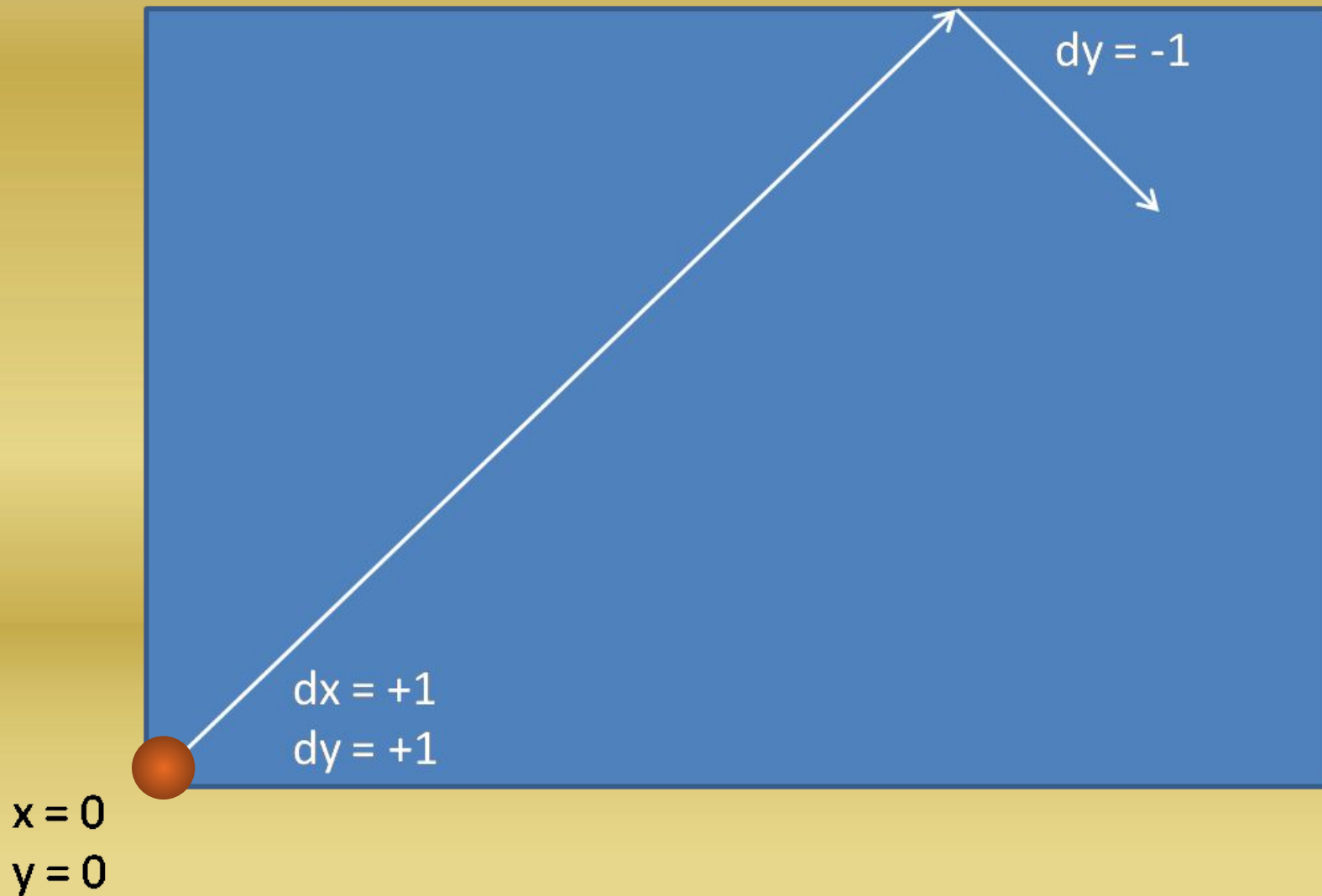


Программирование графической анимации

Задача
«Движение бильярдного шарика»

Пусть у нас уже есть программа движения шарика по диагонали экрана
(щелкните по шарикау на рисунке)

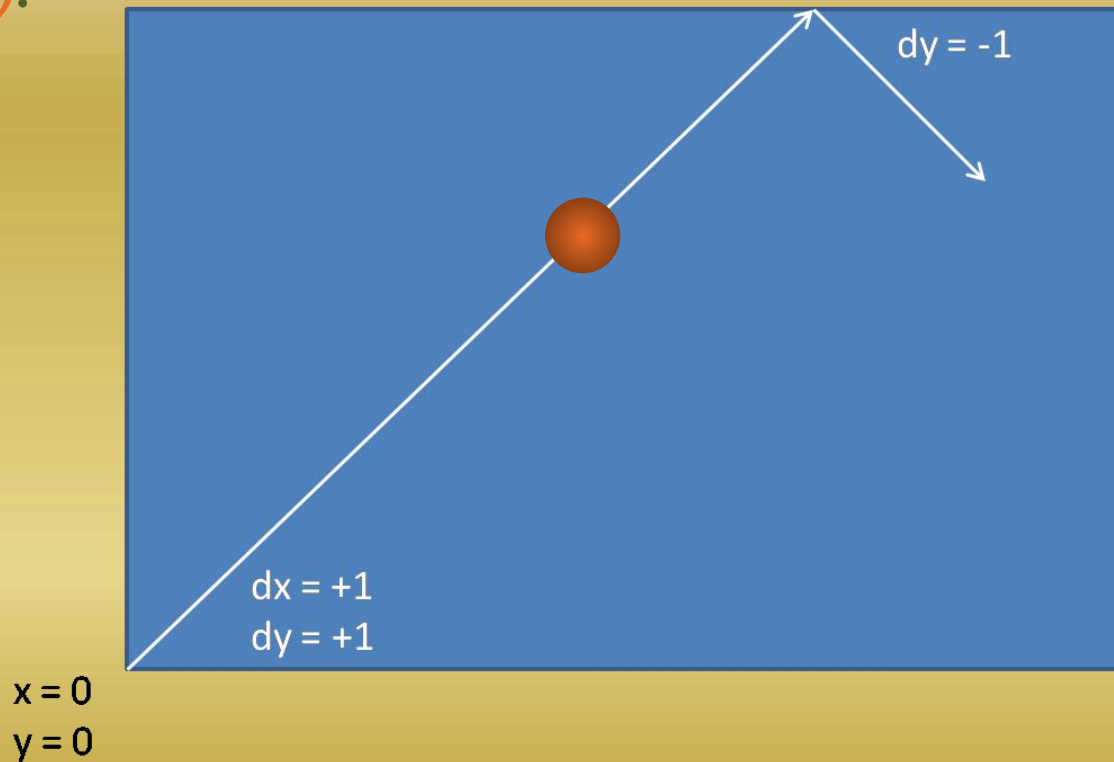


Программа движения шарика по диагонали выглядит следующим образом:

```
SCREEN 9
COLOR 1, 0
WINDOW (0, 0)-(640, 350)
x = 1
y = 1
dx = 1
dy = 1
FOR k = 0 TO 10000
CIRCLE (x, y), 3, 6
PAINT (x, y), 6
FOR i = 1 TO 10000: NEXT i
CIRCLE (x, y), 3, 4
PAINT (x, y), 4, 4
x = x + dx
y = y + dy
NEXT k
```

Здесь **dx**, **dy** – приращения соответствующих координат.

Наша задача состоит в том, чтобы шарик, достигнув какой-либо границы экрана, отскакивал от нее как от стенки (*щелкните по шарiku на рисунке*).



Чтобы шарик не вылетал за верхний край экрана, необходимо, чтобы ордината объекта не превышала 350 пикс (при стандартном размере экрана 640x350)

В связи с возникшей ситуацией в нашей задаче, когда необходимо выполнить определенные условия, напомним, что такое условный оператор.

Условный оператор.

IF <условие> THEN оператор1 ELSE оператор2

**Если условие выполняется, то выполняется «оператор1»,
в противном случае выполняется «оператор2».**

В записи условия можно использовать следующие символы:

=, >, >=, <, <=, <>

Существует неполная форма оператора:

IF <условие> THEN оператор1

Вставим в программу движения шарика условие **IF $y = 350$** , причем, этот условный оператор мы должны добавить в цикл, где в каждой итерации при каждом увеличении координаты Y происходит проверка данного условия. Как только оно выполнится, так координата Y , которая до сих пор увеличивалась с приращением **$dy = +1$** , должна будет уменьшаться и приращение станет равно: **$dy = -1$** .

Вот как будет выглядеть программа с добавлением условного оператора, которое учитывает столкновение шарика с верхней границей экрана:

SCREEN 9

COLOR 1, 0

WINDOW (0, 0)-(640, 350)

x = 1

y = 1

dx = 1

dy = 1

FOR k = 0 TO 10000

CIRCLE (x, y), 3, 6

PAINT (x, y), 6

FOR i = 1 TO 10000: NEXT i

CIRCLE (x, y), 3, 0

PAINT (x, y), 0, 0

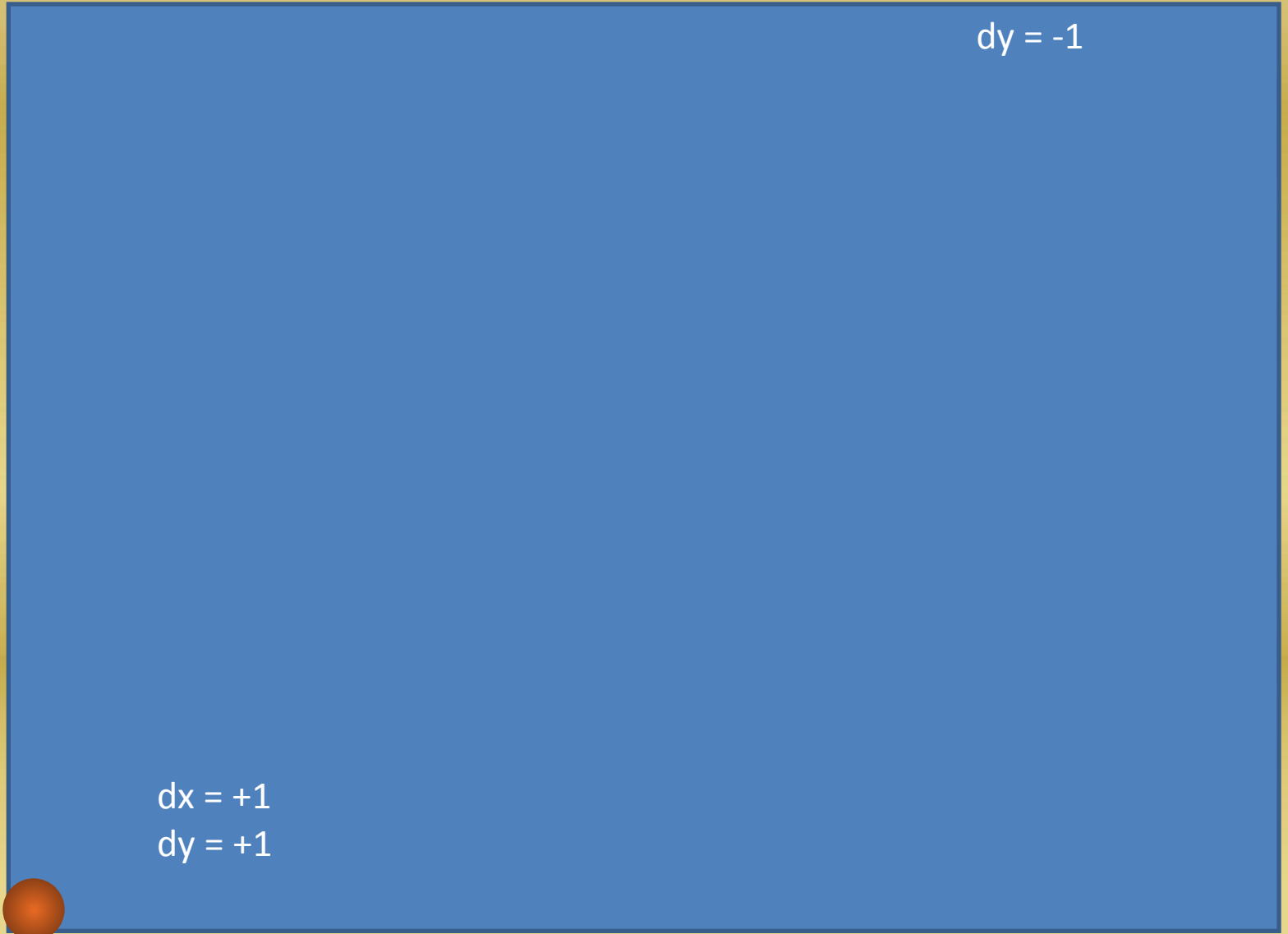
IF y = 350 THEN dy = -1

x = x + dx

y = y + dy

NEXT k

Запустим программу на выполнение:



$dx = +1$

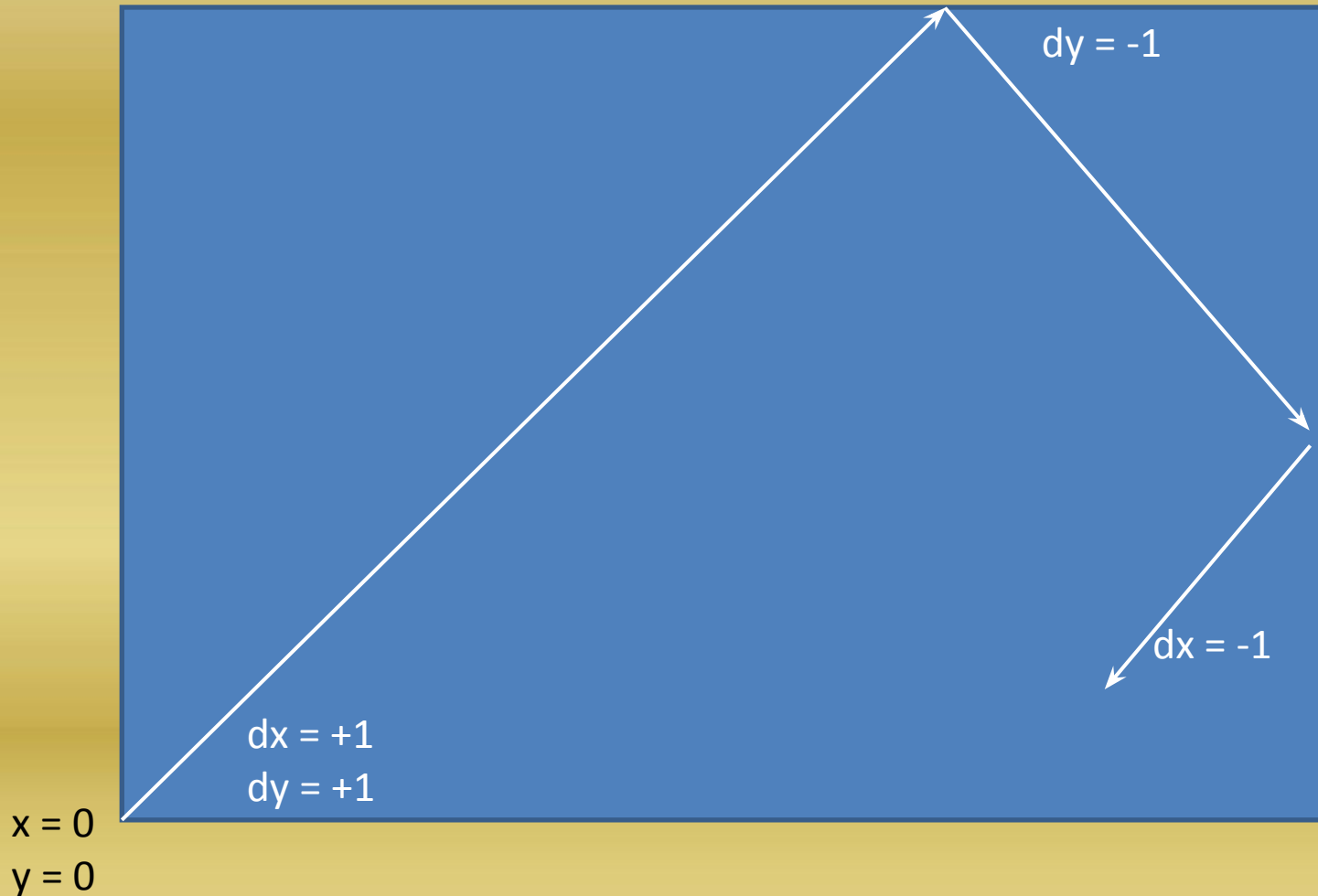
$dy = +1$

$dy = -1$

$x = 0$

$y = 0$

Рассмотрим дальнейшее движение шарика. Достигнув правого края экрана, шарик должен отскочить вниз так, как указано на схеме:



В этом случае координата Y будет по-прежнему уменьшаться, т.е. $dy = -1$. Координата X , которая до сих пор увеличивалась, после выполнения условия $x = 640$ начнет уменьшаться или $dx = -1$, что то же самое.

Вот так будет выглядеть программа с добавлением второго условного оператора, где предусмотрено условие столкновение шарика с верхней границей экрана:

```
SCREEN 9
COLOR 1, 0
WINDOW (0, 0)-(640, 350)
x = 1
y = 1
dx = 1
dy = 1
FOR k = 0 TO 10000
CIRCLE (x, y), 3, 6
PAINT (x, y), 6
FOR i = 1 TO 10000: NEXT i
CIRCLE (x, y), 3, 0
PAINT (x, y), 0, 0
IF y = 350 THEN dy = -1
IF x = 640 THEN dx = -1
x = x + dx
y = y + dy
NEXT k
```

Запустим вновь программу на выполнение:

$dy = -1$

$dx = -1$

$dx = +1$

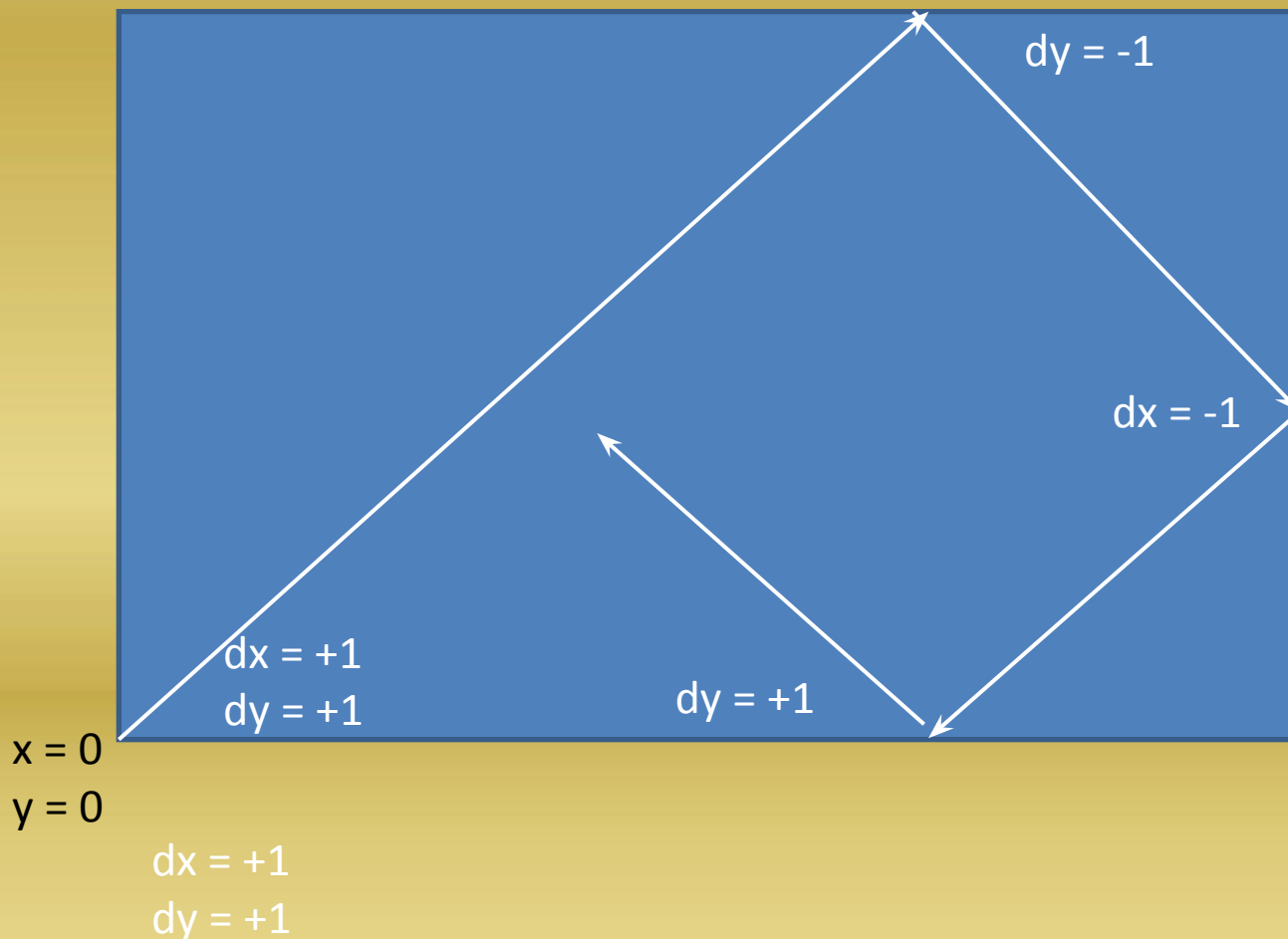
$dy = +1$

$x = 0$

$y = 0$



Рассмотрим дальнейшее движение шарика. Достигнув нижнего края экрана, шарик должен отскочить вверх так, как указано на схеме:



В этом случае координата Y начнет увеличиваться, т.е. $dy = +1$, а координата X будет по-прежнему уменьшаться, т.е. $dx = -1$.

Вот так будет выглядеть программа с добавлением третьего условного оператора, где предусмотрено условие столкновение шарика с нижней границей экрана:

```
SCREEN 9
COLOR 1, 0
WINDOW (0, 0)-(640, 350)
x = 1
y = 1
dx = 1
dy = 1
FOR k = 0 TO 10000
CIRCLE (x, y), 3, 6
PAINT (x, y), 6
FOR i = 1 TO 10000: NEXT i
CIRCLE (x, y), 3, 0
PAINT (x, y), 0, 0
IF y = 350 THEN dy = -1
IF x = 640 THEN dx = -1
IF y = 0 THEN dy = +1
x = x + dx
y = y + dy
NEXT k
```

Запустим программу на выполнение:

$x = 0$
 $y = 0$



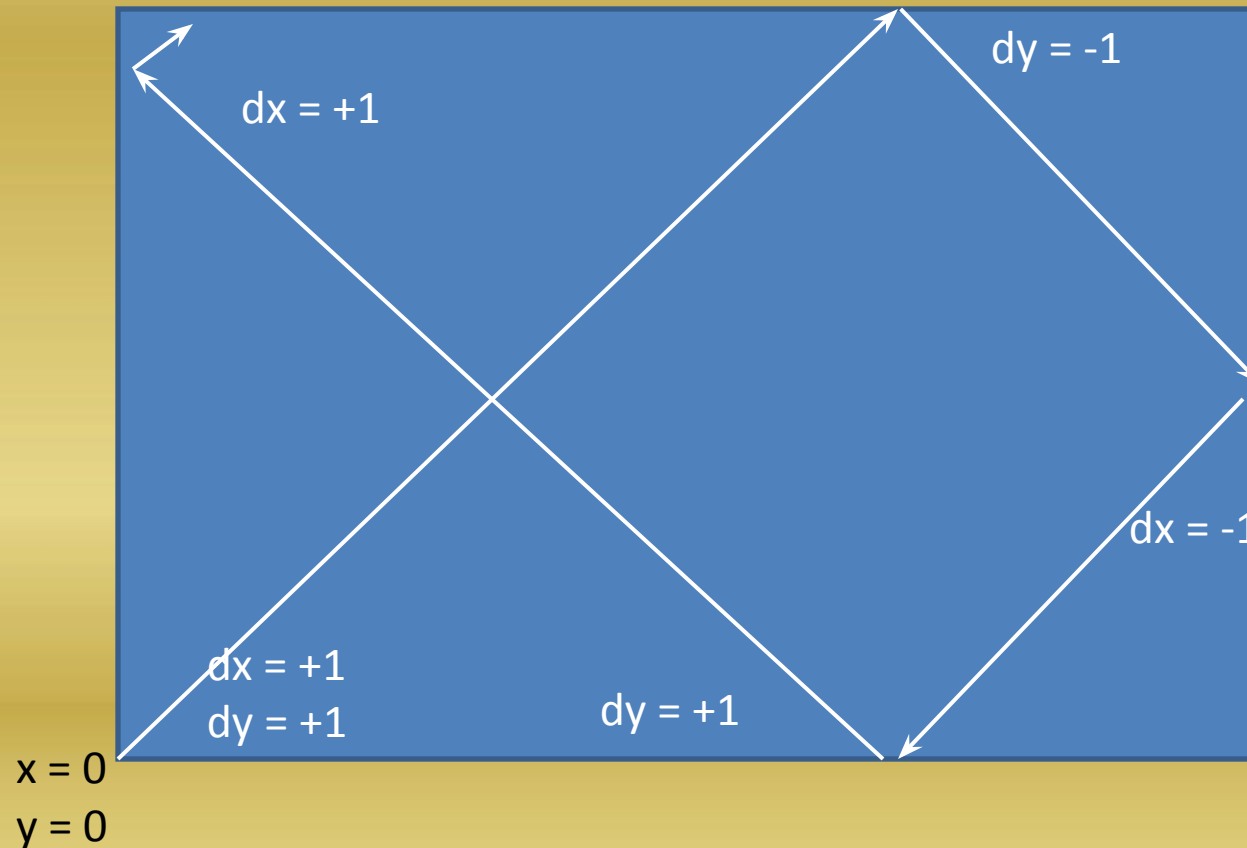
$dx = +1$
 $dy = +1$

$dy = +1$

$dy = -1$

$dx = -1$

Нам осталось рассмотреть последний вариант в движении шарика – столкновение его с левой границей. Достигнув этой границы экрана, шарик должен отскочить вверх так, как указано на схеме:



В этом случае после выполнения условия $x = 0$ координата X должна увеличиваться, т.е. $dx = +1$, а координата Y , которая до сих пор увеличивалась, по-прежнему будет увеличиваться, т.е. $dy = +1$.

Так будет выглядеть программа с добавлением четвертого условного оператора, где предусмотрено столкновение шарика с левой границей экрана:

```
SCREEN 9
COLOR 1, 0
WINDOW (0, 0)-(640, 350)
x = 1
y = 1
dx = 1
dy = 1
FOR k = 0 TO 10000
CIRCLE (x, y), 3, 6
PAINT (x, y), 6
FOR i = 1 TO 10000: NEXT i
CIRCLE (x, y), 3, 0
PAINT (x, y), 0, 0
IF y = 350 THEN dy = -1
IF x = 640 THEN dx = -1
IF y = 0 THEN dy = +1
IF x = 0 THEN dx = +1
x = x + dx
y = y + dy
NEXT k
```


Запустим программу на выполнение и вот, что получим:

$dx = +1$

$dy = -1$

$dx = -1$

$dx = +1$

$dy = +1$

$dy = +1$

$x = 0$

$y = 0$



Эту программу необходимо оптимизировать, заменив четыре условных оператора двумя. Приращения по координатам **dx** и **dy** будут менять знаки на противоположный.

```
SCREEN 9
COLOR 1, 0
WINDOW (0, 0)-(640, 350)
x = 1
y = 1
dx = 1
dy = 1
FOR k = 0 TO 10000 STEP 1
  CIRCLE (x, y), 3, 6
  PAINT (x, y), 6
  FOR i = 1 TO 4000: NEXT i
  CIRCLE (x, y), 3, 0
  PAINT (x, y), 0, 0
  IF y = 350 or y=0 THEN dy =
- dy
  IF x = 640 or x=0 THEN dx =
- dx
  x = x + dx
  y = y + dy
NEXT k
```

Итак, если мы увеличим число повторений в цикле на достаточное количество, то движение бильярдного шарика выглядит следующим образом:



...и т.д.

Далее можно усложнить эту задачу и рассмотреть несколько вариаций на тему «Движение шарика»:

- 1. «Движение шарика в заданном прямоугольнике»;**
- 2. «Движение шарика по экрану с препятствием»;**
- 3. «Муха в графине» (рисует прямоугольный графин с горлышком, располагаем внутри шарик (муху) и заставляем его двигаться внутри графина, отражаясь от его стенок. Меняя начальное расположение шарика, можно добиться, что через какое-то время муха вылетит из графина и продолжит полет, отражаясь от сторон экрана и о стенки графина, как о препятствии на пути мухи.**