TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics and Mechanics
Vol. 55, Issue II, 2012

OBJECT ORIENTED PROGRAMMING (OOP) BEHAVIOR OF PHYSICS PARTICLES

Dan Anastasiu POP, Gabriela Raluca MOCANU, George ARGHIR

Abstract: *The aim of this paper is to propose a new paradigm to explain the intrinsic mechanisms behind physical laws. Object Oriented Programming (OOP) behavior is a new approach to the way interactions get priority and to the way space time stores data. We propose this in the framework of digital physics but add as an original view the OOP behavior.*

Key words*: Object Oriented Programming.*

1. INTRODUCTION

Regarding the meaning of what "digital physics" stands for it seems that there are two major directions currently being researched. The first refers to it as being a collection of theoretical perspectives that start by assuming that the universe is, at heart, describable by information, and is therefore computable. Given such assumptions, the universe can be conceived as either the output of some computer program or as being some sort of vast digital computation device (or, at least, mathematically isomorphic to such a device). The second explores the meaning of the word "digital" in the sense that the fact that space or time are digitized is of major importance [1].

In the next sections we will propose a mechanism which reflects the way this huge computing space leads to observed manifestations of physical laws and more precisely to the interaction between, lets say, electron and electron.

In order to solve the computing space paradigm we propose using Object Oriented Programming (OOP) concepts, such as classes and objects. A Class is a user defined data type which contains the variables, properties and methods in it. A class defines the abstract characteristics of a thing (object), including its characteristics (its attributes, fields or properties) and the thing's behaviors (the things it can do, or methods, operations or features). One might say that a class is a blueprint of factory that describes the nature of something [1, 2]. It is obvious just by reading this definition that OOP fits our purpose best. OOP concepts were used before as a tool in modeling elementary particles interactions [3], but what we propose is different in the sense that OOP is not a tool but a cause. One of the direct consequences of this change in perspective is that if there is an OOP concept that does not seem to lead to a physical/observable counterpart we expect that this counterpart will be discovered at some point in the future. Following this train of thought, elementary particles are the observable counterparts of OOP objects, i.e. of instantiations of classes at run time.

Our idea that physics is driven by behavior of informatics objects was pioneered by Zuse [4], which was limited by the reduced programming tools of his era to give easy examples and great ideas.

The context in which we propose this paradigm may very well be described by Feynman's thoughts, which were most likely

inspired by his PhD advisor, Wheeler, author of "It from bit", and we quote him here: "It always bothers me that, according to the laws as we understand them today, it takes a computing machine an infinite number of logical operations to figure out what goes on in no matter how tiny a region of space, and no matter how tiny a region of time. How can all that be going on in that tiny space? Why should it take an infinite amount of logic to figure out what one tiny piece of space/time is going to do?". With this is mind, the next quote from Feynman opens the way to discussing new responsible mechanisms for the existence of the real world: "So I have often made the hypothesis that ultimately physics will not require a mathematical statement, that in the end the machinery will be revealed, and the laws will turn out to be simple, like the checker board with all its apparent complexities. But this speculation is of the same nature as those other people make— 'I like it,' 'I don't like it'— and **it is not good to be prejudiced about these things**".

2. DISSCUSSION OF ROUTINES

It is clear that however this mechanism works it has to provide results that agree with observational data. There could be two ways to account for observed real motion, namely the physical address of a particle and this leads to what we perceive as particle kinematics or one of the variables hosted in the physical address modifies leading to the same result.

Since the calculating space is not identical to the real space, we choose the second paradigm: objects (particular instantiations of general classes) "live" in memory addresses that do not change during interactions. Their observed motion through real space is given by modifications of the variable (**SpatialCoord**) "position" within the object.

We give some Java programming samples in order to illustrate how OOP behavior leads to observable charged particle interaction, with special emphasis on OOP concepts used. This is strictly so that we can emphasis the idea of OOP **driving** physics and not for programming use. A very good integration of OOP programming in **simulating** physical phenomena has already been proposed by [3]. The routines are given below, and each line is accompanied by a short comment to emphasis the underlying relation between OOP concepts and real-world manifestations of these concepts.

```
public interface Particle

{
/* electrostatic interaction between
particles includes modification of
coordinates due to interaction */

public double ElectrostaticInteraction
(ElectricallyCharged otherParticle);

/* method 1 (), standing, for example,
for the Strong Interaction; */

/* method 2 ()
.
.
.
method n () */
}
```

```
public class ElectricallyCharged
implements Particle

{
/* cannot modify charge of the
particle (encapsulation) */
private int ECharge;

/* can modify spatial address of the
particle, following interaction assume
that the motion is one dimensional */
public double ECoord;

/* cannot modify mass of the particle
(encapsulation) */
private double EMass;

/* cannot modify coupling constant of
the interaction (encapsulation) */
private double coupling = 1;

public ElectricallyCharged(int charge,
double spatialCoord, double mass)
  {
    ECharge = charge;
    ECoord = spatialCoord;
    EMass = mass;
  }

public double
ElectrostaticInteraction(ElectricallyC
harged otherParticle)
```

```
{
  double force;

/* calculation of what we call force.
It is not necessary that  this is the
"formula" the calculating space uses*/
force = coupling *
this.ECharge*otherParticle.Echarge
/distance(otherParticle)/distance(othe
rParticle);

/* modification of spatial (real
world) coordinates following
interaction the right hand side is not
necessarily the way the calculating
space operates the observable change
in coordinates */
this.ECoord = this.ECoord + force /
this.EMass;
otherParticle.ECoord =
otherParticle.ECoord-
force/otherParticle.EMass;

return force;
  }

public double
distance(ElectricallyCharged
otherParticle)
  {
    double dist;

/* proposed formula for the
calculation of the spatial coordinates
*/
dist = Math.abs(this.ECoord -
otherParticle.ECoord);

return dist;
  }

/* method 1 ()
.
.
.
method n () */
}
```

```
public class TwoCharged {

private static double result;

/* this function returns the numerical
value of the force between two charged
particles and it checks the way the
spatial coordinate changes following
interaction */
public static void main (String[]
args)
{
/* instantiation of two charged
particles in the calculating space */
ElectricallyCharged electron = new
ElectricallyCharged (-1, 200, 1);
ElectricallyCharged positron = new
ElectricallyCharged (1, 400, 1);

/* numerical value of force between
the two electrically charged particles
*/
result =
electron.ElectrostaticInteraction(posi
tron);

System.out.println(result);

/* check how the spatial (real
world)position of the charges has
modified following interaction */
System.out.println(electron.ECoord);
System.out.println(positron.ECoord);
electron.ElectrostaticInteraction(posi
tron);
System.out.println(electron.ECoord);
System.out.println(positron.ECoord);
}

}
```

Let us start from the observed physical phenomena of electrostatic interaction, i.e. attraction or repulsion. The subjects of the interactions will be particles, so we need to identify an interface named particle, made of abstract methods saying how a generic particle functions, having mass, electric charger, color charge and so on. Next, we need to consider that they are charged particles and identify another class that implements only those methods of interface Particle needed for electrostatic interaction, class **ElectricallyCharged**. If in the physical word we observe one electron and one proton this is due to the fact that two **ElectricallyCharged** objects were instantiated in the computing space with characteristics specific to electrons and protons, namely electric charge (**ECharge**), initial spatial coordinates (**ECoord**) and mass (**EMass**). They are then left to interact through a method **ElectrostaticInteraction** which leads to changing of some of their characteristics, in this case of their spatial coordinates. Some of the variables cannot be modified by the interaction with other objects (**encapsulation**).

3. CONCLUSIONS

We presented a proposal for how the mechanisms of interaction work at the level of the computing space, and not of how the space itself may or may not be digitized. In the more general context of considering OOP mechanisms as **responsible** for the physical phenomena we observe in the real world, we discussed some simple routines. These routines have the purpose not to serve as simulation software but to provide an example of how we believe the real world is driven.

At this point, the framework we propose is not able to describe an experiment that uniquely distinguishes this framework from others.

4. FUTURE WORK

The mechanism can be further refined by using optimization methods and we propose that the concept of optimization in programming is connected to n^{th} degree approximation methods used in mathematical physics.

In the world of particle physics a very important issue is the value of the coupling constant and one of the ideas springing from our efforts in this work is that unification of all interactions is real and comes from the fact that the numerical value of the coupling constant acts likes a priority index. More clearly, it may be argued that there is just one unified interaction whose different manifestations are based on different access speeds (to information stored in different addresses of the computing space) dictated by the coupling constant.

REFERENCES

[1] Budnik, P., *What is and what will b: Integrating spirituality and science*, published online, http://www.mtnmath.com/willbe.html, 2005.

[2] http://en.wikipedia.org/wiki/Main_Page

[3] Atwood, W.B., Blankenbecler, R., Kunz, P., Burnett, T., Storr, K.M., *Monte Carlo and Detector Simulation in OOP*, SLAC-PUB-5359, 1990

[4] Zuse, K., *Calculating space*, PhD Thesis, MIT, Cambridge, 1970.

[5] Lewis, J., Loftus, W., *Java software solutions*, Addison-Wesley Publishing, 4th Edition, 2004.

[6] Schildt, H., *C: The complete reference*, Teora Publishing, 2003.

[7] Yavorski, B., Detlaf, A., *Aide-memoire de physique*, MIR Publishing, 1975.

ACKNOWLEDGEMENTS

G.M. acknowledges the financial support of the Sectorial Operational Programme for Human Resources Development 2007-2013, co-financed by the European Social Fund, under the project number POSDRU/107/1.5/S/76841 with the title "Modern Doctoral Studies: Internationalization and Interdisciplinary"

COMPORTAMENTUL DE PROGRAMARE ORIENTATA PE OBIECT (OOP) AL PARTICULELOR ELEMENTARE

Rezumat: *Scopul acestei lucrări este de a propune o nouă paradigmă care să explice mecanismele intrinseci care guvernează legile fizice. Comportamentul de Programare Orientată pe Obiect este o nouă abordare a modului în care Natura prioritizează interacțiunile dintre particulele elementare şi a modului în care informația este stocată în ansamblul spațiu-timp. Propunem această abordare în contextul mai larg al domeniului „Digital Physics" dar adăugăm ca şi element original comportamentul OOP.*

POP Dan Anastasiu, Dr. Fiz., Universitatea Tehnica din Cluj – Napoca, Facultatea de Stiinta si Ingineria Materialelor.

MOCANU Gabriela Raluca, Drd. Fiz., Universitatea Babes-Bolyai, Cluj Napoca, Facultatea de Fizica.

ARGHIR George, Prof. Dr. Ing. Fiz., Universitatea Tehnica din Cluj-Napoca, Facultatea de Stiinta si Ingineria Materialelor.