Урок 7. Методи створення рухомих графічних об'єктів засобами мови Паскаль

Мета: показати можливості створення рухомих графічних об'єктів та простих мультиплікаційних зображень засобами мови Паскаль на прикладах розв'язання задач, розвивати логічне мислення, творчі здібності підтримувати прагнення до засвоєння нових знань,заохочувати самостійність і нестандартність мислення, сприяти естетичному вихованню при побудові графічних зображень.

Обладнання: персональні комп'ютери, опорний конспект до заданої теми, роздатковий матеріал.

Тип уроку: урок формування умінь і навичок.

Форма організації уроку.

Робота в комп'ютерному класі - практичне заняття;

1. пояснення деяких завдань, запис програм на дошці;
2. парна та індивідуальна робота за комп'ютером.

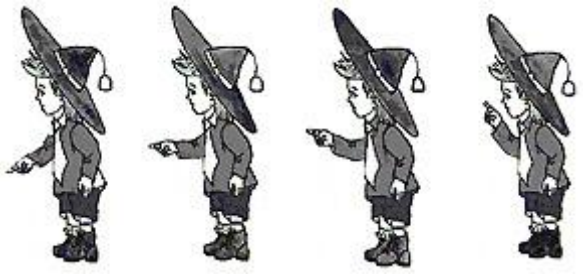
Хід уроку

1. Актуалізація опорних знань.

Перш ніж перейти до нової теми, давайте знову згадаємо цикли:

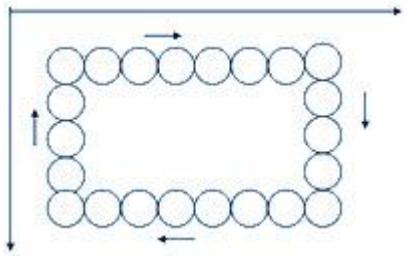
- Як виконувалось завдання про " підскакуючого хлопчика"?
- Як виконувавця рух фігурками у псевдографіці?
- Як виконувалось завдання для побудови концентричних кіл?

2. Пояснення нового матеріалу.
Щоб почати роботу над створення простішого "мультика" (зображення, що рухається), спочатку розберемося, як воно створюється в реальних умовах. Всі знають, що художник мультиплікатор малює серію зображень, в кожному з яких показується один і той же рух з ледь помітними змінами. Наприклад так, як це показано на наступному малюнку.



Тепер, якщо зображення швидко змінювати одне за одним, ми не помічаємо зміну малюнків, а бачимо рух цього персонажу (в даному випадку Незнайко піднімає руку). Поміркуємо, як відтворити послідовність схожих об'єктів на екрані монітору. Перше, що спадає на думку, це намалювати зображення, затримати його трохи на екрані, витерти зображення (очистити екран) та вивести нове зображення з ледь помітними змінами. При достатньо великій швидкості малювання око людини не помітить зміни малюнків і їй буде здаватися, що об'єкт рухається. Розв'яжемо таким методом наступну задачу.

Завдання1. Вивести на екран малюнок: рух кола по прямокутнику.



```
1) for i:=2 to 9 do
  begin
    setcolor(random(16);
    circle(50*i,100,50);
    elay(2000);
  end;

2) for i:=2 to 7 do
  begin
    setcolor(random(16);
    circle(450,50*i,50);
    delay(2000);
  end;

3) for i:=9 downto 2 do
  begin
    setcolor(random(16);
    circle(50*i,350,50);
    delay(2000);
  end;

4) for i:=7 downto 2 do
  begin
    setcolor(random(16);
    circle(100,50*i,50);
    delay(2000);
  end;
```

В даному фрагменті кожне коло виводиться по - черзі на екран і не зникає. Для того, щоб створити ілюзію руху потрібно після кожного **delay(2000)** вставити процедуру чистки екрану- **CleaDevice.**

Завдання 2. Вивести рухоме коло по діагоналі екрану, використати рівняння прямої, яка проходить через дві точки: Y= k*x+b

```
0=k*0+b
480=k*640+b
k=480/640=0.75
y=0.75*x

uses crt,graph;
  var
    dv,md:integer;
    x,y:real;
  begin
    dv:=detect;
    initgraph(dv,md,'');
    x:=25;
    y:=0.75*x;
  While x<= 615 do
```

```
  begin
    setcolor(random(25));
    circle(round(x),round(y),25);
    delay(1000);
    cleardevice
    y:=0.75*x;
    x:=x+1;
  end;
  readkey;
  closegraph;
end.
```

Завдання 3. Зобразити лінію, яка рухається з одним закріпленим кінцем.

```
uses graph, crt;
var
  driver,mode:integer;
  x,y,x1,y1,r:integer;
  kyt:real;
begin
  driver:=detect;
  initgraph(driver,mode,'');
  setcolor(yellow);
  kyt:=0;
  x:=320;
  y:=240;
  r:=100;
  while kyt<2*pi do
  begin
    x1:=round(x+r*sin(kyt));
    y1:=round(y+r*cos(kyt));
    line(x,y,x1,y1);
    delay(2000);
    kyt:=kyt+pi/16;
  end;
  readln;
  CloseGraph;
end.
```

Завдання 4.

Умова: "Годинник". Змоделювати рух годинної та хвилинної стрілок.Якщо змоделювати роботу годинника в реальному часі, то наочність програми буде невеликою, тому що рух стрілок буде ледь помітним. Тому зробимо імітацію роботи годинника, тобто хвилинна стрілка буде рухатися достатньо швидко, а рух годинникової стрілки буде залежати від хвилинної. На початку роботи з'ясуємо, з яких елементів складається годинник. По-перше, це круг з поділками, а, по-друге, два відрізка різної довжини, що імітують стрілки (стрілки можна зробити і більш складними). Круг являється нерухомим об'єктом, тому він малюється статично з абсолютними координатами центру та радіусом, а стрілки рухаються, причому переміщується тільки один кінець стрілки-відрізка, а другий теж являється статичним (центр круга). Формули, за якими обчислюються координати рухомого кінця стрілки, відомі учням з курсу математики (поворот точки на заданий кут відносно нерухомого центру з координатами $x0$, $y0$). Тому наводимо їх тут без пояснень:

$x = x0 + L*\cos a$
$y = y0 + L*\sin a$
де L - відстань, на якій знаходиться точка від центру повороту,
a - кут, на який повертається точка.

Зверніть увагу тільки на те, що в програмі друга формула замість знаку "-" буде містити знак "+", тому що екранні координати мають направленість осей, зворотну до реальних Декартових координат (на екрані значення координати *Y* збільшується в напрямку зверху вниз).
Малювання поділок на циферблаті виконується теж за допомогою вище наведених формул
Програма, що реалізує запропонований алгоритм, наведена нижче.
Зверніть увагу, що в цій програмі
L_min, L_time - довжини хвилинної та годинникової стрілок відповідно;
Color_min, Color_time - кольори хвилинної та годинникової стрілок відповідно;
R - радіус циферблату годинника;
x_centr, y_centr - координати центра екрану (визначаються у відповідності до поточної роздільної здатності за допомогою функцій getmaxx та getmaxy;
x_min, y_min - координати рухомого кінця хвилинної стрілки;
x_time, y_time - координати рухомого кінця годинникової стрілки;
Ang_min, Ang_time - кути повороту хвилинної та годинникової стрілок відповідно

Рух стрілок по циферблату здійснюється за рахунок постійного їх перемалювання то активним кольором малювання стрілки, то кольором тла ("затирання" зображення). Програма завершується після натискання будь-якої клавіші за рахунок використання циклу ***repeat until keypressed.***

```
Program Example_629;
Uses crt,graph; {Підключення бібліотек}
const L_min=174;
  L_time=145;
  Color_min=white;
  Color_time=white;
  R = 200;
var gd,gm:integer;
  S:string[2];
  x_centr, y_centr:integer;
  i,x_min,y_min:integer;
  x_time,y_time:integer;
  Ang_min,Ang_time:real;
begin
  {Ініціалізація графічного режиму}
  gd:=VGA; gm:=VGAHi;
  InitGraph (gd,gm,'egavga.bgi');
  {Визначення центра екрану}
  x_centr := getmaxx div 2;
  y_centr := getmaxy div 2;
  {Малювання статичної частини малюнку}
  SetColor(brown);
  SetFillStyle(1,brown);
  {Малювання циферблату коричневого кольору}
  FillEllipse(x_centr,y_centr,R,R);
  Ang_time:=-90;
  {Встановлення кольору малювання, стилю та вирівнювання тексту}
  SetColor(yellow);
  SetTextJustify(CenterText, CenterText);
  SetTextStyle(DefaultFont, HorizDir, 2);
  {Малювання поділок жовтого кольору та цифр}
  for i:=1 to 12 do
begin
  Ang_time:=Ang_time+30;
  x_time:=round(x_centr+185*cos(Ang_time*pi/180));
  y_time:=round(y_centr+185*sin(Ang_time*pi/180));
  str(i,S);
  OutTextXy(x_time,y_time,S);
end;
```

```
  {Малювання ходу годинника}
  Ang_min:=-90;
  Ang_time:=-90;
  repeat
  x_time:=round(x_centr+L_time*cos(Ang_time*pi/180));
  y_time:=round(y_centr+L_time*sin(Ang_time*pi/180));
  SetColor(Color_min);
  Line(x_centr,y_centr,x_time,y_time);
  x_min:=round(x_centr+L_min*cos(Ang_min*pi/180));
  y_min:=round(y_centr+L_min*sin(Ang_min*pi/180));
  SetColor(Color_min);
  Line(x_centr,y_centr,x_min,y_min);
  Delay(10000); {Затримка зображення на екрані}
  SetColor(brown);
  Line(x_centr,y_centr,x_time,y_time);
  Line(x_centr,y_centr,x_min,y_min);
  Ang_min:=Ang_min+6;
  Ang_time:=Ang_time+0.5;
  until keypressed;
  readkey;
  CloseGraph;
end.
```

3. Закріплення матеріалу. Робота за комп'ютерами

Учні виконують розглянуті завдання, модифікують їх за власним бажанням

4. Підведення підсумків

Питання до класу:
яким чином можна створити на екрані комп'ютера ілюзію руху графічного об'єкта?
Запропонований метод побудови мультиплікаційних об'єктів являється найпростішим, але якщо об'єкт, що рухається, має більші лінійні розміри, ніж в запропонований задачі, він буде суттєво миготіти на екрані. Тому існує інший підхід до розв'язку цієї задачі. В цьому випадку пропонується наступний алгоритм:

а) намалювати бажаний об'єкт;
б) запам'ятати область екрана, з виведеним малюнком;
в) відновити екран в місці, де був малюнок (тобто стерти малюнок);
г) вивести малюнок на нове місце і т.д.

Цей підхід дуже схожий на попередній варіант, але має суттєві переваги в тому, що не потребує багаторазового перемалювання малюнку. Об'єкт створюється один раз, зберігається його копія, а потім виводиться в потрібному місці.
Для зберігання намальованого фрагмента необхідно використовувати оперативну пам'ять, причому так як ми не знаємо розміри об'єкта на початку програми, пам'ять необхідно запрошувати у системи безпосередньо під час роботи програми. Це можна зробити тільки використовуючи динамічну пам'ять за допомогою підпрограм, які ми розглянемо на наступному уроці.

5. Домашнє завдання

Продовжувати працювати над власним малюнком.

Завдання 5. Вивести на екран зображення малюнка, що коливається(маятник)