ЛАБОРАТОРНАЯ РАБОТА № 9. РАЗРАБОТКА ДЕМОНСТРАЦИОННОЙ ПРОГРАММЫ ПО ФИЗИКЕ

Цель: ознакомиться с возможностями разработки демонстрационных программ по физике.

Задание 1. Отобразите движение прямоугольника по горизонтали от начала до конца экрана.

План разработки программы

1. Откройте новый проект.
2. Разместите на форме объекты Shape и BitButton (страница Additional), Timer (System).
3. Установите свойство формы Windowstate как wsMaxsimized.
4. Установите свойства Shape: Shape — stRectangle; Left — 0.
5. Установите свойства Timer: Enabled — False: Interval — 100.
6. Для таймера запишите обработку события OnTime:
   Shape1.left:=Shape1.left+10;
   if Shape1.left>780 then close.
7. Запишите для кнопки обработку события по щелчку:
   Timer1.Enabled:=True; {таймер включить}.
8. Запустите программу на выполнение.

Задание 2.

а) отобразите движение прямоугольника по окружности с центром в середине экрана и радиусом 100;

б) опишите как глобальную вещественную переменную f;

в) измените пункт 6 предыдущего упражнения:

```
x0,y0,y1,x1,r:integer; x,y:real;
x0:=140; y0:=130; r:=100;
x:=r*cos(f);
y:=r*sin(f);
x1:=x0+trunc(x);
y1:=y0+trunc(y);
shape1.left:=x1;
shape1.top:=y1;
f:=f+0.1;
if f >5*2*pi then close;
```

Задание 3. Разработайте проект «Цвета в формате RGB»

Постановка задачи

Создать программу, с помощью которой пользователь мог бы увидеть в зависимости от значений насыщенности красного, зеленого и синего цветов результирующий цвет .

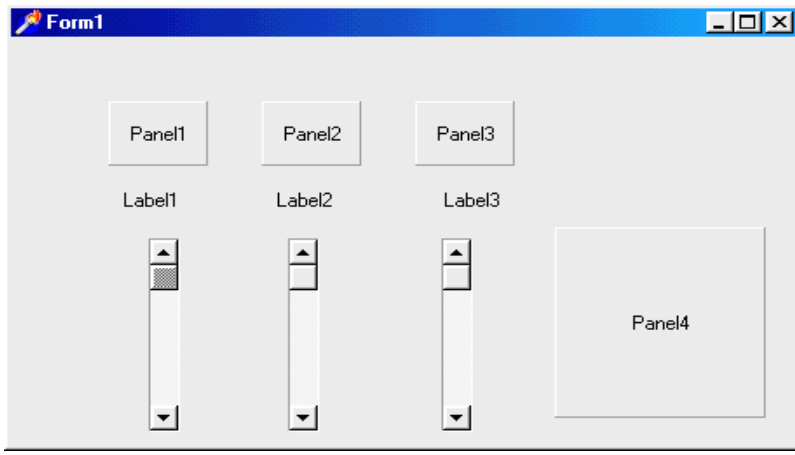

Рис. 9.1

Пояснение. Новым в этой работе будет использование:
•полос прокрутки **ScrollBar** для ввода данных;
•функции преобразования значений цветовых составляющих *TColorRef.*

План разработки программы

1. Открыть новый проект.
2. Разместить на форме экземпляры компонентов в соответствии с рисунком 9.1.

Полоса прокрутки **ScrollBar** может быть горизонтальной (по умолчанию) или вертикальной. Это определяется свойством **Kind.** В нашем случае используется вертикальная полоса прокрутки.

3. Сохранить код программы и проект.
4. Выполнить следующие действия:

ScrollBarl Properties	Name	Установка имени полосы прокрутки «RedBar», под которым компонент будет известен программе
	Max	Установка максимального количества градаций для компонентов RGB: 255
	Position	Установка начального значения: 122

Аналогично задать значения для **ScrollBar2 и ScrollBarS,** присвоив им имена «GreenBar» и «BlueBar».

6.Для всех компонентов формы установить значение свойства Caption пустым.

7.Выполнить следующие действия:

RedBar Events	OnChange	Panel1.Color := TcolorRef(RGB(RedBar.Position,0,0)); Label1.Caption := InttoStr(RedBar.Position); Panel4.Color := TcolorRef(RGB (RedBar.Position, GreenBar.Position, BlueBar.Position));

Аналогично задать значения для **ScrollBar2 и ScrollBar3,** проследить за правильностью записи параметров функций *RGB* и *IntToStr.*

Пояснение. В зависимости от передвижения ползунка **ScrollBar1** будет меняться цвет **Panel1,** выводиться числовое значение кода на месте **Label1** и меняться цвет **Panel 4.**

8.Сохранить проект, запустить и протестировать его.

Задание 4. Разработка проекта «Демонстрация по физике».

Постановка задачи

Создайте программу, изображающую оптическую систему: собирающую линзу и объект, отображаемый в линзе (рис. 9.2).

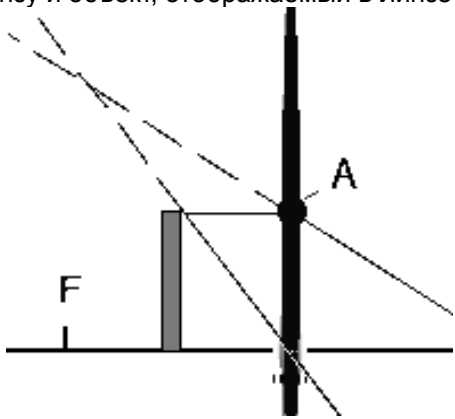

Рис. 9.2

В оптической системе отображаются два основных луча — проходящий через фокус и проходящий через центр системы. При необходимости, если изображение является мнимым, строятся и дополнительные лучи на продолжении основных.

Объект управляется с клавиатуры. При помощи клавиш влево и вправо мы передвигаем объект вдоль оси линзы. При помощи клавиш вверх и вниз изменяем размеры объекта. При этом основание объекта всегда остается на оси линзы.

При любом изменении объекта, как по положению, так и по размеру, лучи, идущие через оптическую систему, перестраиваются.

План разработки программы

1. Открываем новый проект.
2. Размещаем на форме объект ***Panel*** (страница ***Standart***). Затем объекты ***PaintBox*** (страница ***System***), затем ***Image*** (страница ***Additional***), ***Shape*** (страница ***Additional***) помещаем на Panel. Порядок размещения объектов следующий — сначала ***PaintBox***, затем ***Image*** и только потом ***Shape***. При любом другом размещении объектов могут возникнуть сложности с обработкой. ***PaintBox*** необходим нам для рисования на нем хода лучей, в ***Image*** у нас будет находиться фоновый рисунок (изображение линзы, фокусов, оси оптической системы), а ***Shape*** выступает в качестве объекта, изображение которого мы рассматриваем (на нашем рисунке — прямоугольник, от которого отходят лучи).
3. Устанавливаем для объекта ***Panel*** свойство ***Align*** (выравнивание) в ***alClient*** (для того, чтобы объект занимал полностью всю форму независимо от размера формы на экране). Теперь как бы мы ни изменяли размеры формы, ***Panel*** будет занимать полностью всю форму. Убираем надпись ***Panel1*** в центре панели.
4. Аналогично устанавливаем для нашего объекта ***PaintBox*** свойство ***Align*** в ***alClient.*** Так как ***PaintBox*** находится внутри объекта ***Panel***, то его изменения привязаны не к форме, а к панели, его содержащей.
5. Для объекта ***Image*** загружаем картинку ***linza.bmp***. Устанавливаем свойство ***Autosize*** в ***true***, чтобы наш фоновый рисунок был постоянным при переходе от одного размера формы к другому, от одного компьютера к другому или на разных разрешениях на одном компьютере. В противном случае наша картинка будет сбиваться. Свойство ***transparent*** (прозрачность) устанавливаем в ***true***. Свойства ***Left*** и ***Top*** для ***Image*** устанавливаем в 0.
6. Теперь настраиваем ***Shape***. Устанавливаем его размеры, форму, подбираем цвет. Располагаем его таким образом, чтобы он находился за фокусом и своим низом касался оси оптической системы на нашем рисунке.
7. Запускаем программу и получаем изображение, подобное рисунку (рис. 9.3).

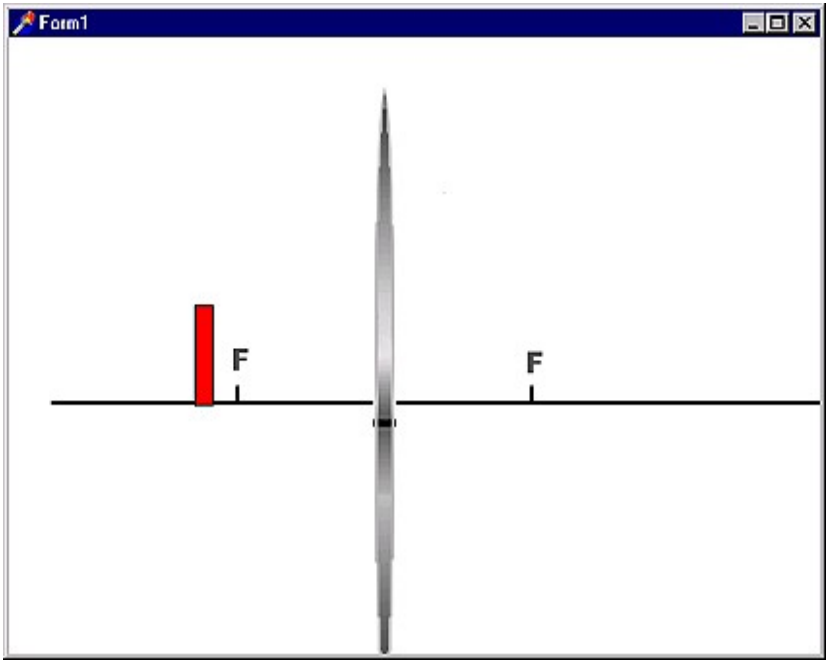

Рис. 9.3

8. Переходим к основной части — программированию рисования линий. Вводим три константы для нашей программы — две, отвечающие за центр оптической системы и третья — за фокусное расстояние ***const x0=250; y0=250; fok=120;***
9. Пишем заготовку для процедуры рисования после строчки ***{$R *.DFM}*** перед конечным ***end*** с точкой.

```
Procedure risovan;
begin
        With form1 do begin

        end; {with form}
end;
```

Блок ***With form1*** введен, чтобы указать, что мы работаем с объектами формы в нашей процедуре. Все остальное будем писать внутри этого блока.

Вводим локальные переменные для нашей процедуры:

dx, dy, fx, fy — будут выступать в качестве шага для наших лучей. ***dx, dy —*** шаг по оси X и по оси У для лучей, идущих через

центр оптической системы.

fx, fy — шаг по оси X и по оси У для лучей, идущих параллельно оптической оси и пересекающих фокус (рис. 9.4).

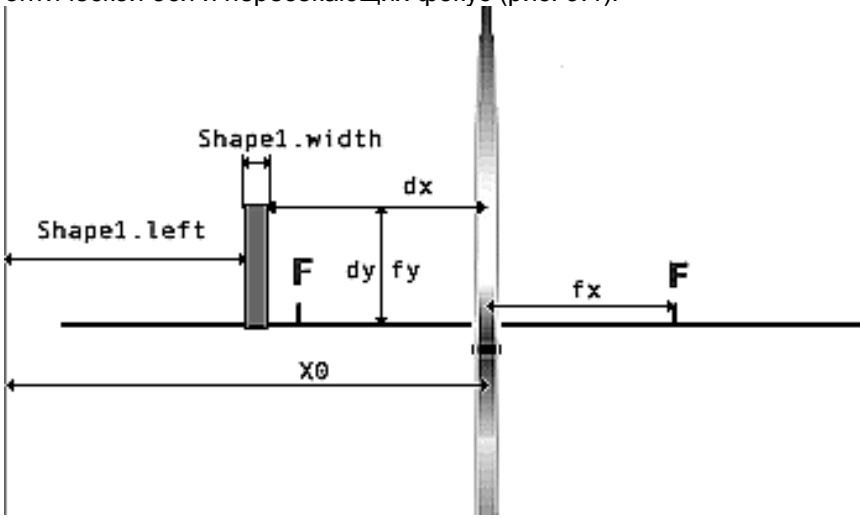

Рис. 9.4

Согласно рисунку ***dy=fy=***высоте нашего объекта (***Shape1.height***).

fx=фокусному расстоянию (нашей константе ***fok***).

dx=x0***-Shape1.left-Shape1.width.***

Записываем наши данные по относительному смещению в начале процедуры.

10. Начинаем рисовать на ***Paintbox1.Canvas***, поэтому вводим блок

    with Paintbox1.Canvas do begin

    end; { with Paintbox1}

    Сразу записываем процедуру очистки холста Paintbox1. Для этого просто рисуем заполненный прямоугольник размером с наш холст.

    FillRect(rect(0,0,PaintBox1.Width,PaintBox1.Height));//очистка

    Затем вводим процедуру перерисовки объекта ***Image:***

    Image1.Refresh;

11. Устанавливаем цвет карандаша, толщину линий 2 пикселя и стиль рисования линий ***psSolid*** (соответственно ***pen.color — clYellow, pen.Width — 2, pen.style — psSolid***).
12. Рисуем отрезок, проходящий через центр нашей оптической системы. Для этого перемещаем карандаш в правый верхний угол нашего объекта (***Shape1***) с помощью команды ***moveTo()***.

Верх нашего объекта — ***top***, правый край — ***left+width*** (т. е. получаем ***moveTo(Shape1.left+Shape1.width,Shape1.top)***).

С помощью команды ***lineTo()*** рисуем отрезок прямой до центра оптической системы (в точку ***x0, y0***).

13. Процедура будет запускаться сразу при активации формы, поэтому вызываем событие формы ***OnActivate()*** и записываем в этом событии вызов нашей процедуры ***risovan***. Запускаем программу, смотрим, как нарисовалась наша линия. При необходимости корректируем наши константы ***x0*** и ***y0*** для более точного попадания в центр оптической системы.
14. Рисуем продолжение построенного отрезка. Для этого в уже написанной строке ***lineTo()*** добавляем к координате ***x0*** смещение ***dx,*** а к координате ***y0 —*** смещение ***dy.*** При необходимости можно добавить коэффициент, только одинаковый для обеих координат (например, ***3*dx*** и ***3* dy***).
15. Рисуем отрезок прямой, проходящий параллельно оси оптической системы. Для этого перемещаем карандаш в правый верхний угол объекта (***Shape1***) с помощью команды ***moveTo()*** (аналогично пункту 12).

С помощью команды ***lineTo()*** рисуем отрезок прямой параллельно оси до линзы (в точку ***x0, shape1.top***).

Затем второй отрезок прямой, который идет к фокусу. ***lineTo(x0+fok,y0).***

16. Запускаем программу, смотрим, как нарисовался второй отрезок. При необходимости корректируем константу ***fok*** для более точного попадания в фокус.
17. Рисуем продолжение второго ключевого отрезка аналогично предыдущему, только используя константы ***fx, fy.***
18. Начинаем программирование передвижения объекта, управляя им с клавиатуры. Для этого вызываем для формы событие ***OnKeyDown***. В заголовке процедуры описаны некоторые переменные, в том числе переменная ***Key,*** которую будем использовать в своих целях. Эта переменная может принимать некоторые постоянные значения, в том числе следующие:

Vk_left — стрелка влево;
Vk_right — стрелка вправо;
Vk_down — стрелка вниз;
Vk_up — стрелка вверх.

При нажатии на кнопку влево, объект должен передвигаться на 5 пикселей влево, что можно записать следующим образом:

if key=vk_left then Shape1.Left:=Shape1.Left-5.

При нажатии на кнопку вверх, объект должен увеличиваться по высоте на 5 пикселей (***Shape1.Height:=Shape1.Height+5***), при этом низ этого объекта должен оставаться на уровне оси оптической системы, поэтому необходимо изменять еще и свойство ***top*** (***Shape1.top:=Shape1.top+5***).

Аналогично делаем смещение вправо и уменьшение размера.

В конце процедуры ***OnKeyDown*** делаем вызов процедуры ***risovan***, чтобы отрезки перерисовывались при каждом передвижении объекта.

19. Следующая задача — рисование продолжений отрезков в случае, если объект находится между фокусом и линзой. Для этого сначала проверяем условие, чтобы расстояние между объектом и линзой было меньше фокусного расстояния (т. е. ***dx<fok***). При выполнении условия меняем толщину карандаша (***pen.width***) до 1, меняем стиль рисования (***pen.Style***) на ***psDash***.

Перемещаем карандаш в правый верхний угол объекта ***Shape1*** процедурой ***moveTo()***.

Рисуем первую линию на продолжении отрезка, проходящего через центр оптической системы. Параметры для ***lineTo()*** такие же как и для основной линии, но так как рисование в обратную сторону, то ***dx*** и ***dy*** имеют отрицательный знак.

lineto(x0-3*dx,y0-3*dy);

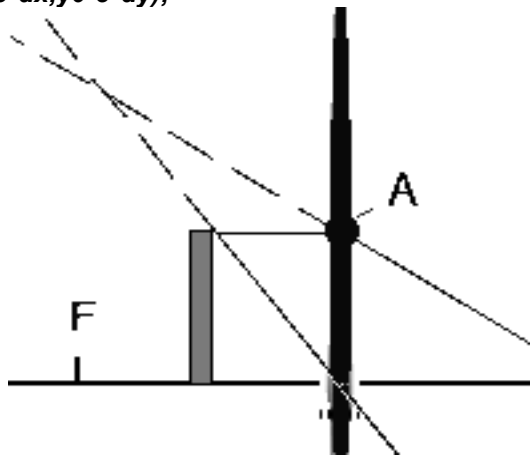

Рис. 9.5

Аналогично для продолжения линии, проходящей через фокус. Только начинаться она должна будет с точки А (рис. 9.5), поэтому необходимо перемещать карандаш в точку А и потом рисовать.

20. Запустите программу, протестируйте, при необходимости измените некоторые данные, координаты в линиях, перемещениях.

Задания для самостоятельной работы

1. Изобразите Солнце и движущиеся вокруг него две планеты.
2. В программу «RGB» внесите следующие изменения:
   2.1. поместите на форму кнопку выхода из программы;
   2.2. предусмотрите, чтобы после запуска программы устанавливались начальные цвета панелей в зависимости от исходных значений ползунков.

   Подсказка. Предусмотрите в реакции на событие **OnCreate** для формы **Forml** обработку значений **ScrollBarl, ScrollBar2 и ScrollBar3;**

   2.3. внесите в программу «RGB» изменения, чтобы значение кода цвета выводилось не только в десятичной, но и в шестнадцатеричной системе счисления. Воспользуйтесь специальной функцией InttoHex(v,n), результат которой — строковая величина, а v — величина целого типа , n — число (например, 7) возвращаемых символов в шестнадцатеричной записи.
3. Внесите в программу «Линза» изменения так, чтобы при переходе на правую сторону линзы луч, проходящий через фокус, проходил через левый фокус (необходимо добавить переменную, которая будет принимать значения 1 или –1 в зависимости от того, с какой стороны находится объект).