Robert Nowotniak[1], Cezary Draus[1], Maciej Nowak[1], Grzegorz Rybak[1]

Modelling Reality in Visual Python

In summer semester of the academic year 2010/2011, the course "Programming in Scripting Languages" was given in Computer Engineering Department. In the course laboratories, Python programming language was used, and the attending students have created various projects in Visual Python. The students projects were created in partial fulfilment of the requirements for receiving a credit for the laboratories. Based on the students projects, the present paper concerns modelling reality, visualizing real-world phenomena and writing simple educational games in Visual Python[3] (VPython).

The general aim of science is to describe, explain and predict the behaviour of the world around us. However, reality is usually much too complex to be described accurately, without any simplification or approximation. Fortunately, computer modelling of reality[4] can shed light on rules governing the behaviour of complex objects. In the paper, special attention has been stressed to possibilities of Visual Python use in education and other areas of knowledge and science. It includes new approaches to teaching, evaluation and assessment. During the course laboratories, several interesting students projects have been created. The projects concerns various exciting real-world phenomena, including:

- **Neural Networks[6] –** visualization of the networks structure
- **L-Systems[5,7]** – modelling the growth processes of plants development
- **Electrostatic fields visualization** – educational game
- **Waves propagation** – reflection and interference phenomena
- **Fuzzy controller[6,9]** – vehicles overtaking on a two-way road
- **Robotic arm** – controller with arbitrary degrees of freedom

This paper is organized as follows. In Section 1, basics of Visual Python has been briefly introduced. In Section 2, selected students projects have been presented with more details. In Section 3, the paper has been briefly summarized. Additionally, on the CD accompanying the conference proceedings, supplementary materials have been provided (additional screenshots, projects source codes and animations), greatly enhancing the pedagogical value and enjoyment of learning.

[1] Computer Engineering Department, Technical University of Lodz, Poland

1. Visual Python – Introduction

Python[2] is a modern scripting programming language, and VPython[1,3] is a 3D graphics module which allows creating interactive animations easily. VPython allows users to create objects such as spheres and cones in 3D space and to display these objects. As a side effect of computations, real-time, navigable 3D animations are generated automatically. Thus, it is easy to create simple visualizations, allowing programmers to focus more on the computational aspect of their programs. **The simplicity of VPython has made it a great tool for the illustration of simple physics, particularly in the educational environment.** In VPython, it is easy to create interactive 3D displays and animations, even for people with limited programming experience. Also, due to the nature of Python programming language, it has much to offer for experienced programmers and researchers.

A very simple program in Visual Python has been presented below. Running the code displays a green ball and a red box in 3D space, which has been presented in Figure 1. It is possible to navigating in the space with mouse, holding right mouse button, and to zoom in / zoom out with middle mouse button.

```
from visual import *
redbox = box(pos=vector(4,2,3), size=(8,4,6), color=color.red)
ball = sphere(pos=vector(4,7,3), radius=2, color=color.green)
```

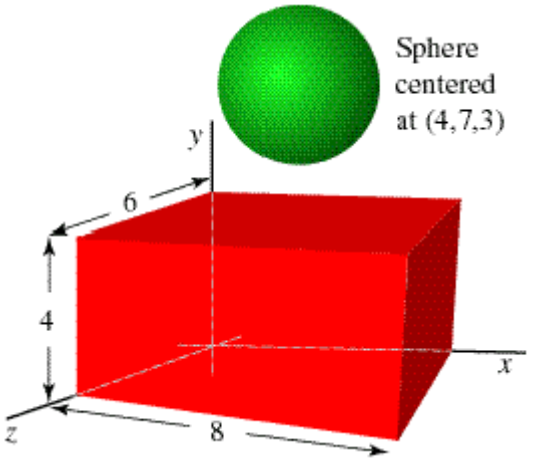


Figure 1. Sphere and box in Visual Python 3d space

Attributes (e.g. position, colour, rotation) of existing objects in VPython can be modified as follows:

```
redbox.pos = (-5, 5, 5)
redbox.opacity = 0.5
```

Similarly, simple animations can be created by modifying the objects attributes in loops, such as:

```
while True:
    rate(100)
    redbox.pos += (0.03, 0, 0)
    redbox.rotate(angle=pi/100)
```

In result, the box object on the screen moves and rotates simultaneously (calling the rate() function is required to keep the animation speed accurately). Moreover, Visual Python library offers a variety of geometric solids, user interface components and user interaction procedures. Selected geometric solids have been presented in Figure 2.

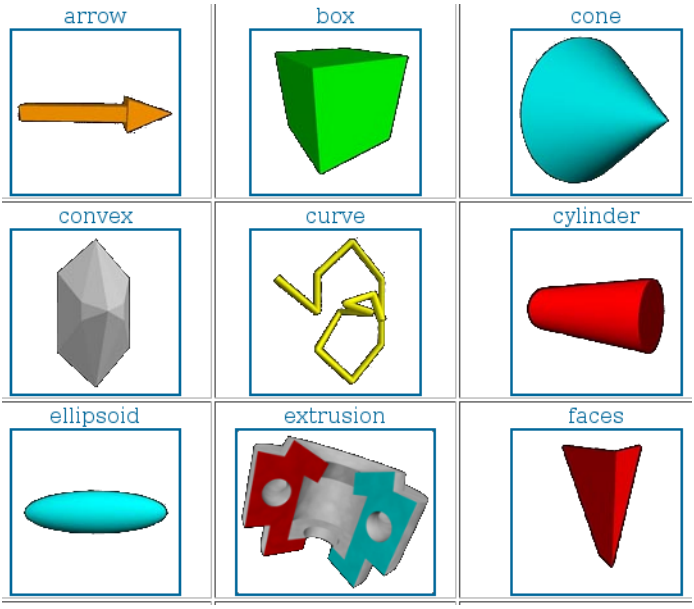


Figure 2. VPython 3d Objects

2. Selected Projects

2.2. Electrostatic Game

The aim of this project was to create a game which helps children to understand the physics of electrostatic phenomena. The author of this project is Maciej Nowak, and the project presents clearly possibilities of computer games applications in education. The game screen has been presented in Figure 3.

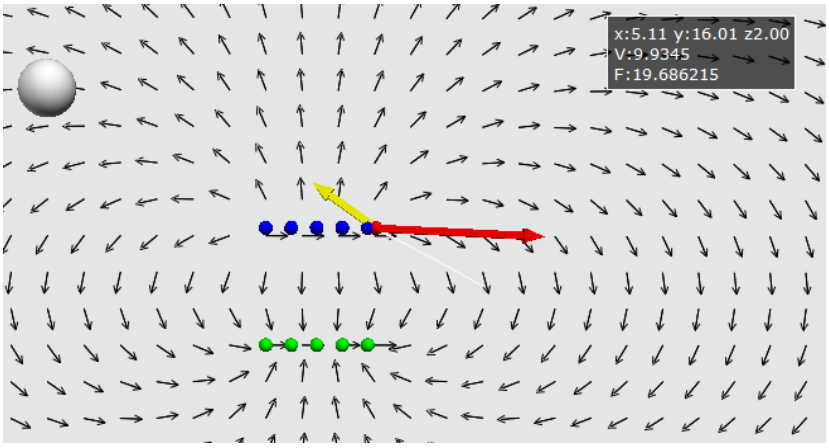
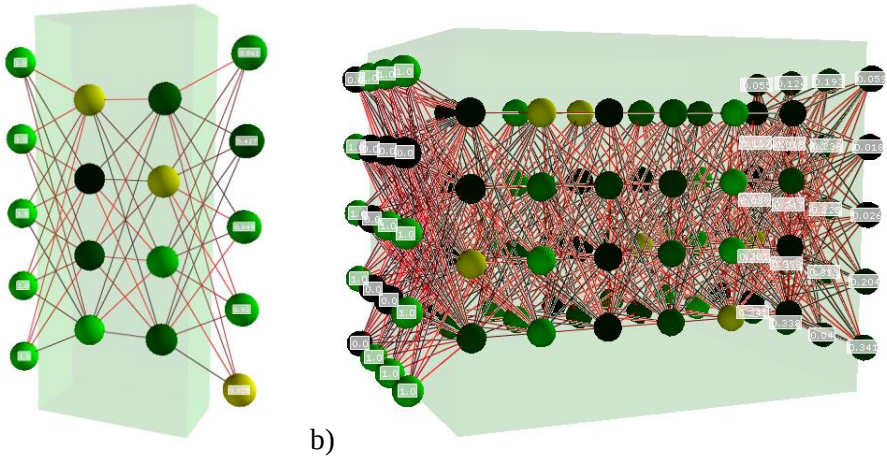


Figure 3. Screenshot from the game. The electrostatic field is visualised with arrows.

The player is supposed to hit the white target ball with lightweight red particle. To achieve this, the player must enter the aiming mode by pressing *Ctrl+A*, and then aim by setting the initial velocity and direction with mouse click. However, some game levels contain obstacles, which are unmoving, stationary charges. They change the bearing of the particle due to the electrostatic force. Therefore, to hit the target, player has to add new stationary charges which would alter the bearing, compensating for the influence of the obstacles. Adding charges is possible in the adding mode, activated with *Ctrl+N* key combination. Moreover, player may change the sign of the added charges with *"+"* and *"-"* keys, as well as increase or decrease the total charge with *"up"* and *"down"* arrow keys. To facilitate aiming and making corrections, the electrostatic field is visualized with the help of thin arrows. Moreover, moving particle leaves a trail. Positive charges are coloured blue, whereas negative are green. Moving particle is always red, which allows easy recognition. The attraction force is depicted with thick orange arrow, and velocity with magenta arrow. **The presented game may be used as a didactic aid to diversify physics lessons, as well as encouragement to learn physics with fun.**

2.1. Neural Network Visualization

This projects has been created by Grzegorz Rybak. Aim of the project was to create an application that allows visualization of neural networks structures. Special attention has been stressed to Self-Organizing Maps (Kohonen's networks[8]).



a) b)

Figure 4. Examples of neural networks structures

Final results of the project have been presented in Figure 4. Connections, weights and neurons output values are depicted with different colours. It is possible to insert arbitrary number of layers and arbitrary number of neurons in each layer. Also, it is possible to display labels with values in the input and output layers of the neural network. In the beginning of the program, input values are loaded from the provided input file. The algorithm simulating artificial neural network starts, when the user presses "c" button on the keyboard. In the end of the simulation, the values in the output layer are set accordingly to Kohenen's algorithm. In the project source code, reflection mechanism provided in Python programming language has been used intensively. This modern programming technique allowed to reduce length of the program source code considerably.

Due to the nature of scripting language, it is ease to introduce further extensions to the application. Possible further development may include changes in the input interface, algorithms, or different activation functions.

2.3. Growth processes of plants development

Growth processes of plants development has been implemented in two projects, created by the students, Michał Łojanowski and Piotr Leszczyński.

Lindenmayer system[5] (L-System) is a parallel rewriting system, namely a variant of a formal grammar, most famously used to model the growth processes of plant development. L-systems can also be used to generate self-similar fractals. L-systems are due to the Hungarian theoretical biologist and botanist from the University of Utrecht, Aristid Lindenmayer, and they were introduced and developed in 1968.

In Figure 5, examples of artificial plants have been presented. In the project, several extensions to classical L-System have been considered. For example, L-Systems with random factors have been implemented which have been presented in Figure 6.

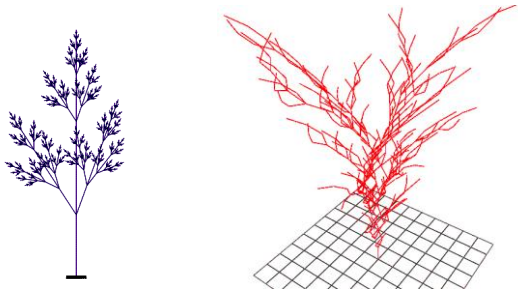


Figure 5. Simple L-Systems

Figure 6. Examples of L-Systems with random factors

2.4 Two-dimensional waves propagation

The aim of this project was to visualize 2d waves propagation. The project author is Cezary Draus, and the project has been created in a typical Problem-Based Learning (PBL) approach. The author considered several possible approaches to solving the problem. In the beginning, approach based on regular discrete cellular automata has been taken. However, this approach turned out to be too computationally intensive and time-consuming.

The application is focused on the phenomena of reflection and interference of waves. Source of vibration is located in the middle of square medium, limited by four walls. In the beginning, the source stimulates neighbouring particles, they stimulate their neighbours and so on. Stimulating particles lie on a circle with a centres of source and a radius of one greater than the moment prior to. When the wave forehead touches the wall, reflection is created. Reflection is modelled as a mirror source of vibration with axis on the reflecting wall. As a result, the points are stimulated by main source and sources which are formed by reflection. Tilt of the particle is calculated as the sum of the component deflections. As in acoustics wave, amplitude decreases in direct proportion to distance from its source. Oscillation amplitude is computed on basis of below formula:

$$A_i = A_k \times b \times d_{ik}$$

where:

A_i – Oscillation amplitude for point *i*

A_k – Source k basic amplitude

b – proportionality factor

d_{ik} – distance between source k and point i

While particle can be stimulated by more than one oscillation source, to compute output deflection, the presented formula is used:

$$y_i = \sum_{j=1}^{n} A_{ij} \sin(\omega t + \sigma_{ij})$$

y_i – output deflection

n – number of oscillations

A_{ij} – Amplitude for oscillation j and point i

σ – start phase for point i and oscillation j

ω – pulsation of oscillation

t - time

In the project, several considerable simplifications has been made. The implemented model assumes that source is stable and generates wave all the time. Therefore, once stimulated point vibrates till the end of the program. Also, addi-

tionally reflected waves do not reflects more, because their impact is negligible. In Figures 7 and 8, reflection and interference phenomenon has been presented, respectively. Equilibrium position of the medium is grey coloured. Bigger positive deflection is whiter, and bigger negative is darker.

Figure 7. Simple wave reflection simulation

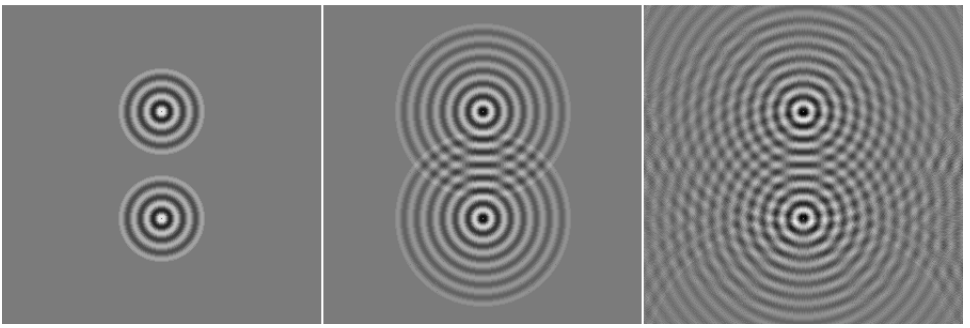


Figure 8. Interference of two waves

2.5 Other projects

During the course "Programming in Scripting Languages", several other interesting projects have been also created, such as (projects and their authors, respectively):

1. **Robotic arm controller** – Marcin Tokarski
2. **Fuzzy controller for overtaking vehicles** – Bartosz Koziak
3. **Maze generation and solving** – Jakub Kimmer.
4. **Billiard table** – Tomasz Zieliński

Screenshots of the projects have been presented in Figure 9.

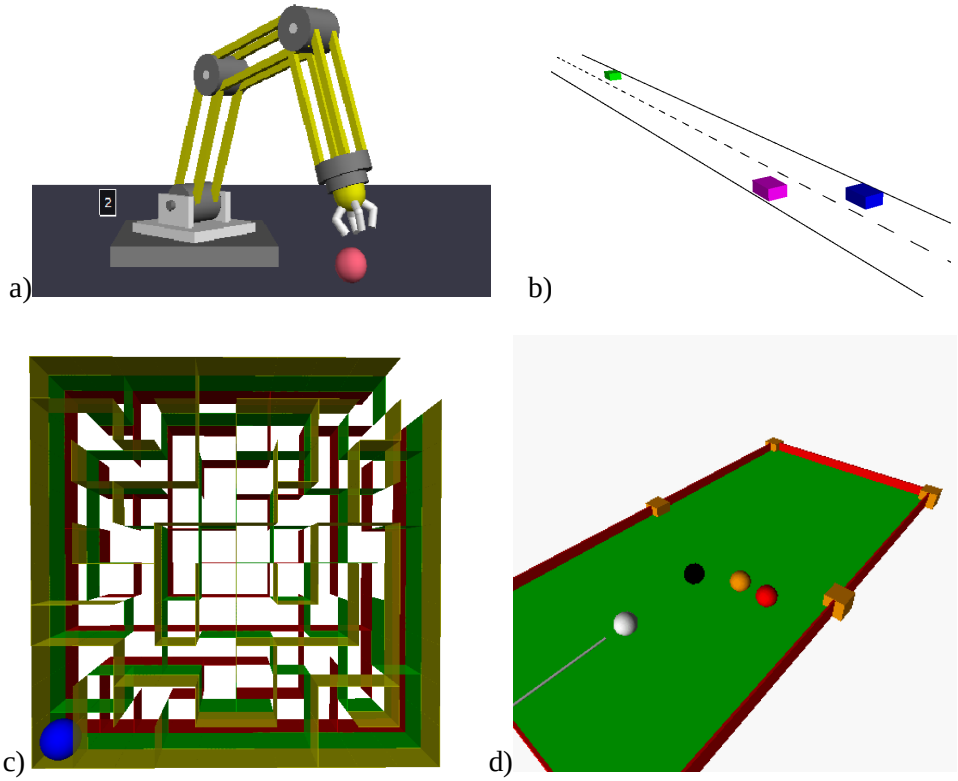


Figure 9. Screenshot of the other projects in Visual Python

3. Conclusions

In this paper, it has been demonstrated that Visual Python is a great tool for creating simple physics visualizations, especially in the educational environment. Possible use of Visual Python in education and other areas of knowledge and science have been presented. Visual Python is recommended especially for academic teachers giving laboratories in applied physics to model physical phenomena and creating computer simulations. It has been presented that in Visual Python it is easy to write impressive interactive three-dimensional animations. It is particularly useful to make nearly every physical phenomena visualizations, simulations or simple computer games which may be useful to support interactive education.

Acknowledgements

The author Robert Nowotniak is a scholarship holder of project entitled "*Innovative education* ..." supported by European Social Fund.

4. References

[1] "VPython." Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. Retrieved 13 June, 2011

[2] Beazley D. M., Python Essential Reference, Sams Publishing, 2006

[3] VPython, http://www.vpython.org/, Retrieved 13 June, 2011

[4] Białynicki-Birula I., Białynicka-Birula I., Modelling Reality. How computers mirror life, Oxford University Press, 2004, ISBN: 0198531001

[5] Prusinkiewicz P., Lindenmayer A., The Algorithmic Beauty of Plants, Springer-Verlag, 1990, pp. 101-107

[6] Rutkowski L., Metody i techniki sztucznej inteligencji, Wydawnictwo Naukowe PWN, 2006

[7] Rozenberg G., Salomaa A., The mathematical theory of L systems, Academic Press, New York, 1980, ISBN 0-12-597140-0

[8] Kohonen T., The self-organizing map, Proceedings of the IEEE, 1990

[9] Zadeh L. A., Fuzzy sets, Information and Control 8 (3) 338-353.