Усенков Дмитрий Юрьевич

«МАГИЧЕСКИЙ ГЛАЗ» – НА ЭКРАНЕ КОМПЬЮТЕРА

С десяток лет тому назад стали модными различные альбомы «Магический глаз» (или аналогичные с другими названиями) с довольно занимательными картинками. На первый взгляд такая картинка выглядит как прямоугольник, заполненный какой-нибудь регулярной графической структурой – например, это может быть зеленый фон с регулярной «сеткой» из красно-синих цветочков. Но если присмотреться повнимательнее и немного скосить глаза, чтобы совместить друг с другом соседние ряды цветочков (иногда для облегчения задачи над картинкой печатаются «метки совмещения» в виде черных кружочков или крестиков), и на этом пестром фоне явственно проступает какое-либо объемное изображение. Подобные трюки нередко используются и в качестве рекламы на обложках газет и журналов, на различных настенных календарях и т. п.

Читателям, знакомым с основами формирования объемных изображений с помощью стереопар, такая картинка может показаться удивительной: объемность есть, а стереопар как таковых нет! Но при более внимательном рассмотрении нетрудно понять, что в качестве стереопар здесь выступают соседние вертикальные полосы картинки, несущие в себе фрагменты «скрытого» объемного изображения.

Секрет реализации стереоэффекта «магического глаза» не очень сложен, хотя понять его удается не сразу. Общая идея здесь заключается в следующем.

Мысленно пронумеруем слева направо одинаковые вертикальные полосы, из которых состоит картинка (рис. 1). Крайняя левая полоса (с нулевым номером) будет «опорной» и всегда остается без изменения.

Предположим, что нам нужно расположить «объемный» квадратик в полосе под номером 1. Тогда, рассматривая полосы 0 и 1 как половинки стереопары, мы должны в соответствующем месте полосы 1 часть фона, лежащую под нашим квадратиком, сдвинуть на несколько миллиметров вправо или влево (в зависимости от того, должен ли квадратик быть «выпуклым» или «вдавленным» в фон). Но если мы сделаем это только в полосе 1, а полосу 2 оставим без изменений, то, вместе с требуемым квадратиком, на полосе 2 проявится еще один «паразитный» квад-

Но если присмотреться повнимательнее... на этом пестром фоне явственно проступает какое-либо объемное изображение.

ратик, смещенный «по глубине» в противоположном направлении относительно нашего (например, если в полосе 1 мы сформировали «выпуклый» участок, то в полосе 2 он окажется «вдавленным»). Причина этого ясна: различие между измененной полосой 1 и оставшейся без изменения полосой 2 тоже можно интерпретировать как такую же стереопару, что и из полос 0 и 1, только с переставленными местами левым и правым «кадрами».

Что мы должны сделать, чтобы убрать этот «паразитный» квадратик? Ответ прост: нужно сделать полосу 2 точно такой же, как полоса 1, то есть точно так же сместить на ней соответствующий участок фона. Но тогда «паразитный» квадратик появится уже на оставшейся неизменной полосе 3, поскольку приведенные выше рассуждения верны и здесь. Следовательно, чтобы правильно сформировать на полосе 1 «объемный» участок, нужно сдвинуть влево или вправо соответствующий фрагмент фона на ней и правее на всех последующих полосах до крайней правой включительно.

Если теперь нам нужно расположить еще один «объемный» участок в полосе 2, то мы делаем для соответствующего ему фрагмента фона те же самые действия – смещаем вправо или влево ранее измененное изображение во всех полосах со второй до крайней справа. И так далее (рис. 1).

Если же нужно создать картинку «магический глаз» для сложного объемного изображения, то ход работы будет следующим:

1) разбить исходную объемную картинку на «минимальные элементы» определенного размера (своего рода «пиксели»; в компьютерном представлении их наименьший размер равен одной точке, но ради ускорения процесса можно принять в качестве «пикселя» фрагмент 2×2, 3×3, 4×4 точки или более, пожертвовав четкостью объемного изображения);

2) для каждого «пикселя» определим требуемое значение «глубины»/»выпуклости», формируя двумерный массив этих значений (для упрощения задачи можно указывать значения «глубины» с некоторым шагом, формируя набор «изолиний» – как на географической карте для гор и низменностей – и разбивая таким образом исходную объемную картинку на набор наложенных друг поверх друга плоских «срезов»; далее каждый из таких «срезов» делится на вертикальные полоски той же ширины, что и полосы фона, а смещение участков фона ведется для тех «кусочков среза», которые попали на данную полосу);

3) производим соответствующее одному текущему горизонтальному ряду «пикселей» смещение фрагментов фона в следующем порядке:

– цикл перебора сверху вниз строк высотой в один «пиксель»;

– в пределах каждой строки – цикл по возрастанию номеров вертикальных полос фона;

– для каждой полосы – цикл по возрастанию значения «выпуклости»;

– для каждого значения выпуклости – формирование горизонтальных рядов «пикселей-масок» и цикл перебора таких масок слева направо;

– для каждой маски – смещение фрагмента фона соответствующего размера вначале точно под этой маской, а затем на всех последующих полосах до правого края фона.

На первый взгляд, все сказанное кажется очень сложным. Но причина сложности здесь скорее лишь в нео-

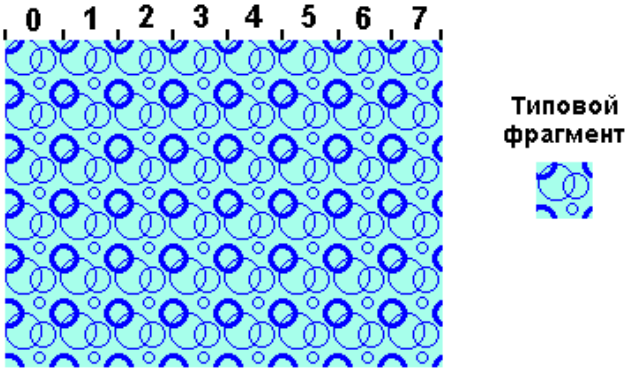

Рис. 1. Возможная заготовка фона для картинки «магический глаз»

бычности (точнее, в непривычности) рассматриваемого явления. Во всяком случае, требуемая последовательность действий, записанная на каком-либо языке программирования (а без компьютера при создании таких картинок обойтись вряд ли удастся), выглядит гораздо проще и понятнее.

Возьмем за основу язык БЕЙСИК (QuickBASIC для MS-DOS) как наиболее простой язык программирования для освоения начинающими (но при желании не составит особого труда переписать программу и на любом другом языке, например, на Си, Паскале или в среде Visual Basic).

Итак, пусть у нас имеется некоторое объемное изображение, для первого раза не очень сложное. Например выпуклые надпись «ПРОБА», номер года («1996» – год, когда секрет «магического глаза» для автора этой статьи перестал быть секретом; кстати, никаких его описаний в литературе тогда не было, так что автору пришлось «изобретать велосипед» совершенно самостоятельно, ведя рассуждения «с нуля») и разделяющая их горизонтальная «цепочка», более выпуклая, чем текст и цифры (рис. 2).

Вообще же, как уже было сказано ранее, исходная объемная картинка может быть любой, в том числе с плавными линиями рисунка и градациями «выпуклости». Мы же ради упрощения задачи будем кодировать надпись в виде набора квадратиков размером в одно символьное знакоместо (на экране компьютера), а в процессе построения работать с графическими точками экрана, попадающими на тот или иной квадратик либо на область фона. Но в любом случае нужно сообщить программе значение выпуклости для каждой точки объемного изображения, и мы записываем в ячейки кодирующего нашу картинку массива следующие числа:

0 – «нулевая глубина», относительно которой будет строиться объемная надпись;

1 – «первая градация выпуклости» для букв и цифр года;

2 – «вторая градация выпуклости» для «звеньев цепочки» посередине рисунка.

Рис. 2. «Шаблон» создаваемого объемного изображения

Теперь выберем фон для построения картинки. Для простоты удовлетворимся точечной фактурой, сгенерированной с помощью оператора RND в двойном цикле по *I* и *J*. При этом дополнительно введенный параметр `ColKol%` определяет максимальное количество цветов точек фона, ограничивая диапазон генерируемых с помощью RND целых случайных чисел, которые затем преобразуются в коды цветов: черного (задействован всегда), белого, красного, зеленого и синего. Если указать `ColKol%` равным 1, то будут использоваться только черный и белый цвет, при `ColKol%=2` – черный, белый и красный, а при максимально допустимом для данного листинга значении 4 – все перечисленные цвета одновременно. Кроме того, еще один параметр `Corr`, представляющий собой дробное число от 0 до 1, позволяет управлять «густотой засева» фона точками (при `ColKol%=1` желательно выбрать значение `Corr` равным 0.2 или 0.3; в противном случае допускается и нулевое значение `Corr`).

Так как желательно проследить, чтобы в получаемой фактуре не проявлялось слишком крупных групп из стоящих рядом точек одного и того же цвета (иначе возможна нечеткость полученного объемного рисунка), предусмотрим в программе предварительный визуальный контроль полученного образца фрагмента фона и повторную генерацию случайных элементов массива по команде пользователя. А затем сформируем на экране фон для будущей объемной картинки, выводя полученные прямоугольные фрагменты в четыре ряда по восемь фрагментов в каждом (плюс не используемый при построении картинки

неполный нижний ряд). В итоге должна получиться требуемая регулярная структура, состоящая из одинаковых вертикальных полос шириной в 32 графических точки каждая. (Заметим, что нам важна идентичность именно вертикальных полос, хотя при нашем способе генерации фона ряды тоже будут одинаковыми. Вообще же можно было сгенерировать случайным образом сразу целую вертикальную полосу 32×200 точек и повторить ее вывод на экран восемь раз, но тогда потребуется увеличить размеры массива **F%**.)

И вот теперь можно приступать к построению объемного изображения. Внешний цикл по **Y0** (первый в порядке вложенности) производит перебор точек фона по высоте.

Крайние полосы (нулевую и последнюю) мы оставляем в исходном виде – левая полоса будет «опорной» (в объемной картинке тоже предусмотрено левое поле «нулевой выпуклости» шириной в четыре символьные позиции, что как раз равно 32 точкам), правая нужна для предохранения выхода за край экрана. Начинаем перебирать вертикальные полосы слева направо начиная с первой – в программе этому действию соответствует цикл по **NP**.

Далее в листинге располагается цикл перебора по «выпуклости» точек начиная с значения 1 и до максимального (напомним, что у нас «выпуклость» задана целыми числами 1, 2 и т. д., хотя в принципе можно предусмотреть и плавное изменение этой величины).

И, наконец, производится непосредственное построение объемного изображения. Просматривая исходную картинку, мы выделяем в ней горизонтальный ряд точек текущей выпуклости, длина которого равна ширине вертикальной полосы фона, формируя «буферный» массив **B%**. В нем для точек с текущим значением выпуклости хранятся цвета «лежащих под ними» точек фона (как если исходная картинка была бы просто наложена поверх фона), а для прочих точек – нули. Далее мы выводим этот ряд со смещением **SM%**, накладывая его поверх предыдущего фона и при необходимости выходя за правый край текущей вертикальной полосы. А «освободившиеся» точки слева от смещенных мы заполняем точками, имеющими случайный цвет (это позволит избежать их «двоения»). Величина смещения здесь зависит от значения выпуклости и фактически играет роль параллакса в обычной стереопаре. Мы же ради упрощения программы используем простейшую зависимость: **SM%=GL%*2**. В любом случае для значения смещения нужно выполнить два условия: для наименьшего значения **GL%** оно должно быть не менее размеров минимального элемента фона (в нашем случае это одна графическая точка на дисплее), а для максимального **GL%** – не более ширины фоновой полосы.

Завершив обработку точек для текущего значения «выпуклости», аналогичным образом выведем ряд «более выпуклых» точек поверх полученных на предыдущем этапе, затем повторим тот же самый процесс (для выпуклостей от 1 до максимальной) для следующей справа по порядку полосы, а когда будут отработаны все точки с данной вертикальной координатой **Y**, увеличим ее и продолжим цикл до достижения самого нижнего ряда в крайней справа полосе. И в заключение выведем в верхней строке экрана (не занятой картинкой) две «метки совмещения» в виде сетчатых квадратиков, расстояние между которыми равно ширине одной вертикальной полосы: для правильного рассматривания полученной картинки нужно будет скосить глаза так, чтобы обе «метки совмещения» слились в одну.

Напоследок отметим следующее. Точно так же, как и обычные стереопары, картинка «магический глаз» может быть «прямой» (когда надо для получения эффекта объемного изображения смотреть «сквозь» нее в бесконечно удаленную точку) и «обратной» (когда надо скосить глаза к носу, фокусируясь на некоторую точку, расположенную ближе, чем картинка). И точно так же «обратный» способ рассматривания картинки гораздо легче, чем «прямой». Поэтому в предлагаемом лис-

тинге мы строим именно «обратное» изображение. Если же смотреть «сквозь» картинку (по «прямому» способу), то «выпуклые» участки превратятся в «вдавленные»(см. листинг 1).

Конечно, эту программу нельзя назвать наилучшей из возможных (в Интернете сегодня можно отыскать немало аналогичных программ, более красиво оформленных, быстрее работающих, реализующих более высокое качество картинок). «За бортом» осталось построение многоцветных картинок с плавным изменением выпуклости и глубины одновременно и с

...картинка «магический глаз» может быть «прямой»... и «обратной»...

Листинг 1

```
'   Построение объемных картинок без стереопар
' ------------------------------------------

DIM A%(31, 19), F%(31, 39), B%(31)
RANDOMIZE TIMER ' реинициализация генератора случ. чисел

' исходные данные
DX = 32          ' ширина фрагментной полосы картинки
DY = 40          ' высота фрагмента фона
X0 = 20          ' отступ картинки от левого края экрана
Y0 = 20          ' отступ картинки от верхнего края экрана
ColKol% = 1      ' максимальное кол-во цветов в фоне (без учета черного) - до 4
Corr = .3        ' коррекция генератора случ. чисел для получения
                 ' требуемой плотности расположения точек фона

' включение графического видеорежима и очистка экрана
CLS
SCREEN 7   ' EGA или VGA, 320x200 точек, 16 цветов

' генерация случайного узора фона
DO
  CLS
  COLOR 15 ' белый цвет
  PRINT ' Генерация случайного узора фона '
 PRINT '--------------------------------'
  PRINT
  COLOR 10 ' зеленый
  PRINT "    Если фон нравится, жмите ПРОБЕЛ,"
  PRINT "    иначе - любую другую клавишу."

  FOR I = 0 TO DX - 1
     FOR J = 0 TO DY - 1
        NColor% = INT(ColKol% * RND + Corr)
        SELECT CASE NColor%
          CASE 0: F%(I, J) = 0  ' черная точка
          CASE 1: F%(I, J) = 15 ' белая точка
          CASE 2: F%(I, J) = 12 ' красная точка
          CASE 3: F%(I, J) = 10 ' зеленая точка
          CASE 4: F%(I, J) = 9  ' синяя точка
        END SELECT
```

```
        PSET (I + 20, J + 70), F%(I, J) ' вывод точек на образце фона
     NEXT J
  NEXT I
  DO: K1$ = INKEY$: LOOP WHILE K1$ = ""  ' дождаться нажатия клавиши
LOOP WHILE K1$ <> " "   ' если не ПРОБЕЛ, перерисовать фон заново

' считываем массив объемной картинки
FOR J = 0 TO 19
   FOR I = 0 TO 31
      READ A%(I, J)
   NEXT I
NEXT J

' массив значений глубины/выпуклости квадратиков 8*8 точек
' соотв. одному знакоместу)
' слева обязательно оставляем "пустыми" четыре знакоместа,
' что соответствует первой фрагментной полосе ("опорной")
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,0,0,1,1,0,0
DATA 0,0,0,0,0,0,1,0,0,1,0,1,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,1,0,1,0,0
DATA 0,0,0,0,0,0,1,0,0,1,0,1,1,1,1,0,1,0,0,1,0,1,0,0,0,0,1,0,0,1,0,0
DATA 0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,1,0,0,1,0,1,1,1,1,0,1,1,1,1,0,0
DATA 0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,1,0,0,1,0,1,0,0,1,0,1,0,0,1,0,0
DATA 0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,1,1,1,1,0,1,1,1,1,0,1,0,0,1,0,0
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2
DATA 0,0,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0
DATA 0,0,0,0,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2,2,2,0,0,2,2
DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,0,0,0
DATA 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0
DATA 0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0
DATA 0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0

' вывод на экран фонового поля
CLS
FOR XI = X0 TO 256 - DX + X0 STEP DX
   FOR YJ = Y0 TO Y0 + 200 - DY STEP DY
      FOR X = 0 TO DX - 1
         FOR Y = 0 TO DY - 1
            PSET (XI + X, YJ + Y), F%(X, Y)
         NEXT Y
      NEXT X
   NEXT YJ
NEXT XI

' фоновое поле выведено - выводим объемные точки
'
' определить максимальную глубину
GM% = 0
FOR I = 0 TO 31
   FOR J = 0 TO 19
      IF A%(I, J) > GM% THEN GM% = A%(I, J)
   NEXT J
NEXT I
```

```
IF GM% <> 0 THEN

   FOR YY = Y0 TO Y0 + 159
   ' перебор точек по высоте

      FOR NP = 0 TO 256 / DX - 2
      ' перебор полос слева направо без учета крайних двух

         FOR GL% = 1 TO GM%
         ' перебор по глубинам

          SM% = GL% * 2 ' упрощенная формула расчета параллактического смещения

              FOR I = 0 TO DX - 1 ' инициализация буферного массива
               B%(I) = -1
             NEXT I

             FOR XX = DX - 1 TO 0 STEP -1
              ' обработка точек в пределах текущей полосы
              IF A%((NP * DX + XX) \ 8 + 4, (YY - Y0) \ 8) = GL% THEN
                B%(XX) = POINT(NP * DX + XX + X0, YY)
                ' если данной точке соответствует текущая глубина,
                '  записать в буфер ее цвет, иначе остается "-1"

                IF XX - SM% >= 0 THEN
                   ' а "прежнюю" точку закрасить случ. цветом
                   NColor% = INT(ColKol% * RND + Corr)
                   SELECT CASE NColor%
                     CASE 0: B%(XX - SM%) = 0  ' черный
                     CASE 1: B%(XX - SM%) = 15 ' белый
                     CASE 2: B%(XX - SM%) = 12 ' красный
                     CASE 3: B%(XX - SM%) = 10 ' зеленый
                     CASE 4: B%(XX - SM%) = 9  ' синий
                   END SELECT
                END IF

              END IF

             NEXT XX
            ' буфер заполнен, накладываем его на полосы со смещением

             FOR N = NP + 1 TO 256 / DX - 1
             ' перебор полос от текущей+1 вправо

               FOR XX = 0 TO DX - 1
               ' перебор точек в буфере

                 IF B%(XX) <> -1 THEN
                 ' обрабатываем только смещенные точки
                   IF N * DX + XX + SM% < 256 THEN
                     PSET (N * DX + XX + X0 + SM%, YY), B%(XX)
                     ' если в буфере не -1 и не вышли за край, вывести точку
                   END IF

                 END IF

               NEXT XX ' цикл по всему буферу

            NEXT N ' цикл по всем полосам вправо от текущей
         NEXT GL% ' цикл по глубинам

      NEXT NP ' цикл по горизонтали

   NEXT YY ' цикл по высоте
END IF
' все сделано, выводим метки совмещения
```

```
COLOR 15 ' белый цвет
LOCATE 1, 1
PRINT TAB(13 + X0 \ 8); «-»; TAB(17 + X0 \ 8); «-»;

LOCATE 1, 1
DO: LOOP WHILE INKEY$ = «» ' ожидание нажатия на клавишу

SCREEN 0 ' возвращаемся в текстовый видеорежим
CLS

END
' все....
```

красивым «фотографическим» фоном. Кроме того, все используемые графические операции (копирование с экрана в буферный массив, наложение со смещением и особенно копирование фрагмента фона на весь экран) правильнее было бы производить по принципу работы со спрайтами. Все эти и другие доработки оставлены читателям для самостоятельных изысканий, приведенный же здесь листинг является скорее иллюстрацией наших теоретических рассуждений.

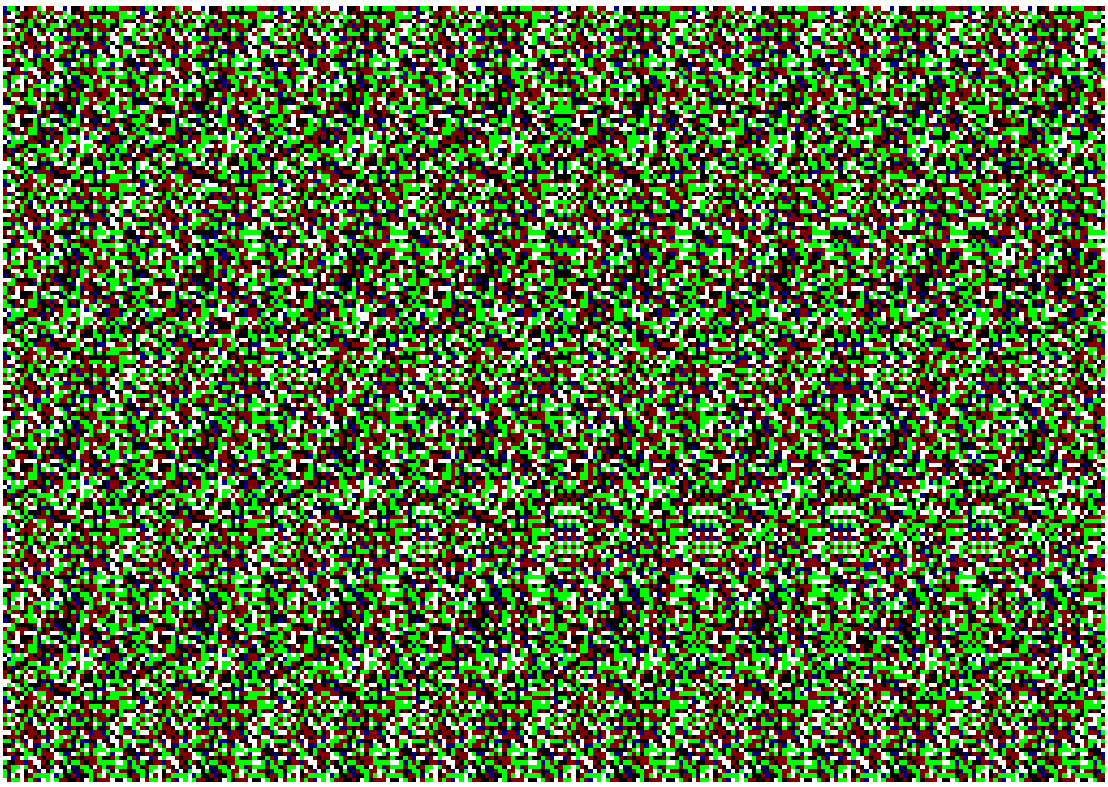

Рис. 3. Результат работы программы

Усенков Дмитрий Юрьевич,
старший научный сотрудник
Института информатизации
образования Российской академии
образования, Москва.