www.iop.org/journals/physed

# Playing Newtonian games with Modellus

**Vitor Duarte Teodoro**

Faculty of Sciences and Technology, New University of Lisbon, Portugal

E-mail: vdt@mail.fct.unl.pt

**Abstract**

This article is a short introduction on how to use Modellus (a computer package that is freely available on the Internet and used in the IOP *Advancing Physics* course) to build physics games using Newton's laws, expressed as differential equations. Solving systems of differential equations is beyond most secondary-school or first-year college students. However, with Modellus, the solution is simply the output of the usual physical reasoning: define the force law, compute its magnitude and components, use it to obtain the acceleration components, then the velocity components and, finally, use the velocity components to find the coordinates.

M This article features online multimedia enhancements

## Introduction

The essence of physics lies in the free creation of concepts, models and theories, like art. But, unlike art, ideas in physics must be matched with experience and have internal coherence. Newtonian theory is usually quoted as a paradigm of scientific theory and its influence has been enormous, both on theoretical aspects and also on practical ones: it is used today by astronomers, aeroplane pilots, aerospace engineers and... game developers!

Let's start the game.... Descartes proposed that points in space can be represented by coordinates. Modellus Animations are typically made on a plane using pixel coordinates, $x$ and $y$, on the Animation Window (see figure 1). The screen size in pixels varies, but a typical screen can have $1024 \times 768$ pixels and a Modellus Animation is usually smaller.

Let's think of a particle as an object with mass but no size—this is a typical physical concept, an abstraction from reality. Students learn in general science courses that the distance $d$ travelled by an object with constant acceleration is given by the function $d = \frac{1}{2}at^2$, where $a$ is the magnitude of the acceleration and $t$ is the independent variable time ($t$ must be measured from the beginning of the motion with constant acceleration and when $t = 0$ the object is at rest).

Instead of using $d$, we will use the pixel coordinate $x$ to represent the *distance travelled*. If $x = 0$ when $t = 0$, $d$ and $x$ can be made equal in magnitude. This motion can be represented as a Modellus Animation such as in figure 2.

Figure 2 shows a kinematics approach to motion, invented by Galileo (not exactly with this formalism!), before Newton. Newton's dynamics approach starts with the definition of the net force on the particle and then uses his laws to compute the acceleration (figure 3).

A model like the previous one uses a function of time. This is an **explicit model**, since all values of $x$ are completely defined once you write the equation. As a consequence, models of this type are **not suitable for creating interactive games**, where users need to change variables (e.g., parameters or current values of position and

0031-9120/04/050421+08$30.00  

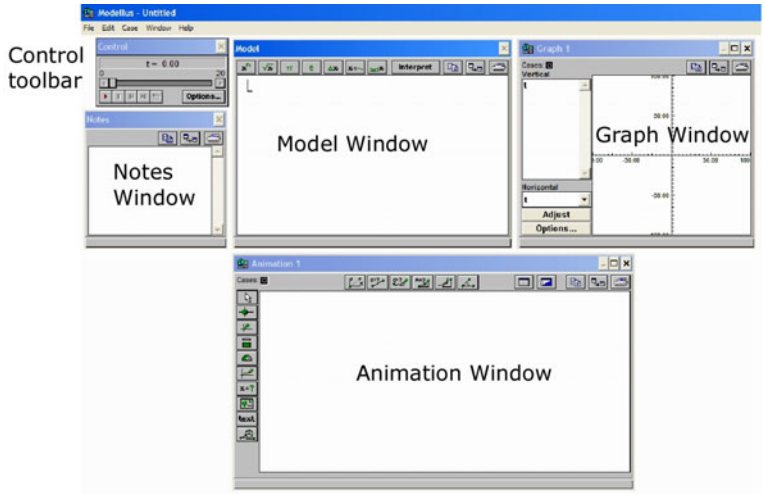


**Figure 1.** Modellus windows.

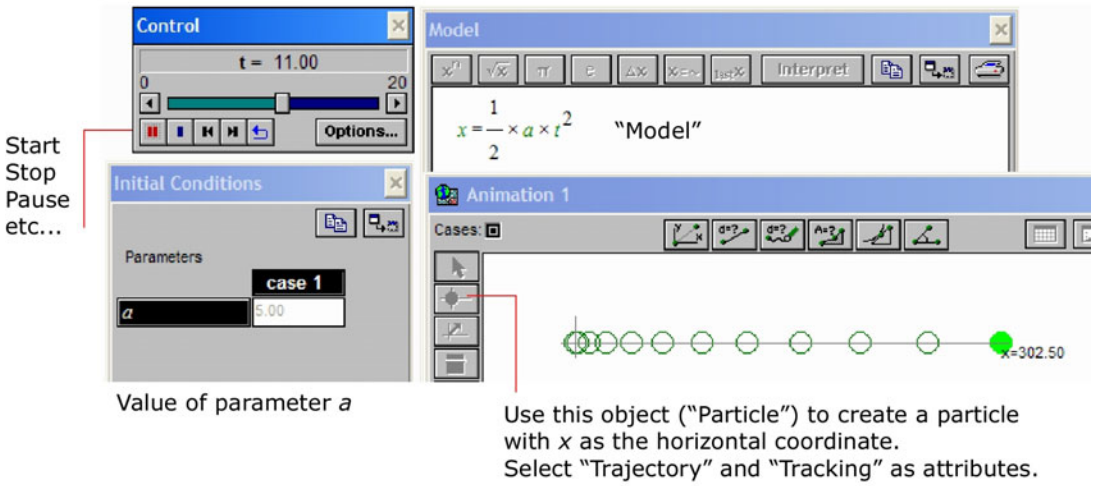


**Figure 2.** (*a*) Write the Model on the Model Window; (*b*) press the Interpret button; (*c*) give a value to the parameters; (*d*) create an object (a Particle in the current example) with the correct properties; (*e*) press the Start button.

velocity) to attain certain goals, such as controlling the motion of a object to reach a specific position.

Newton's theory states that velocity is the **instantaneous rate of change** of the position vector (a vector that has its tail at the origin of the reference frame and its head at the current position). For one-dimensional motion, this can be stated as 'the scalar component of velocity, $v_x$, is the instantaneous rate of change of $x$'. Similarly, acceleration is the instantaneous rate of change of velocity. Again, for one-dimensional motion, we can say that 'the scalar component of acceleration, $a_x$, is the instantaneous rate of change of velocity $v_x$'. A Modellus model of these statements looks like figure 4.

The model in figure 4 uses instantaneous rates of change to relate velocity and acceleration, and position and velocity. Equations like $\mathrm{d}x/\mathrm{d}t = v_x$ are called **differential equations**. They establish how one or more variables change. The origin of the term 'differential' has to do with the fact that a 'change' is a 'difference' between a 'new value' and an 'old value'.

Differential equations have 'solutions', like algebraic equations. **A solution of a differential equation is a function** (or a family of functions).

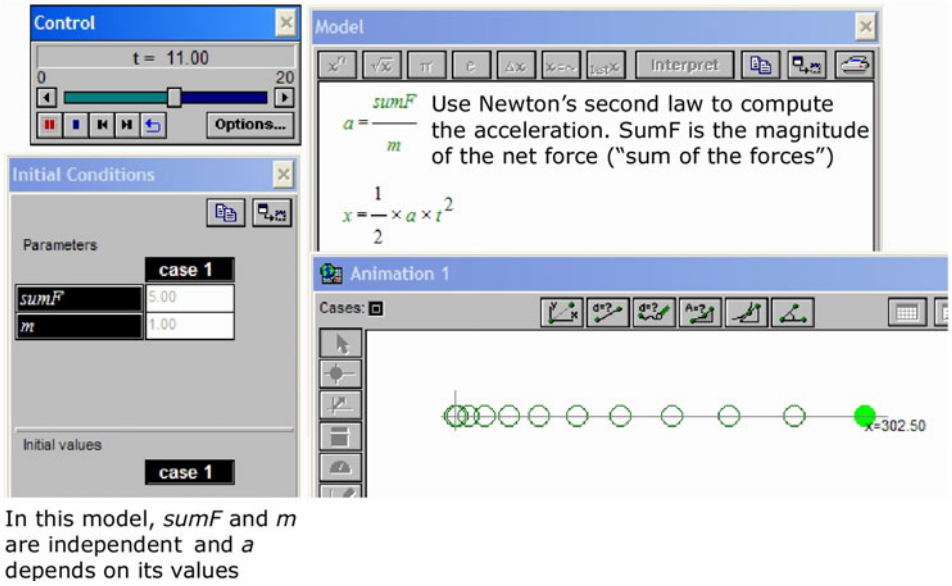


**Figure 3.** A model of motion with constant acceleration: the net force is constant. The model is made with a function of time.

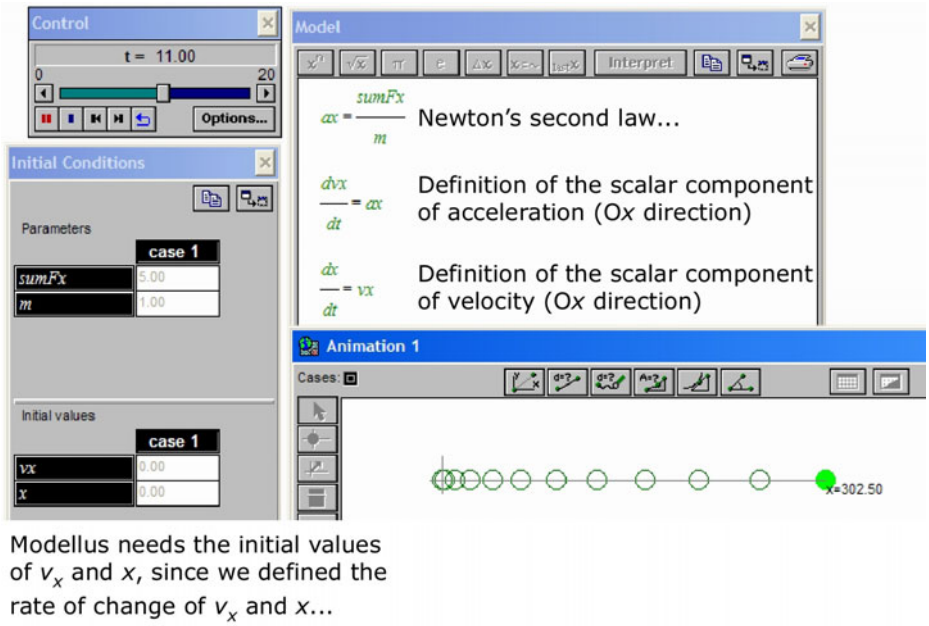


**Figure 4.** A Newtonian model of a particle with constant acceleration. The model is made using instantaneous rates of change, i.e. differential equations.

Sometimes, it is very easy to find the solution, i.e. the function that solves the differential equation. For example, if the initial value of the velocity is 10 units and the net force is zero units, the solution is $x = 10t$. Why? Simply because there is no rate of change of velocity. And, since distance changes by 10 units per unit of time, its value is 0, 10, 20, 30 etc, when time is 0, 1, 2, 3 etc, respectively (see figure 5). Finding the function that solves the differential equation is a process known as finding the **analytical or symbolic solution** of the equation.

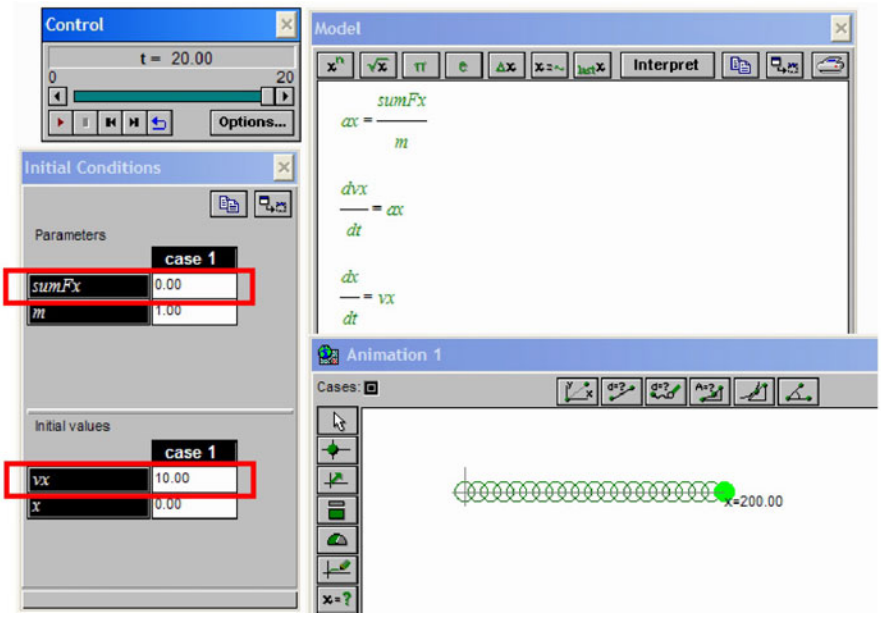


**Figure 5.** A particle moving with constant velocity (note that the net force is zero). The model was created with differential equations and it is equivalent to a model with the function $x = 10t$.

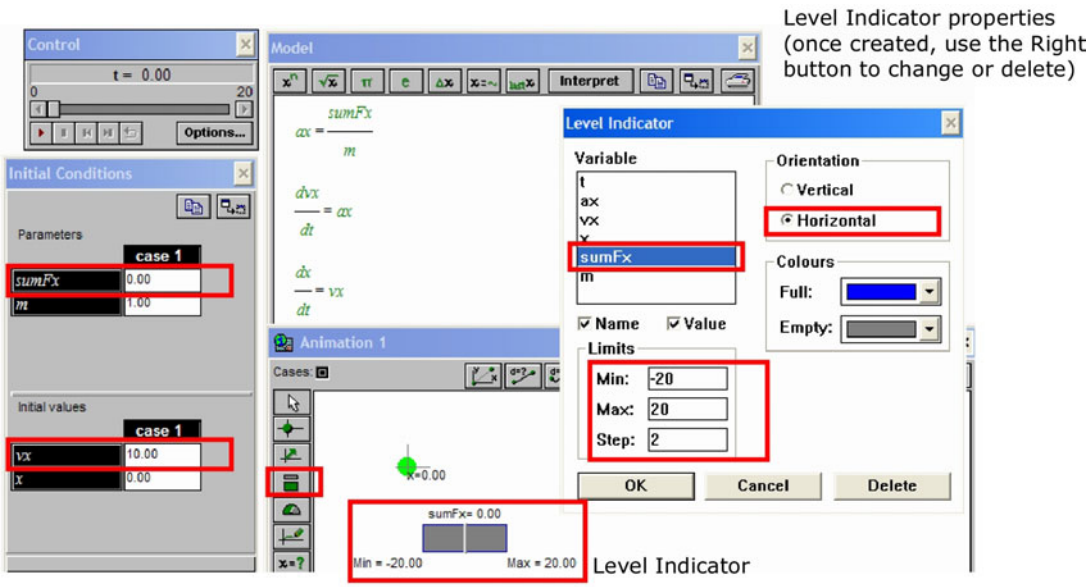


**Figure 6.** The Level Indicator was set to control the variable *sumFx*. This variable is the scalar horizontal component of the net force.

The units used are important, but in this case it is sufficient to say that they must be coherent—if distance is measured in metres (=pixels) and time in seconds, velocity is measured in metres per second and acceleration in metres per second per second.

Modellus computes solutions of differential equations using a **numerical method**. This means that, for example, in the case of the model of figure 4, it doesn't know that the solution for the function $x$ is $\frac{1}{2}a_x t^2$. But the method used by Modellus is powerful enough to give a numerical

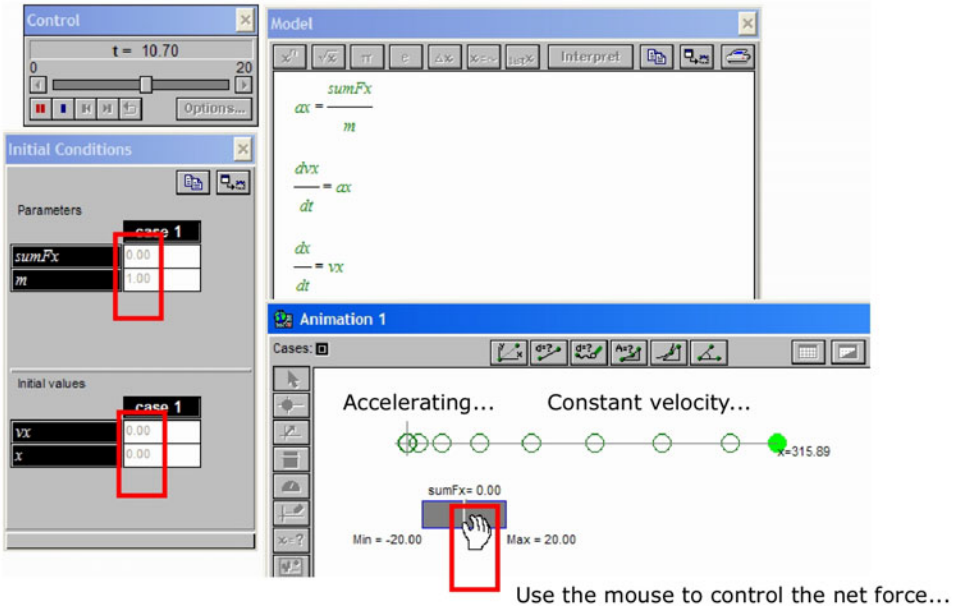


**Figure 7.** Getting the feel for the mathematical meaning of inertia.

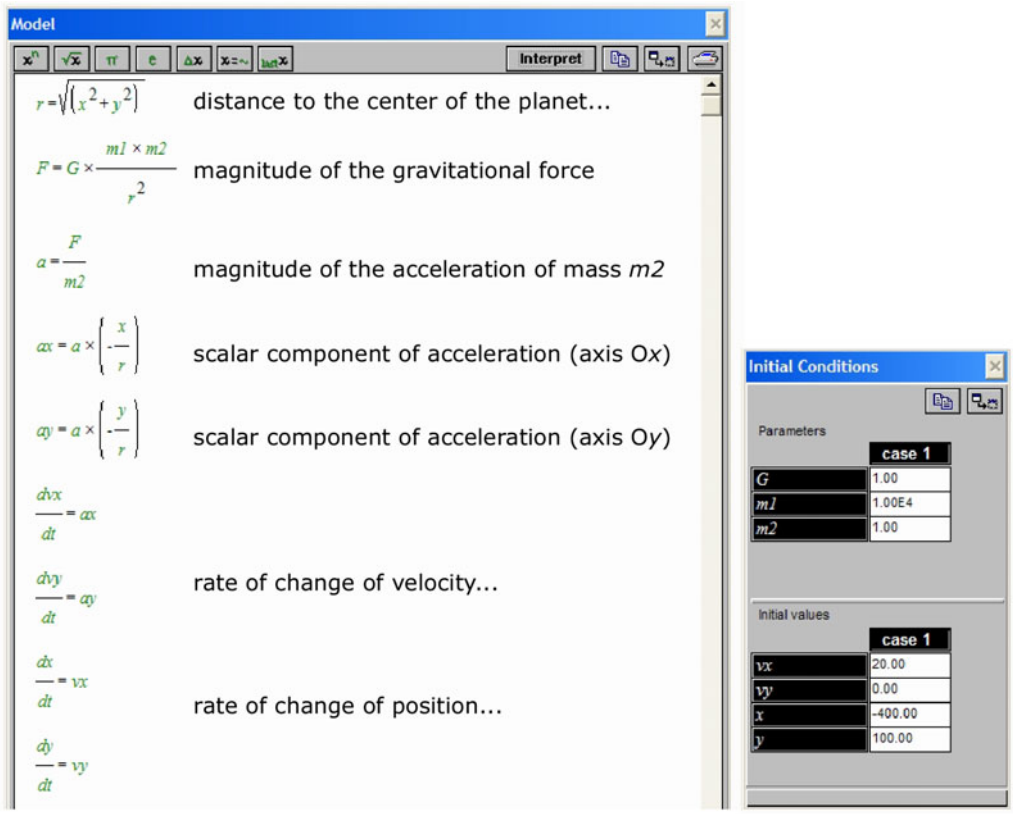


**Figure 8.** A model (and its initial conditions) to create a gravitational interaction between a fixed planet of mass $m_1$ and a moving particle of mass $m_2$.

solution that is equal or almost equal to the symbolic solution in this case and in many more cases. The basic idea in solving differential equations is that a new value of a function (let's say $x$) is computed using an algorithm of the type 'new value = old value + rate of change × time step'. If the time step is conveniently small and the 'rate of change' is known, this iterative computation can compute new values of the function as time goes on.

The advantages of using numerical methods to solve differential equations to create **interactive games** are evident: by controlling the rate of change, the user can control how something moves on the screen; let's see a simple example.

## Creating an interactive model that illustrates the law of inertia

Modellus allows different forms of interaction when a model is running. For example, you can place a Level Indicator on the Animation Window, assign it to a parameter and, when the model is running, move a slide to change the value. The example in figure 6 has a Level Indicator to change the variable *sumFx*.

This model can illustrate how inertia is described mathematically. Assign the value 0 to all parameters except mass (which can be 1) and assign to the Level Indicator the properties highlighted in figure 6. Run the model and, after a few seconds, drag the *sumFx* Level Indicator to

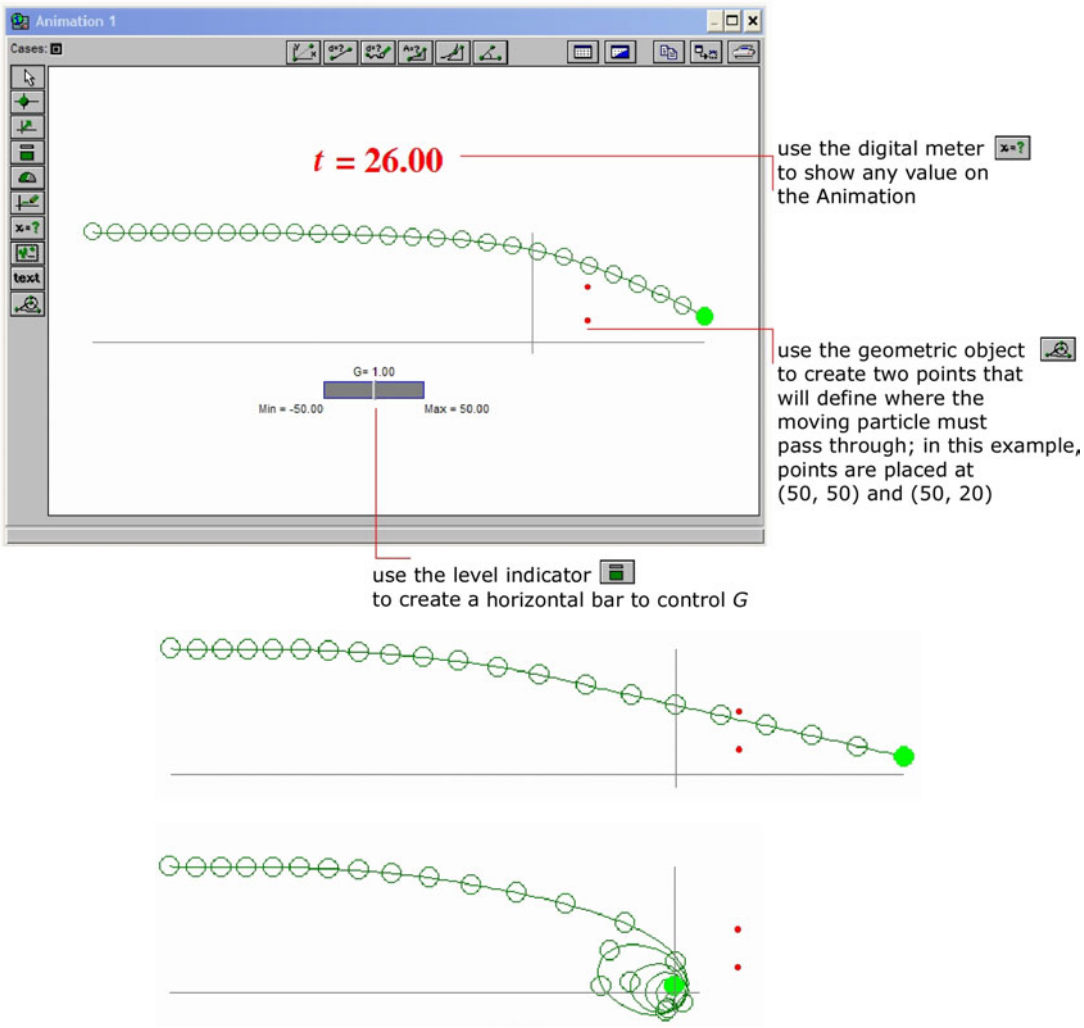


**Figure 9.** A gravitational game and two trajectories of the moving particle: the first one was a success but the second was a failure: the object was captured by the planet.

the right. The particle starts moving, accelerating. Then move the slider to 0 (net force becomes zero) and you will see that the particle continues moving, unless you change the net force again—see figure 7.

Try to explore this model with different control sequences of the net force. For example, try to stop the particle, and keep it stopped... and then accelerate the particle backwards.

## An interactive game using Newton's laws

You are now ready to construct an interactive model to play a **gravitational game in two dimensions**. Let's start by assuming a particle (a planet) with a large mass $m_1$ placed at the origin of a reference frame $xOy$. Another particle (a spaceship, a satellite, anything you want...) is travelling in the gravitational field created by $m_1$, acted upon by the gravitational force.

The model (see figure 8) starts by computing the distance $r$ between the two particles, then the magnitude of the gravitational force and the magnitude of the acceleration of particle $m_2$. This acceleration is then decomposed into the two components, $x$ and $y$, using simple trigonometric relations (cosine and sine) and taking into account that forces are attractive (the minus signs on the scalar components indicate that acceleration and force point towards the origin of the reference frame). Once we have the acceleration components, just write that the rate of change of velocity is acceleration and that the rate of change of position is velocity (in the two dimensions).

As can been seen from the Initial Conditions window (figure 8), the gravitational constant $G$ has an arbitrary value of 1 (just to simplify things), the mass $m_1$ is 1000 times bigger than $m_2$ (that's why we can neglect the gravitational force on $m_1$), the initial position of mass $m_2$ is (100, −400) and it is moving in the O$x$ direction with an initial speed of 20 units.

Finally we have a simple gravitational game. Creating a target (see figure 9), the goal is to make the moving particle pass through two marks by controlling the gravitational constant $G$. But be

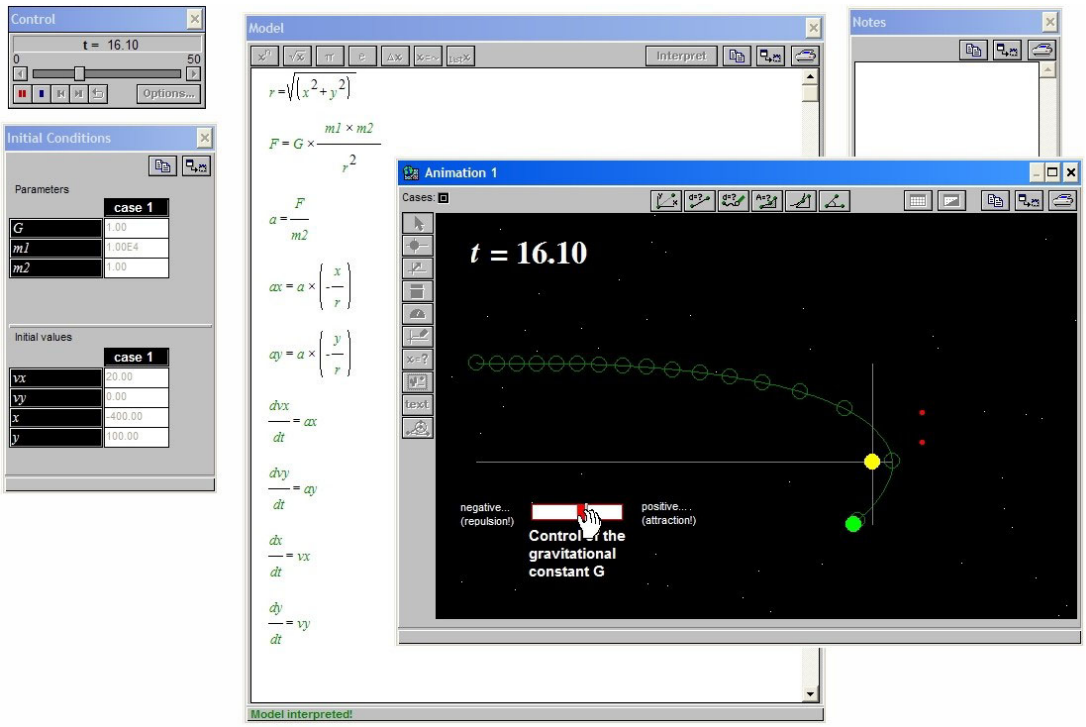


**Figure 10.** The Animation window can be changed to make the 'game' look more realistic. For example, it is very easy to put 'stars' as a background, to include text and colour etc. The user can also choose any image ('gif' or 'bmp' filetype) as the background, as well as a moving object (e.g. the spaceship) or a static one (the attracting body).

careful: it is more difficult than it seems.... The 'game' can be made more realistic by changing the Animation window (figure 10).

## A final word

Physics is a process of creating and testing models about the world. Physical quantities like force, energy, time etc are abstract mathematical concepts that physicists and engineers use to describe and manipulate the world. Playing with models like those above can help students appreciate the way in which models are built, tested and used, particularly those models that use mathematics as the language of Nature.

Modellus is available as a free download from phoenix.sce.fct.unl.pt/modellus/. The Modellus files for the models shown in figures 2–10 are available from the electronic version of this journal (www.iop.org/journals/physed).

*Received 15 July 2004*
*PII: S0031-9120(04)83355-0*
*doi:10.1088/0031-9120/39/5/005*

**Vitor Duarte Teodoro** is a professor at the New University of Lisbon. He has wide experience of teaching physics in schools and at university. He and two of his students designed Modellus in the 1990s, inspired on work done in the UK (mainly by Jon Ogborn) and in the US (by Judah Schwartz and others).