Programming as a method of interdisciplinary relations in learning

Vladimír Jehlička

***Abstract*—** The presented article describes one of the potential system approaches to teaching of programming. The focus is given on the interdisciplinary interconnection between physics, informatics and mathematics. First, a simple analysis of the researched process is carried out - that means the decomposition of the light into a colour spectrum with a simultaneous arising of a rainbow on one side and, on the other side, the additive composition of colours demonstrated by lightening a white area with the help of three light sources of varied colours. This is followed by a mathematical description of the composition of colours from three basic components – red, green and blue - on the screen of the computer monitor. The numerical code of the resulting colour is expressed in the binary, hexadecimal and decimal numerical systems. The final part of the paper presents a simple program which enables using a computer monitor for an analysis of the additive composition of colours. It simultaneously also demonstrates advantages and disadvantages of expressing the numerical code of the composed colour in various numerical systems.

***Keywords*—** additive composition of colours, computer simulation of experiments, decomposition of the light, definition of colours on a computer monitor, experimental composition of colours, numerical systems.

I. Introduction

IN the last years the students´ interest in humanities has been increasing in the Czech Republic, and simultaneously the interest in classical natural sciences, as mathematics, physics or chemistry, has been declining. In courses in informatics it is not problematic to give explanations concerning the basic operations of using the most varied programs. But when a certain activity has to be supported by a mathematical calculation, students usually start being defensive, and their interest in further training vehemently declines. One of the potential causes of this situation is an isolated way of teaching mathematics as a purely theoretical subject. The time not long enough is devoted to solutions of practically aimed logical tasks; and there is no continuous emphasis on interdisciplinary relations.

II. Problem Formulation

The system approach to teaching of programming will be further demonstrated on linking a task in optics with using of numerical systems in mathematics. This way, informatics, concretely programming, creates a link between physics and mathematics. (Similar topic, i.e. using algorithm development and programming in education of physics and mathematics, can be found e.g. in [1], [2], [3], [4]).

In courses in physics (the chapter concerning Optics) students are (among other issues) made familiar with the decomposition of the light which falls from the Sun to the Earth. This light is considered as white but its decomposition results in occurrence of the colour spectrum.

In physical laboratories the light decomposition is realized with help of a glass prism [5]. However, students can consider this experiment as an artificially created situation. In that case they are to be referred to a real life, in which they may have experienced a light decomposition on rain drops, while, simultaneously, a rainbow came into being (see Fig. 1).

Fig. 1: A rainbow over a city

Fig. 2: A rainbow on a fountain

In summer months it is possible to observe a similar process of the occurrence of the rainbow also in towns, where such a decomposition of the light happens for example on water drops from fountains and waterworks onto which Sun-

rays are falling (see Fig 2).

For students who are more advanced in optics, a more detailed description of this natural event should be made available. Sun-rays which fall on individual water drops refract on the interface air – water, they go through the water, they partly bounce off from the back drop-wall (most of the drops go through the back drop-wall without a back bounce-off), they refract again on the interface air – water and they proceed to an observer's eye. At this point students are to be explained principles of the fracture of rays on the interface of an optically thinner and an optically thicker environment. It is necessary to explain when the rays refract towards the perpendicular and when, on the contrary, the rays refract off the perpendicular which is set to the level of the impingement at the point of the hit of the ray. It is obvious that students should also know the principle of the rays bouncing off the mirror, which is what happens on the back drop-wall.

The radiation of a varied wave length refracts under a varied angle on the given interfaces of environments of a varied optical thickness. And just this reality is a cause of a colour decomposition of the Sun-light. It is possible to continue with a concrete analysis and calculation of the angles formed by the rays entering into a water-drop and the rays coming out of this drop. This calculation concerns radiations of varied wave lengths, and thus of varied colours. Students should be then able to give reasons for the order of the colours in the rainbow, and they should be able to answer the question whether this order is random or, on the contrary, always the same.

Looking at the above given questions, a question concerning the radius curvature of the rainbow could arise. Is this radius always the same or does it vary? What does the value of this radius depend on? Can (in an extreme case) the radius of a rainbow be infinitely big, it means, can a rainbow have a shape of parallel lines? Or, on the contrary, can a rainbow exist in the shape of concentric circles of different colours?

It is obvious that this well-known physical phenomenon can provoke a number of interesting questions which cannot be answered without a proper analysis of the researched phenomenon. If a student then wants to seriously find the correct answers to the above given questions, then s/he cannot do that without knowledge of the principal laws of optics, the principles of mathematics and geometry. An interdisciplinary importance of solutions of this type is obvious.

Let's come back to the initially researched physical phenomenon, which means to the light decomposition. Having in mind the fact that the white light can be decomposed into a colour spectrum; we can come to a logical conclusion that the reverse process should be realizable, when the white light is re-composed from the colour spectrum.

PHYSICS – AN EXPERIMENTAL ADDITIVE COMPOSITION OF COLOURS

Within the framework of teaching of the principles of physics, students are (among other issues) made familiar with the principles of the electronic transfer of colour pictures. In case of classical screens of colour TV sets or monitors, the final colour of each point of the screen is composed of three basic colour components - the red, green and blue one.

If a teaching process at schools is to be illustrative, then the above given facts are necessary to be supported by a suitable experiment. For a demonstration of this process it is necessary to have three sources of the light radiation with the exactly defined wave lengths and of the same light intensity. This is realizable in a highly equipped laboratory environment. But within the framework of school teaching, this is a hardly solvable problem.

Even the above given example of the decomposition of the light coming out of the Sun is not as easy as it could seem at the first sight. All of us are used to the Sun light, which is considered to be white. However, if we switch on the light over the table at home, this light is considered to be white, too. But an artificial light can be realised by a lit classical bulb, an economical fluorescent tube etc. From our own experience we know that each of these sources is characterized by a certain colour of the light. Once the light is considered as "warm", then as "cold", but every time the light is considered to be white. It is clear that human eyes are very tolerant to the phenomenon of the "white light".

But in case of precise physical experiments we should and we must, above all, precisely define "white light". Theoretically, the white light is defined by temperature of radiation of absolutely black body. However, the c of the Sun radiation, which is decomposed with a glass prism, resp. which is decomposed into the form of a rainbow does not fulfil the above given definition of the white light. He radiation falling to the Earth from the Sun is generally considered as white light, but in reality it contains the spectrum of radiations with different wave lengths, which only get close to the theoretical white light.

If we want, on the contrary, demonstrate the composition of the white light from three basic colours, precisely defined wave lengths of the radiations of the individual sources [6], [7], [8] should be available. In ordinary teaching practice these assumptions are not realizable. That is why we have to accept the reality that to the students we will demonstrate experiments which more or less only get near the theoretical ideal.

The use of the Lego Mindstorms NXT 2.0 construction sets [9] can be considered as an example of simple and easily realizable experiments. A kind of a programmable construction set is concerned, which has not only several input sensors, but also output members which make e.g. a movement of the compiled programmed robots [10] possible. These construction sets are primarily intended for training of the programmers - beginners.

One of the output elements is the source of the light radiation producing a red, green or blue light. If three robots are used, when each of them radiates one of the basic colours, then it is possible to realize simple experiments with composing of colours, see Fig. 3.

Fig. 3: Lego Mindstorms robots in a fully lit experimental room

As it has been stated above, no professional experimental device is concerned - a school experiment using an easily available construction set of the Lego Mindstorms is in question. In case of these construction sets, none of the above given basic colours is precisely defined through the wave length of a particular radiation. That is why it is not possible to expect that the composition of all the three colours can lead to an ideal white light.

The basic problem is rooted already in the overall lightening of the room in which the given experiment is carried out, and also in the distance of the robots from the lit surface. In Fig. 3, which was taken in the conditions of a standard artificial lightening of the experimental room, is the resulted lightening nearly white in the whole lit surface. In Fig. 4, which was taken in the conditions of a considerably weaker lightening of the experimental room, individual colours and their composition are better visible.

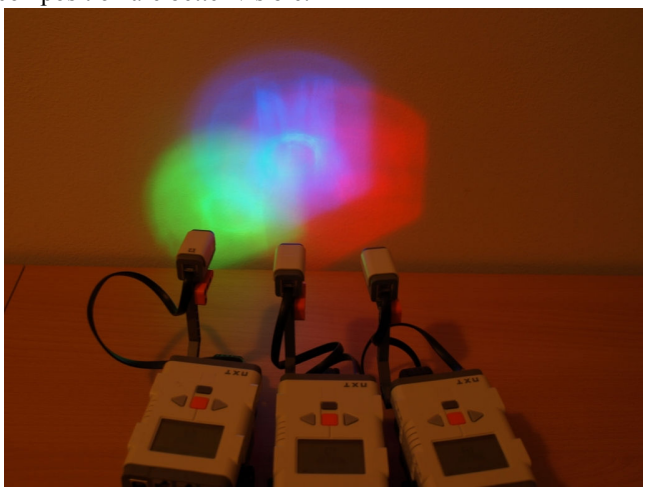

Fig. 4: Lego Mindstorms robots in a merely partly lit experimental room.

The best possible results can be reached in darkened rooms where the light rays radiated by individual robots fall onto the white surface. In Fig. 5 the composition of green and red lights is demonstrated. A yellow light comes into being in the overlapping sphere.

Similarly, Fig. 6 demonstrates the composition of red and blue lights. The overlapping sphere is lit in violet. It is necessary to keep constantly in mind that only simple school experiments are discussed, the results of which get only partly close to the ideal condition.

Other experiments, which are not demonstrated by concrete pictures in this presentation, are based on placing the individual robots in different distances from the lit surface. It is a well-known fact that the light intensity decreases with the second power of the distance between the light source and the lit surface. If the individual robots with their light sources are continuously moved, the students can be demonstrated continuous colour transitions between two selected colours, or continuous colour transitions of the three basic colours can be combined.

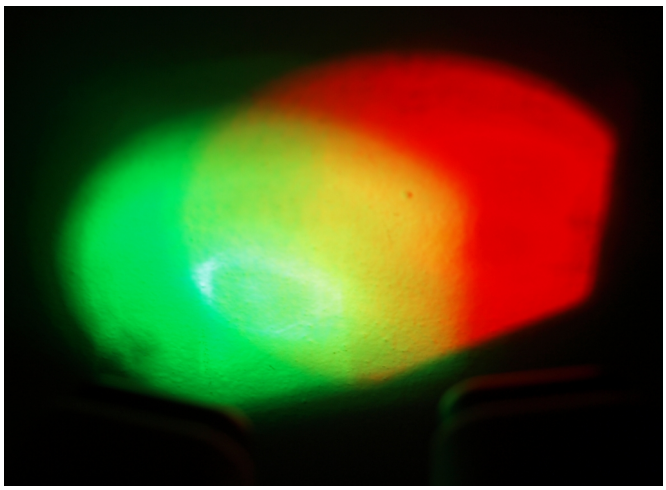

Fig. 5: Composition of green and red lights.

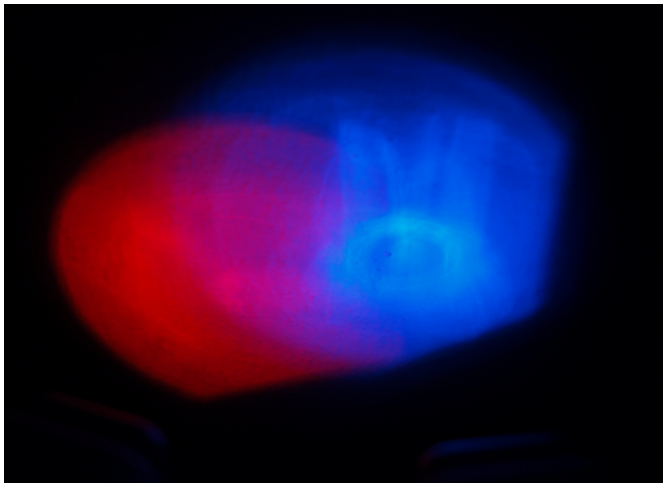

Fig. 6: Composition of red and blue lights

INFORMATICS – COMPOSING OF COLOURS ON THE COMPUTER SCREEN

Within the framework of teaching of programming, the issue

of solving of the quadratic equation is usually the content of one of the first seminars. A typical task, suitable for a presentation of a program branching, is concerned. Depending on the value of the discriminant, the solution to the quadratic equation are two real different roots or one real multiplicative root, or resp. two complexly joined roots. The algorithm thus contains simple conditions, on the basis of which the solution of the tasks makes three individual branches.

However, students´ relation to programming is negatively influenced due to a purely mathematical essence of the solved problem. On the contrary, tasks which attract students´ attention because of a practical use, as it is for example stereoscopic displaying of three-dimension objects [11], are accepted very positively.

While teaching, it is not possible to skip some needed but sometimes not enjoyable topics. A sensitive approach to students is essential, as well as a continual progress from easy tasks to more difficult ones and a constant focus on keeping students´ attention. Within the framework of the computer graphics [12] it is thus suitable to deal with the creation of colours.

On a computer monitor it is possible to realize a composition of colours in various programming languages. With respect to our purposes, the development environment Delphi has been chosen as it enables to create very well-arranged and transparent programs.

As it has been stated, each point on the screen has its own colour which is composed of three basic components: red, green and blue ones. The resulted colour is expressed with a numeral which is stored in the computer memory. For generating of this colour it is possible to use e.g. the function `rgb (Red,Green,Blue)`, where the individual parameters of the function can be of the value from 0 to 255 and they express the intensity of the given colour component. The numerical values of the three given parameters can be entered in the decimal system, so it is not essential to deal with the real way of storing the colour numbers in the computer memory. However, a deeper knowledge of the computer work, concerning the way of storing data in its memory, is useful not only for future professional programmers.

It is generally known that the smallest part of the computer memory is created by one bit, into which it is possible to write only the values zero or one. A bigger memory unit is one byte, which includes 8 bits. Into one byte it is possible to store binary numerals within the range from 00000000 to 11111111. For storing of the numeral which expresses the colour of the particular point on the computer monitor it is necessary to allocate minimally 3 bytes, see Table 1.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
blue								green								red								

Table 1: Storing of the colour number in the computer memory

III. MATHEMATICS – NUMERICAL SYSTEMS

Table 1 presents a content of a part of the computer memory, where number one is stored in all the appropriate bits. That means that all the three basic colour components have the maximal intensity, which is expressed by the numerical value of 255. The resulting colour is white. If, on the other hand, the value of zero was stored in all the bits, then all the three colour components would have the zero light intensity and the resulting colour would be black.

Let us come back to the above given value 255. Why is specifically this value given? Let us realize that for each colour component there is a limitation of one byte, i.e. 8 bits, in which just 0 or 1 can be stored. That means, into the part of the memory which is limited in this way it is possible to store maximally eight-place binary numerals. The value of the highest eight-place binary numeral expressed in the decimal system is:

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = \\ 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255 \quad (1)$$

If each of the three colour components can have the value from 0 to 255, then the given system enables to define

$$255^3 = 16\,777\,215 \quad (2)$$

colour tones in total.

The white colour can be numerically expressed in the binary system as $11111111111111111111111_2$ or in the decimal system as $16\,777\,215_{10}$. It is obvious that both the ways are not transparent because the light intensity of the individual colour components is not visible at the first sight.

The expression of numerical values in the binary system is essential in the work of computers, but it is very non-transparent for human eyes. A partial transparency is reached at the moment when a numeral from the binary system is converted into the octal, or hexadecimal system. Practically it means that in case of the octal system, three numerals of the binary expression of the number are joined into just one numeral of the octal expression.

Then in case of the hexadecimal system, four numerals of the binary entry are then joined into just one numeral of the hexadecimal system. This way, the original entry in the binary numerical system becomes more transparent. The basic conversions of the numerical values between decimal, binary, octal and hexadecimal systems are given in Table 2.

The optimal solution is to express the number of the colour in the computer memory in the hexadecimal system. Each colour component occupies 8 bits in the computer memory, which corresponds to an eight-place numeral written down in the binary numerical system or a two-place numeral expressed in the hexadecimal numerical system.

Numerical systems			
Decimal	Binary	Octal	Hexadecimal
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
...	...	...	...

Table 2: Conversion of numbers between numerical systems

The maximal intensity of the colour component is thus expressed in the given numerical systems as follows:

$$11111111_2 = FF_{16} = 255_{10} \quad (3)$$

The white colour, which is written down in the binary system in Tab. 1, can be written in a more transparent way in the hexadecimal system, see Tab. 3.

F	F	F	F	F	F
blue		Green		Red	

Table 3: Expressing of the white colour in the hexadecimal system

Colours and their hexadecimal expressing, which were discussed in the previous text, can be summarised into the following survey:

- 00 00 00 black,
- 00 00 FF red,
- 00 FF 00 green,
- FF 00 00 blue,
- 00 FF FF yellow,
- FF FF 00 violet,
- FF 00 FF cyan,
- FF FF FF white

IV. Programming – an Exemplary Program

A simple program called "Color" has been created for a transparent presentation of the additive composition of colours in the Delphi environment

While creating the program, it is first necessary to define the width and height of the basic form, and then to place the following components into it:

- image – place for drawing of graphical outputs of the program,
- scroll bar – in total three scroll bars for setting of the intensity of the lightening by individual basic colours, i.e. red, green and blue,
- label – in total,

The overall design of the placement of the above given components in the form of the final program is given in Fig 7.

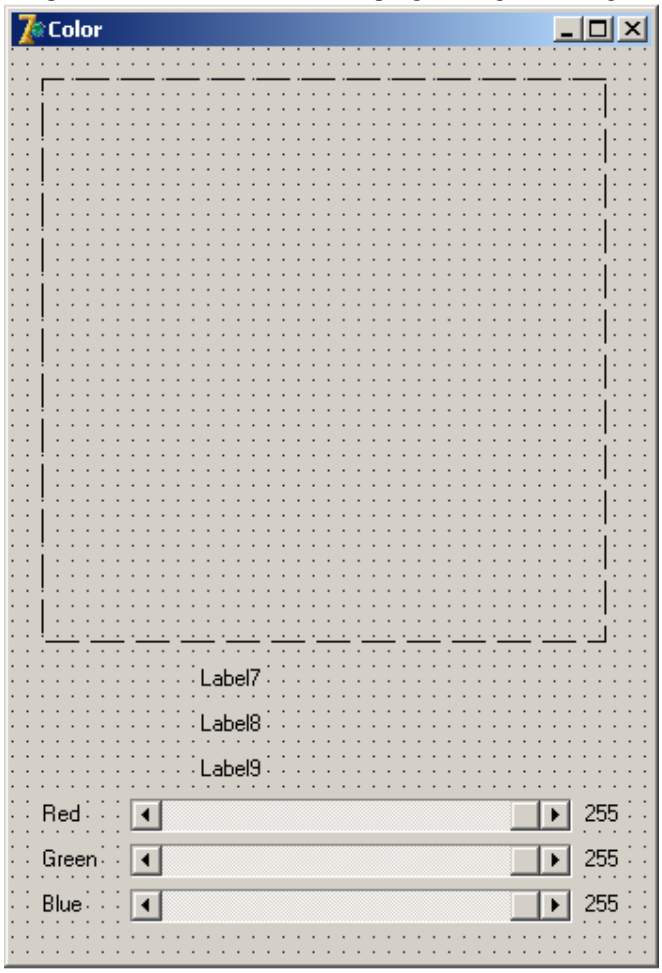

Fig. 7: Design of the form of the program

The developing environment Delphi creates automatically the basic structure of a future program. While individual components are placed into the form, the sort of the `TForm1` of the created form is automatically modified. Through the placement of the components into the form, their qualities are automatically specified, among which above all their sizes and placement are important. The programmer's task is to write and successfully prove the sub-program which will be evoked after the program is opened, and, followingly, at the time of any change of the position of any sliding box of any scroll bars.

The following print-out of the basic unit presents the concrete declaration of the type of the used form, and, above all, the sub-program `TForm1.FormCreate`, which is evoked not only when the form is opened but also as a reaction to a shift of the sliding box of any scroll bars, and assures all the activities of the program..

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils,
  Variants, Classes, Graphics,
  Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls;
const
  Hexa: array[0..15] of String[1] =
  ('0','1','2','3','4','5','6','7',
  '8','9','A','B','C','D','E','F');
type
  TForm1 = class(TForm)
    Image1: TImage;
    ScrollBar1: TScrollBar;
    ScrollBar2: TScrollBar;
    ScrollBar3: TScrollBar;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    procedure         FormCreate(Sender:
TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure      TForm1.FormCreate(Sender:
TObject);
begin
 with image1.canvas do
 begin
  brush.Color:=rgb(0,0,0);
  pen.Mode:=pmcopy;
  Rectangle(0,0,400,400);
  pen.width:=0;
  pen.Mode:=pmmerge;
  brush.Color:=
    rgb(ScrollBar1.Position,0,0);
  Ellipse(0,0,200,200);
  brush.Color:=
    rgb(0,ScrollBar2.Position,0);
  Ellipse(100,0,300,200);
  brush.Color:=
    rgb(0,0,ScrollBar3.Position);
  Ellipse(50,100,250,300);
  Label4.Caption:=
    IntToStr(ScrollBar1.Position);
  Label5.Caption:=
    IntToStr(ScrollBar2.Position);
  Label6.Caption:=
    IntToStr(ScrollBar3.Position);
  Label7.Caption:='RGB('+
    IntToStr(ScrollBar1.Position)
    +', '
    +IntToStr(ScrollBar2.Position)
    +', '
    +IntToStr(ScrollBar3.Position)
    +')';
  Label8.Caption:='$00'+
     Hexa[(ScrollBar3.Position
           div 16)]
    +Hexa[(ScrollBar3.Position
           mod 16)]
    +Hexa[(ScrollBar2.Position
           div 16)]
    +Hexa[(ScrollBar2.Position
           mod 16)]
    +Hexa[(ScrollBar1.Position
           div 16)]
    +Hexa[(ScrollBar1.Position
           mod 16)]
    +' ... hexadecimal';
  Label9.Caption:=IntToStr
    (256*256*ScrollBar3.Position
     +256*ScrollBar2.Position
     +ScrollBar1.Position)
     +' ... decimal';
 end;
end;
end.
```

If any of the above given situations come into being (i.e. opening of the form, any change of the position of sliding box of any scroll bar of any basic colour), then the subprogram `TForm1.FormCreate` first draws a black coloured square and, followingly, three mutually overlaying colour circles, see Fig. 8.

While creating a program, it is necessary to correctly set the qualities of the `image` component, the quality of the drawing pen `image.canvas.pen.mode` specifically. Implicitly it is set to `pmcopy`. If we kept this setting, then the individual colour circles would be coloured everywhere in the colour to which the brush is just set. In that case the original colour of the background would not have any impact on the resulting colour of the given area, and it would not be possible to demonstrate the composition of colours of two mutually overlaying circles.

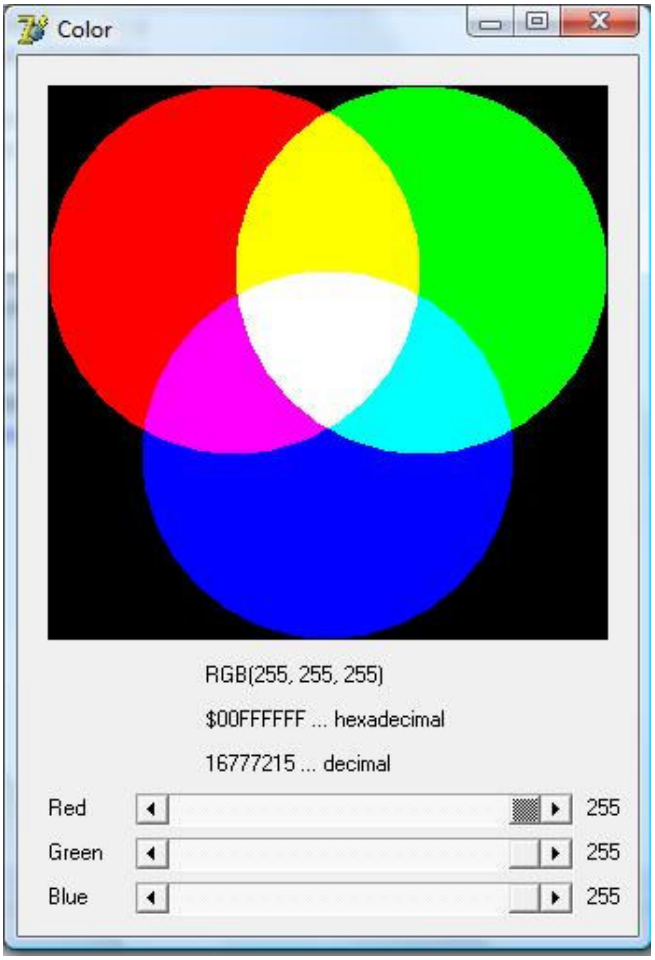

Fig. 8: Output of the Color program

To make the composition of colours possible, the `Mode` parameter of the drawing pen is necessary to be set from the original form `pen.Mode:=pmcopy;` to a new form `pen.Mode:=pmmerge;`. When individual circles are coloured, then the colour of the circle being coloured is added to the colour of the background. Thus the numerical values of the appropriate colours are added and the resulting number gives the number of the colour filling the given area. This way, the additive composition of colours is made possible on the monitor screen.

The composition of colours can be explained on the example when the yellow colour comes into being. First, a rectangle will appear in the black colour in the whole range of the image component. Then a red circle is drawn – the resulting colour comes into being through adding of the colour of the background and the colour of the new circle.

- background 00 00 00 black
- circle 00 00 FF red
- resulting colour 00 00 FF red

Followingly, a green circle appears. The summation of colours on the so far untouched background is as follows:.

- background 00 00 00 black
- circle 00 FF 00 green
- resulting colour 00 FF 00 green

At the place where both the circles are overlaying, the resulting summation of the colours is as follows:

- background 00 00 00 black
- 1st circle 00 00 FF red
- 2nd colour 00 FF 00 green
- resulting colour 00 FF FF yellow

The intensities of the composed colours can be set with help of sliding boxes on the appropriate scroll bars marked with red, green and blue colours. The current position of the sliding box is in case of each scroll bar written through a numeral in the decimal numerical system.

The area in which all the three colour circles are overlaying is coloured in the resulting summation colour. Its numerical expression is given not only individually in the RGB code, where the contributions of the individual colours are listed in the decimal system, but also as a resulting numerical value of the colour in the hexadecimal and also in the decimal numerical system.

To enter the RGB code in the decimal numerical system is very easy. In the above given printout of the unit, the Title `Label7` is composed from a chain of signs in which individual numerical values are converted through the `IntToStr` functions into particular chains of signs.

For the expression of a particular colour in the hexadecimal system it is first necessary to convert the intensity of each colour individually from the decimal numerical system into the hexadecimal numerical system. The converted number, e.g. for the blue colour, is first divided by sixteen through the integer division with the help of the operation `ScrollBar3.Position div 16. The result` The result of this integer division is the first numeral of the resulting hexadecimal entry. The second numeral is gained through the operation `ScrollBar3.Position mod 16`. The result of the operation is the remainder after the integer division, which simultaneously represents the second numeral of the resulting hexadecimal entry. The whole operation in the above given unit is described while designing the `Label8`.

The description of the resulting colour in the decimal numerical system is in the above given unit given by the Label9. The resulting numeral in the decimal numerical system is given by the summation of the value of the intensity of the blue light multiplied by 256*256, the value of the intensity of the green light multiplied by 256 and the value of the intensity of the red light.

V. Using of the "Color" program while teaching

The "Color" Program can be used as a suitable motivation for students, which supports understanding of numerical systems and mutual transformations of numerical expressions. As it has been already shown in Tab 1 and in relation (3), the expression of numbers in the binary system is hardly transparent. The conversion of numbers from the binary system into the octal system makes the work with numbers originally expressed in the binary system easier and more

transparent. The octal system was used decades of years ago, mainly while working with mainframe computers. Nowadays it is not significantly applied any more.

Then it is suitable to give a focus on a comparison of the entries in the decimal and hexadecimal numerical systems. A survey of the basic colours and their numerical expressions are given in Table 3.

Color	Decimal	Hexadecimal
Black	0	$00000000
Red	255	$000000FF
Green	65 280	$0000FF00
Blue	16 711 680	$00FF0000
White	16 777 215	$00FFFFFF

Table 3: Numerical expression of basic colours

Let us imagine a situation when we are to create a program which would simulate switching on and switching off of three colour reflectors and through that it would create continual colour transitions.

Let us start from the black colour and let us gradually increase the intensity of the red light from the minimal value of 0 to the maximal value of 255. The value of 1 will be added in the decimal system in the cycle from 1 to 255, or the value of $00000001will be added in the hexadecimal system. In both the systems the expression of colours is comparable.

Let us make the same consideration for the green colour. In that case in each step of the given cycle the value of 256 will be added in the decimal system or the value of $00000100 will be added in the hexadecimal system. The advantage of the hexadecimal system is already quite obvious here.

If the same consideration is made for the blue colour, then in each step of the given cycle the value of $256^2 = 65\,536$ will be added in the decimal system or the value of $00010000 will be added in the hexadecimal system. The advantage of the hexadecimal system is then absolutely without any discussion.

And what would be a continual transition like, e.g. from the blue to red the colour, as it is given in Fig 9.

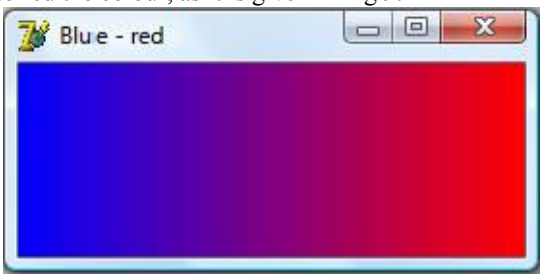

Fig. 9: Colour transition from blue to red colour

First the width and height of the form of the resulting program is decided. Then the Image1 component is inserted into the form, its size is tailored in accordance with the inner size of the form. See Fig. 10.

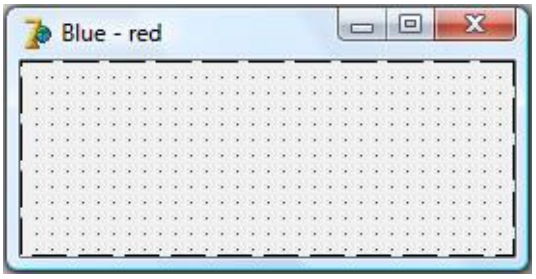

Fig. 10: Form of the Blue –red program

The numerical values of both the colours are given in Table 3. In the hexadecimal system it is started from the value of $00FF0000. The blue light is gradually reduced, that means the value of $00010000 will be subtracted. Simultaneously, the red light will be added, thus the value of $00000001 will be added. Let us suppose that the up-dated value of our colour will be stored into the particular characteristics of the drawing pen of the variable called `MyColor`. Then in each step the new value of the colour will be set through the order:

```
Pen.Color:=Pen.Color-
$00010000+$00000001;
```

If this operation is carried out 255 times in total in the cycle, then the numerical code of the resulting colour will be $000000FF; that means the resulting colour will be the required red colour.

It is obvious that in this case it is not sensible to deal with the entry of the above given order with help of the decimal numerical system although the task is solvable in any numerical system.

A print-out of the basic unit of the Blue – red Program follows:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls;
type
  TForm1 = class(TForm)
    Image1: TImage;
    procedure          FormCreate(Sender:
TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure       TForm1.FormCreate(Sender:
TObject);
var I: Integer;
begin
  with image1.Canvas do
  begin
    Pen.Color:=$00FF0000;
```

```
      for I:=0 to 255 do
      begin
        MoveTo(I,0);
        LineTo(I,100);
        Pen.Color:=Pen.Color-
$00010000+$00000001;
      end;
    end;
  end;
  end.
```

The "Color" Program can be used for demonstration of the conversion of numerals from the decimal system into the hexadecimal one. This conversion is realized with help of the integral division and calculation of the division remainder.

VI. Simulation Programs

After students master the work with colours while creating programs, they can be given tasks to create programs which enable to simulate real processes of everyday life. Simple and for students interesting tasks are represented e.g. by a simulation of mixing of cold and hot water. While running the water into a washbasin, bathtub or shower, the cold water runs when the lever battery is set to the blue mark, the hot water runs when the lever battery is set to the red mark. Analogically, on the computer monitor, cold water can be illustrated by the blue colour; hot water can be illustrated by the red colour. Besides, the resulting colour of the mixed water can be illustrated by the colour which is created by the transition between the blue and red colour, the colour of the mixed water corresponds to its temperature.

To make it simple, let us suppose that the blue colour will illustrate the water of temperature 0 °C, the red colour will illustrate the water of temperature 100 °C. Fig. 11 illustrates the output of the program which enables to illustrate a rectangle area whose colour corresponds to the current water temperature. The sliding box of the scroll bar enables to set the required water temperature. The currently set water temperature is illustrated also by the colour corresponding to the water temperature. All the numerical values are given in °C.

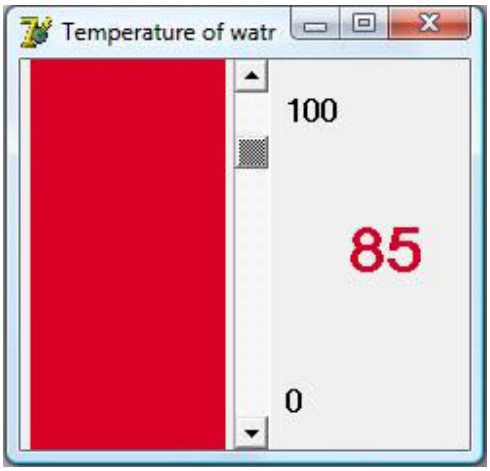

Fig. 11: Simulation of heating of water

First the width and height of the form of the resulting program is decided. Then these components are placed into the form. See Fig. 12:

- Image for a colour expression of the water temperature,
- ScrollBar for setting of the water temperature,
- Label for pro setting the lower and upper limit of the temperatures which are set, and for displaying the resulting temperature.

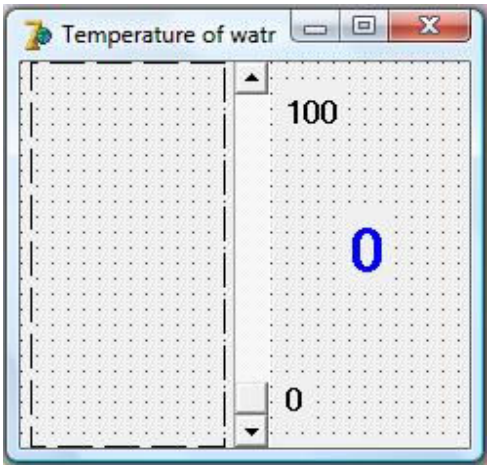

Fig. 12 Form of the program Temperature of water

The printout of the unit of the program Temperature of water follows. For setting of the colour of the mixed water the RGB function is used for change.

```
  unit Unit1;
  interface
  uses
    Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ExtCtrls;
  type
    TForm1 = class(TForm)
      Image1: TImage;
      ScrollBar1: TScrollBar;
      Label1: TLabel;
      Label2: TLabel;
      Label3: TLabel;
      procedure          FormCreate(Sender:
TObject);
      procedure    ScrollBar1Change(Sender:
TObject);
    private
      { Private declarations }
    public
      { Public declarations }
    end;

  var
    Form1: TForm1;
  implementation
```

```
  {$R *.dfm}
  procedure       TForm1.FormCreate(Sender:
TObject);
  begin
    with Image1.Canvas do
    begin
      Pen.Color:=clBlue;
      Brush.Color:=clBlue;
      Rectangle(0,0,100,200);
    end;
  end;
  procedure
TForm1.ScrollBar1Change(Sender: TObject);
  var MyColor: TColor;
  begin
    with Image1.Canvas do
    begin
      MyColor:=RGB(Round(255*(100-
ScrollBar1.Position)/100),0,
  Round(255*(ScrollBar1.Position)/100));
      Pen.Color:=MyColor;
      Brush.Color:=MyColor;
      Rectangle(0,0,100,200);
    end;
    Label3.Caption:=IntToStr(100-
ScrollBar1.Position);
    Label3.Font.Color:=MyColor;
  end;
  end.
```

VII. Conclusion

The system approach to the way of teaching of programming has been proved to be worth realizing. The emphasis on the interdisciplinary relations between individual subjects has been proved as increasing students´ attention and, simultaneously, also their motivation for their independent activities.

The privileged position of informatics is given by the fact that, for example, within the framework of programming it enables to interconnect an experimental teaching subject (e.g. physics) with a purely theoretical mathematics. On the given example it is possible to demonstrate the use of a relatively arid issue of numerical systems while practically creating simple programs with created colour definitions. Manifold varieties of simple school tasks linked with the given issue are quite fruitful and they have been accepted in a very positive way by students.

References

[1] S. Hubalovsky, E. Milkova, P. Prazak P, "Modeling of a Real Situation as a Method of the Algorithmic Thinking Development and Recursively Given Sequences", *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS*, Issue 8, Volume 7, August 2010, pp.1090-1100

[2] S. Hubalovsky, E. Milkova, "Modelling of a real situation as a method of the algorithmic thinking development", In: *Advanced Educational Technologies, Proceedings of 6th WSEAS/IASME International Conference on Educational Technologies (EDUTE'10)*, WSEAS Press, Kantoui, Sousse, Tunisia, May 3-6, 2010, pp. 68–72

[3] S. Hubalovsky, "The system approach to *teaching* of algorithm development", in *Proc. WSEAS/IASME International Conference on Applied Computing (ACC'10)*, WSEAS Press, Timisoara, Romania, 2010, pp. 22-27.

[4] M. Musilek, S. Hubalovsky, "Cryptoanalysis as a Method of the System *Approach* in the Algorithm Development", in *Proc. WSEAS/IASME International Conference on Applied Computing (ACC'10)*, WSEAS Press, Timisoara, Romania, 2010, pp. 16-21.

[5] (The Joy of Visual Perception) [online]. c2009 [cit. 2010-10-11]. Newton's Experiment. Available: <http://www.yorku.ca/eye/newton.htm>.

[6] (Wikipedia) [online]. [cit. 2010-10-11]. *Additive color*. Available: <http://en.wikipedia.org/wiki/Additive_color>.

[7] (The Joy of Visual Perception) [online]. c2009 [cit. 2010-10-11]. *Additive Color Mixing of Light*. Available: <http://www.yorku.ca/eye/3color.htm>.

[8] (American WideScreen Museum) [online]. c1998 [cit. 2010-10-11]. Additive and Subtractive Color Systems Explained. Available in WWW: <http://www.widescreenmuseum.com/oldcolor/additive-subtractive.htm>.

[9] *Lego Mindstorms manual*. Available: http://mindstorms.lego.com/en-us/Default.aspx.

[10] L. Stirb, Z. Marian, M. Oltean, *An autonomous approach to wheel changing problem*, Studia univ. Babes-Bolyai, Informatica, Vol. LV, No. 1, 2010, pp. 41-50.

[11] S. Hubalovsky, A. Hubalovsky, "*Utilization of ICT in creation and modeling of the stereoscopic picture*", Media4u Magazine. No. 1, 2010, pp. 73-76. Available: www.media4u.cz (in Czech).

[12] H. Kohout, "*Use of colours in computer graphics*". Media and education 2009, Media4u Magazine. No. X6, 2009, pp. 52-54. Available: www.media4u.cz (in Czech).

Vladimír Jehlička was born in Pardubice, Czech Republic in 1951. In 1975 he received degree Ing. in the Processes, equipment and automation of chemical processes at the University of Pardubice, in 1981 he received degree Ph.D. in the Technical cybernetics at the University of Pardubice, in 1998 he became associate professor in the Automation of machines and technological equipment at the Technical University in Ostrava.

Since 1975 he worked as an assistant professor at the University of Pardubice, since 1998 worked as a associate professor at the University of Hradec Kralove, where since 2005 he is Dean of the Faculty of Education.

Doc. Ing. Vladimír Jehlička, CSc. is a member of the Department of informatics, Faculty of Science, University of Hradec Kralove, where he focuses on programming, modeling and simulations of real systems.