия 1: *«F-F++F-F»*

Стадия 2: *«F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F»*

Стадия 3: *«F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F -F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F ++ F-F++F-F-F-F++F-F»*

Стадия 0

Стадия 0 – сама аксиома. *«F»* значит – иди вперед на один шаг и рисуй линию. Так как черепаха первоначально смотрит на Восток, в результате получается горизонтальная линия.



Стадия 1

Применяя заменяющее правило к предыдущей строке *«F»* получаем строку: *« F-F ++ F-F »*.

Это значит – рисуй вперед один шаг, поворот против часовой стрелки (-), рисуй вперед один шаг (*F*), поворот по часовой стрелке дважды (++), рисуй вперед один шаг (*F*), поворот против часовой стрелки (-), рисуй вперед один шаг (*F*). При этом программа автоматически уменьшила размер шага, чтобы делать фрактал, вписывающимся в окно.

Стадия 2

Те же самые правила, действующие на предыдущую строку дают: *«F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F»* и так далее.

ЗАДАНИЕ 3.
Описание построения фрактала в среде Лого

В среде Лого мы хотели создавать декорации для различных иллюстраций, игр.

Для этого на рабочем поле потребовалось ограничить пространство сцены – см. процедуру **fence**.

```
to fence
cg setc 5 fill pu
fd 165 rt 90 setx xcor - 300
pd setc 9
repeat 2 [fd 500 rt 90 fd 280 rt 90]
pu fd 10 rt 90 fd 10
pd fill pu setc 1 home pd
make "#p "pd
end
```

Мы отказались от вычисления масштаба для рисования итогового фрактала. Для подбора параметров на рабочем поле расположены бегунки и кнопки (см. рисунок 4).

Назначение бегунков:

delta – угол поворота черепашки при рисовании;

smin – минимальная длина отрезка, при достижении которой рисование фрактала останавливается;

length – начальная длина отрезка;

level – число уровней – считается при рисовании фрактала и выводится на бегунок.

Назначение кнопок:

reset – сброс текущего состояния, рисование начального состояния сцены;

koch – запуск рисования фрактала Коха с заданными параметрами.

Команды, предложенные в описании системы FracTree, интерпретированы в среде Лого следующим образом (см. таблицу 1).

ЗАДАНИЕ 4.
Пример описания и моделирования фрактала Коха

Выбираем начальные установки – положение Черепашки, угол поворота, цвет.

```
to 0pos "spos "shead
pu setpos :spos seth :shead setc 5 + random
120
end
```

Задаем *правило* совокупности отрезков для фрактала Коха. Это правило приведено в задании 2.

```
to rule_koch
fpd -h fpd +h +h fpd -h fpd
end
```

Программа построения остальных стадий формируется так: каждая команда **fpd** заменяется на **s_koch**. При этом длина отрезка каждый раз уменьшается на 10 %,

```
to s_koch :side
ifelse colorunder = 4.4 [ stop]
  [ifelse :side < smin [ rule_koch setlevel
                                  level + 1 ]
    [ s_koch /side -h
    s_koch /side +h +h
    s_koch /side -h
    s_koch /side ]
   ]
end
```

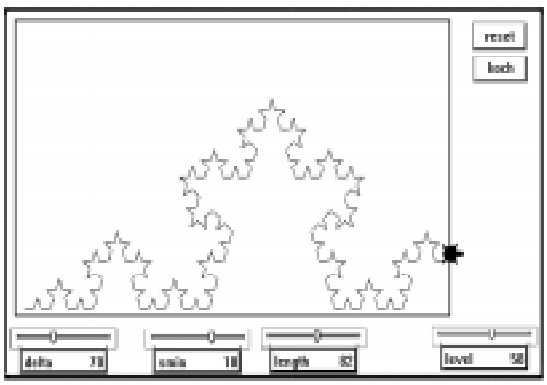


Рисунок 4.

Таблица 1.

Команда системы *FracTree*	Процедуры в среде *Лого*
«+» – поворот по часовой стрелке на угол delta	Поворот по часовой стрелке (**+h**) на угол **delta** **to +h** **rt delta** **end**
«-» – поворот против часовой стрелки на угол delta	Поворот против часовой стрелки (**-h**) на угол **delta** **to -h** **lt delta** **end**
«|» - поворот на 180 градусов	Поворот на угол 180 градусов (**!**) **to !** **seth 180** **end**
«[» - запомнить состояние черепахи в стеке. Это создает точку ответвления.	Запомнить состояние черепахи в стеке (состояние пера запоминается при движении вперед) **to in** **make "stack fput (list pos heading color) :stack** **end**
«]» – восстановить состояние Черепашки из стека. Это возвращает Черепаху к точке ответвления.	Восстановить состояние Черепашки из стека **to from** **if not empty? :stack [make "#st first :stack** **frompos fromhead fromcolor** **make "stack bf :stack]** **end** Восстановление координат Черепашки **to frompos** **make "#w item 1 :#st** **pu setpos :#w run :#p** **end** Восстановление направления головы Черепашки **to fromhead** **make "#w item 2 :#st** **seth :#w** **end** Восстановление цвета Черепашки **to fromcolor** **make "#w item 3 :#st** **setc :#w** **end**
«*» – увеличение размера шага на 10 процентов	Увеличение размера шага на 10 процентов **to *side** **op 1.1 * :side** **end**
«/» – уменьшение размера шага на 10 процентов	Уменьшение размера шага на 10 процентов **to /side** **op 0.9 * :side** **end**
«,» – изменение размера шага случайным образом	Изменение размера шага случайным образом **to ,side** **op (random 100) / 10** **end**

Подготавливаем запускающую программу, которая будет запускаться кнопкой:

```
to kokh
0pos [-290 -110] 90
setlevel 0
make "side length
s_koch :side
end
```

Устанавливаем на бегунках значения параметров:
length = 70
delta = 80
smin = 15

Щелкаем на кнопку *kokh* и получаем узор, показанный на рисунке 4.

Домашнее задание

1) Попробуйте сами подобрать параметры для получения контуров, изображенных на рисунках 5 и 6. Сколько уровней потребовалось для построения?

2) Составьте интерпретацию фрактала *Brush* по следующему описанию:

Этот пример иллюстрирует переходы.

Начальное состояние (аксиома) – «--- *F*», **правило** – «$F \rightarrow F\ [+\ F]\ F\ [-F]\ F$», угол поворота *delta* – 360/12.

Аксиома требует установки черепахи так, чтобы она смотрела на Север.

Стадия 1: « --- *F [+ F] F [-F] F* ». После продвижения вперед на один шаг («*F*»), состояние черепахи сохраняется в стеке («*[*»). Знак «+» означает поворот черепахи по часовой стрелке на один угол *delta*.

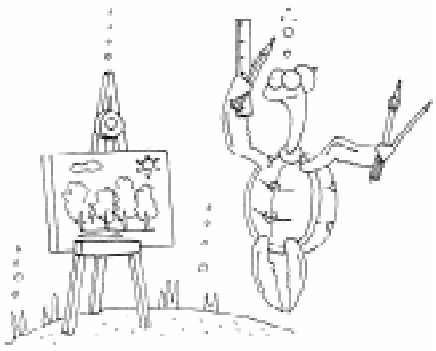
Следующий знак «*F*» рисует ветвь на правой стороне от стебля.

Знак «*]*» восстанавливает состояние черепахи из стека. Это возвращает черепаху назад к началу ветви. Следующий знак «*F*» рисует следующую часть стебля, в то время как подстрока «*[-F]*» ведет к рисованию ветви слева от стебля.

Последний знак «*F*», наконец, заканчивает стебель.

При обработке строки на второй стадии каждый из 4 знаков «*F*» заменяется на строку «*F [+ F] F [-F] F*». Прямые части стебля получают дополнительные участки, и предыдущие ветви получают подветви. Каждая новая стадия добавляет большое количество деталей растения.

ЗАКЛЮЧЕНИЕ



На скромных примерах мы хотели показать, как с помощью математики можно создавать пейзажи, узоры. Фракталы – не единственная область математики, чарующая своей красотой.

Рисунок 5.

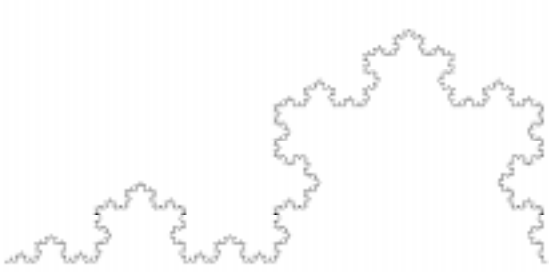


Рисунок 6.

В марте 2002 года в сети Интернет появился обзор статьи «A generic geometric transformation that unifies a wide range of natural and abstract shapes» – «Общее геометрическое преобразование, которое объединяет широкий ряд естественных и абстрактных форм» (http://astronomy.swin.edu.au/~pbourke/curves/).

Автор этой статьи Johan Gielis предлагает новый подход для моделирования разнообразных абстрактных, природных и созданных человеком форм. Взяв за основу уравнение окружности, автор рассказывает, как, добавляя логарифмические и тригонометрические зависимости в это уравнение, можно получить самые разнообразные геометрические формы.

На следующем занятии мы разберем этот материал и покажем в среде Лого, что может нарисовать черепашка, следуя математическим зависимостям.

Кузнецова Ирина Николаевна, учитель информатики школы № 640, координатор секции Лого-Лего Международной конференции «Школьная информатика и проблемы устойчивого развития».