

## QUANTUM DUCK HUNT—A COMPUTER GAME

By Tarun Biswas

**H**ISTORICALLY, USEFUL ADULT SURVIVAL SKILLS HAVE OFTEN BEEN TAUGHT TO CHILDREN THROUGH GAMES. IN MOST SOCIETIES THROUGH THE AGES, WARFARE AND HUNTING GAMES HAVE BEEN ENCOURAGED FOR BOYS, AND CHILDCARE

games have been encouraged for girls. Today, you might argue the gender bias of such games and maybe even their usefulness, but you can't ignore the power of games as educational tools.

Present-day adult survival skills have become significantly more varied and complex. Even the age-old skills of warfare, hunting, and childcare have become more complex, but we still lay the seeds of learning in young minds through games. This is because experience gathered at a young age in the relaxed and enjoyable setting of a game becomes part of one's common sense or intuition.

Physics education starts at a very early age when children learn to play with toys such as balls, building blocks, spinning tops, and so forth. However, such games seem to build good intuition for classical physics alone. Quantum physics games are hard to find,<sup>1</sup> and the reason, of course, is the microscopic scale of most direct quantum phenomena. This is where the recent trend of computer games replacing "real" games might actually be useful! On a computer, simulating a massive-scale magnification is not a problem; therefore, I present a classic game with a quantum twist—Quantum Duck Hunt.<sup>2</sup>

Using computers for education (in particular, physics education) has been researched extensively and continues to be so<sup>3,4</sup>—people have written different kinds of computer software for this purpose.<sup>5</sup> It has also been suggested that such software be somewhat game-oriented,<sup>6–8</sup> so the suggestion here is to use software programs that really are games.

### The game

Quantum Duck Hunt opens with a picture of the sky. When ready, the player selects "start" from the Start menu, and, with an initial quack, a duck appears from either the left or the right side of the screen from a random height with a random speed and direction (see Figure 1a). The duck's flight is appropriately animated. The mouse pointer becomes a cross-wire, and the player proceeds to shoot the duck with it. Gunshot sound is included to provide some realism, but hitting the duck does not always kill it. I discuss this quantum effect in the next section. For now, we see that, if the duck is killed, it makes an appropriate noise and falls to the ground. If the duck manages to leave the screen without being killed, it survives. The next duck appears on the screen only after the previous

one has left (dead or alive).

In each game, a total of 10 ducks will appear. Each duck announces its entry with a quack and, like the first one, might appear anywhere on the left or right side of the screen moving at a random velocity (speed and direction). On the bottom left corner of the screen, next to the picture of a duck face, the current number of ducks killed is displayed. The number of shots fired is shown just above it (next to the picture of a shot). When the game is over, a score is displayed based on the number of ducks killed and the number of shots fired. The score could be negative if too many shots are fired to kill too few ducks. The Start menu includes a choice of "pause," letting the player freeze all action, take a breather, and study the game a little. The Help menu provides a description of the game and some explanations of quantum effects.

### The quantum effect

The quantum duck is modeled after a quantum free particle, and shooting a duck is like measuring the position of a quantum free particle using a probe (such as a photon). Therefore, we need to look at a free particle's position eigenfunctions. Exact position eigenfunctions are Dirac delta functions. Thus, it is appropriate to use a finite approximation of the Dirac delta function for our purposes. One such approximation is a sharply peaked Gaussian wave packet. We can compute the time development of such a wave packet with the time-dependent Schrödinger equation.<sup>9,10</sup> For

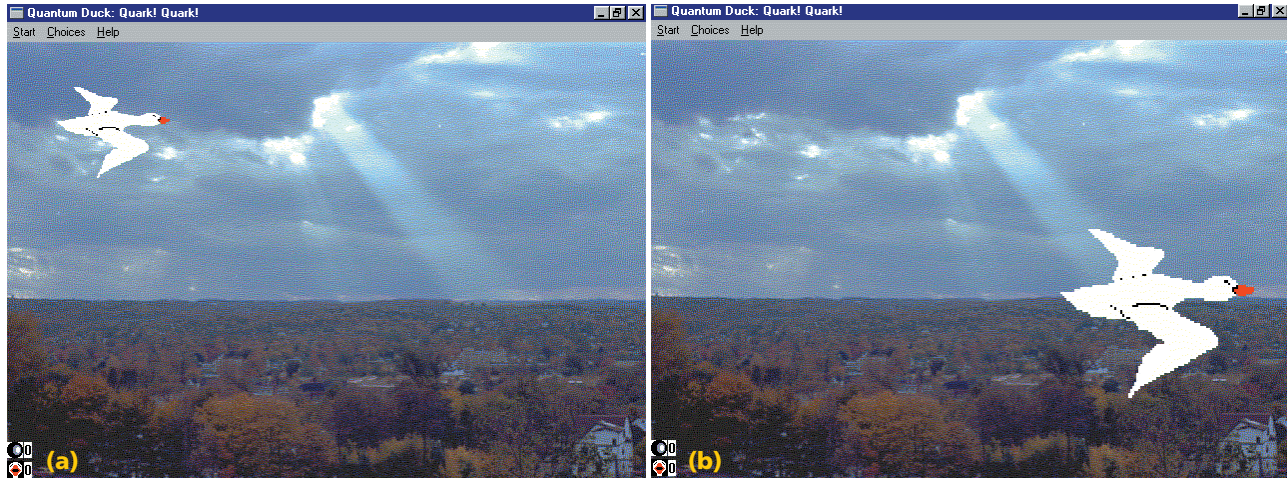


Figure 1. The duck's (a) initial appearance and (b) appearance at a later time.

a stationary particle in one space dimension, it is as follows (not normalized):

$$\psi = \left[ -\frac{t+ib}{t^2+b^2} \right]^{1/2} \exp \left[ \frac{imx^2 t}{2\hbar(t^2+b^2)} \right] \exp \left[ -\frac{mbx^2}{2\hbar(t^2+b^2)} \right] \quad (1)$$

where  $m$  is the particle's mass,  $\hbar$  is Planck's constant divided by  $2\pi$ ,  $t$  is time,  $b$  is a constant related to the wave packet's initial width, and  $x$  is the position at which the wave function is specified. The probability density computed from this wave function is

$$|\psi|^2 = (t^2 + b^2)^{-1/2} \exp \left[ -\frac{mbx^2}{\hbar(t^2 + b^2)} \right] \quad (2)$$

which shows that the probability density spreads with time (the wave packet becomes wider) while the peak value decreases. The total probability ( $\int |\psi|^2 dx$ ), as expected, does not change with time.

As the duck moves, we must consider the wave packet's peak to be moving. A first instinct might be to replace all the  $x$ 's in Equation 1 with  $(x - vt)$  ( $v$  being the peak's velocity) to achieve this motion. However, such a  $\psi$  does not satisfy the Schrödinger equation—this is because the Schrödinger equation is not Galilean invariant. After some tedious computation using the propagator,<sup>9</sup> we find the appropriate wave function for a moving particle wave packet to be

$$\psi = \left[ -\frac{t+ib}{t^2+b^2} \right]^{1/2} \exp \left[ \frac{imx^2}{2\hbar t} \right] \times \exp \left[ -\frac{imb^2(x-vt)^2}{2\hbar t(t^2+b^2)} \right] \exp \left[ -\frac{mb(x-vt)^2}{2\hbar(t^2+b^2)} \right] \quad (3)$$

where  $v$  is the velocity of the wave packet's peak. It might be a little uncomfortable to see the particle wave function not as a Galilean invariant, but the discomfort is only minor because the probability density is still Galilean invariant:

$$|\psi|^2 = (t^2 + b^2)^{-1/2} \exp \left[ -\frac{mb(x-vt)^2}{\hbar(t^2 + b^2)} \right] \quad (4)$$

It is this probability density that is the inspiration for the quantum duck. It moves at a constant velocity while simultaneously growing in size. However, the total probability over all space remains unchanged, which means that the probability density decreases near the peak.

For the duck, it is not necessary to be mathematically precise as long as the concept is illustrated. So, for simplicity and speed, we choose a rectangular probability density function for the duck rather than a Gaussian. (For that matter, even the Gaussian is an approximation. For analytical manipulations, the Gaussian approximation is more convenient, but for numerical work the rectangular one is better.) It is also chosen to be in two dimensions because it resides on the computer screen. So, the probability density of the duck (see Figure 2) is

$$|\psi|^2 = \frac{1}{\sigma^2} \text{rect}(\mathbf{r} - \mathbf{v}t, \sigma) \quad (5)$$

where  $\mathbf{r}$  is the position vector and  $\mathbf{v}$  is the velocity. Also,  $\sigma$  is the packet's width, which changes linearly with time as follows:

$$\sigma = \sigma_0 + \alpha t \quad (6)$$

where  $\sigma_0$  and  $\alpha$  are constants. The function  $\text{rect}$  is defined as

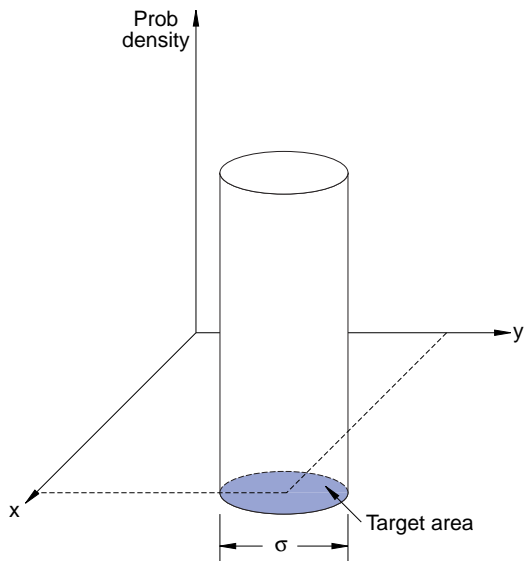


Figure 2. The two-dimensional probability distribution of the duck wave function. The x-y plane is the computer screen.

$$rect(\mathbf{r}, w) = \begin{cases} 1 & \text{if } |\mathbf{r}|^2 < w^2 / 4 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Hence, the duck wave packet shares the qualitative aspects of the particle wave packet. The center moves at a constant velocity, while the width increases and the probability density decreases. This keeps the total probability over all space fixed.

The circular region in which the probability density is nonzero is called the *target area*. The actual image of the duck is superimposed on this target area and moves with it. The image of the duck grows in size with time just like the target area (see Figure 1), but the game player does not see the actual target area and he or she must estimate its size from the visible duck’s size.

A direct hit within the target area means that there is a nonzero probability of killing the duck (as given by Equation 5) but it is not a 100%. Hitting outside the target area gives a zero probability of killing. Thus, waiting for the duck to grow larger makes it easier to hit but harder to kill, even with a perfect shot, as the probability density decreases with time. However, if the duck is hit as early as possible, it will be harder to hit the target area due to its smaller size—but once hit, the probability of a kill is high. Figure 3 shows the code fragment used to implement such quantum characteristics. (The program was written entirely in C++.)

The program calls the code in Figure 3 every time the player presses the left mouse button (to fire a shot). Pressing the left mouse button will immediately play the gunfire

sound (called `ShotGun`), increment the `shotcount` variable by one, and turn on the `Showfire` variable, which displays a flash at the bottom of the screen. Then, if the duck is alive, the program assigns variable `sigma` the current size of the target area as returned by the function `Duck->Blur()`. This size changes in every frame displayed. The coordinates of the center of the target area are returned by the function `Duck->Heart()`. Subtracting these coordinates from the coordinates of the point of mouse click gives the coordinates `x` and `y`. Hence, we find `z`, the distance of the point of mouse click from the center of the target area. If `z > sigma/2`, the shot was fired outside the target area and nothing more needs to be done by this function. Otherwise, we compute the constant probability density (`prob`) of the rectangular distribution. It is adjusted to give a value of 1 when `sigma` has its starting value (`InitBlur`). It is also defined as a probability density in two dimensions (see Equation 5). Next, a random number (`rnum`) between 1 and 100 is generated. If `rnum` turns out to be less than `100 * prob`, the duck is considered killed and appropriate code is inserted to show a falling dead duck. If `prob` is not zero, but the duck was not killed, some aspects of wave function collapse are implemented at the expert level.

### Satisfying the quantum mechanic

The quantum-mechanical critic might argue that the duck wave function cannot spread if the hunter is constantly looking at it. The act of looking is a measurement of position and thus would continually collapse the wave function. For the game to make sense to such a spoilsport, the hunter has to be a blind one. He or she knows the duck’s initial position and velocity from the sound of its initial quack as it enters the screen. From then on, the hunter tracks the duck by assuming it moves at constant velocity. The screen image of the duck is only in the hunter’s imagination.

This is not a complete resolution of the wave function collapse issue. If the duck is a macroscopic object (like most ducks are), its enormous number of degrees of freedom complicates the collapse of its wave function. The software must then disentangle each degree of freedom that is involved in the observation of the whole duck’s position and track the result of the corresponding eigenstate’s collapse. This would be too complex a task to be implemented in a practical computer game, even if we could resolve the mathematical as well as philosophical details of such a process.

Hence, I adopted a more simple-minded interpretation for the purpose of this game. The duck is somewhat like a structureless single particle that collapses to a position eigen-

```

void
QDuckWindow::EvLButtonDown(uint, Tpoint& point)
{
TRect rect;
float prob=0, sigma, x, y, z;
int rnum, bl;
if(Duck      && timeron)
    {
        sndPlaySound((LPCSTR)ShotGun, SND_ASYNC|SND_MEMORY);
        shotcount++;
        ShowFire = true;
        if (duckalive)
            {
                bl = (Duck->Blur()-InitBlur)/2;
                sigma = Duck->Blur();
                x = point.x - (Duck->Heart()).x;
                y = point.y - (Duck->Heart()).y;
                z = sqrt(x*x +y*y);
                if (z<sigma/2) {prob = InitBlur/sigma; prob = prob*prob;}
                if(prob)
                    {
                        rnum = MyRandom(100);
                        if (rnum<prob*100)
                            {
                                // Duck is dead. Insert code showing dead duck falling.
                            }
                        else if(Level == 2) // if in expert level.
                            {
                                // Code implementing expert level features of collapse of wave
                                // function etc.
                            }
                    }
                }
            // Repaint window to update changes.
        }
    }
}

```

Figure 3. C++ code fragment of Quantum Duck Hunt.

state as soon as its position is measured (see next section). This is like having a Planck's constant of the order of 1. However, unlike a structureless particle, we let the duck look and quack like a duck.

### The levels of difficulty and wave function collapse

The game can be played at three different levels—beginner, intermediate, and expert. At the beginner level, the starting target area is fairly large. Each duck's speed is randomly picked within a certain range, but the maximum possible speed is kept fairly low. At the intermediate level, the starting target area is the same as for the beginner level, but

the maximum possible value of the randomly picked speed is twice that of the beginner level. At the expert level, the starting target area has half the diameter as that of the beginner level. The maximum possible value of the randomly picked speed is twice that of the beginner level.

Also at the expert level, I have added an additional quantum-mechanical feature. If the player hits the duck within the target area but does not kill it, we expect it to be at least frightened. So it lets out a high-pitched squeal and makes a random change in velocity (speed and direction). The squeal lets even the blind duck hunter make a position measurement. Hence, the duck wave function collapses to its initial

## WINNERS OF THE *CiSE* FIRST ANNUAL EDUCATIONAL SOFTWARE CONTEST

By *Denis Donnelly*

*Computing in Science & Engineering* has just concluded its First Annual Educational Software Contest, modeled on past *Computers in Physics* software contests. *CiSE* proudly presents the first-place award to Eugene Butikov of the University of St. Petersburg, Russia, for his program on tides, and the second-place award to Wolfgang Christian and James Nolan of Davidson College for their applet modeling the resonance and interference effects of waves passing through various layers. Both of this year's contest winners have been *CiP* winners in the past, and both continue to build up a repertoire of first-rate software.

### Ocean tides

The Ocean Tides program provides a dynamic picture of the tide-generating forces and tidal waves for a model of a planet covered by a water envelope of uniform depth. This program, like previous programs that Butikov has developed (for example, Planets and Satellites), provides an attractive, intuitive interface that lets users engage the material directly.

On opening the tutorial, the student will see two windows, one for the simulation, the other providing information about the simulation. The tutorial lists 10 items for investigation; clicking on any one of them brings up the appropriate simulation and corresponding explanatory text. The text information is qualitative but important, providing information, for example, about reference frames or the details of parameter choice.

Students should find the presentation engaging and enlightening. On examining the different choices, they will get a sense of the forces involved, as well as their components. They will observe a rotating tidal bulge, a driven oscillating standing wave, and other examples. Once they become involved, students can consult Butikov's manual "The Ocean Tides," which provides a theoretical background to the subject in general and specifically to the phenomena that were simulated.

### EM and QM Reflection

Wolfgang Christian and Jim Nolen have provided the scientific and engineering communities with another fine physlet (an applet with physics applicability). This entry deals with reflective properties. Readers unfamiliar with Christian's work should consult his Web site to obtain a more complete sense of the scope of his work.

This particular physlet lets students explore two different areas associated with the concept of reflection. One area treats electromagnetic phenomena and models the transmission of light through multilayer films. The second area is from the field of quantum mechanics and models the transmission of de Broglie waves through piecewise continuous potential barriers. In the EM mode, the layers have different indices of refraction. In the QM mode, the different layers have different potential energies.

The opening page presents a narrow window on the left as a generalized table of contents. On the right is a larger simulation window. The opening screen gives us two demonstration scripts showing waves passing through various layers. Preprogrammed choices include interference, coating, mirror, and components for the EM mode. For the QM mode, the choices include step up, step down, or barrier (in two forms).

Students should have an exciting time exploring these phenomena. One can change, for example, the energy of a free particle as it encounters an increase or decrease in potential and observe the change in the wave function. Color is used for phase information. In the first barrier example, users can change the energy of the incident wave; in the second barrier example, users can change both the height and width of the barrier. The results are striking.

Both of the software packages described here are essentially qualitative in their presentation of phenomena. However, I expect that few students would be able to guess the results of these simulations. Both packages could be neatly incorporated into the appropriate courses.

Butikov's program, The Ocean Tides, is available at [www.unitel.spb.ru/~butikov](http://www.unitel.spb.ru/~butikov), and Christian's EM and QM Reflection physlet is available at [webphysics.davidson.edu/applets/Reflection/default.html](http://webphysics.davidson.edu/applets/Reflection/default.html). *CiSE* also describes these programs at [computer.org/cise/contest.htm](http://computer.org/cise/contest.htm). For further information, contact the authors at [butikov@spb.runnet.ru](mailto:butikov@spb.runnet.ru) and [wochristian@davidson.edu](mailto:wochristian@davidson.edu).

size. This makes the game significantly more challenging.

### Some technical details

It is important to give this kind of game the look and feel of a real computer game. However, the computer hardware requirements should be kept low to allow greater accessibility. With this in mind, I gave the game the following features.

The background picture is a 256-color 640 × 480 bitmap, and the duck image is a 4-color sprite. To animate the flying duck's wing movements, I used three images with different wing positions and a separate image for the dead duck dropping to the ground. The program changes the duck's size by simply expanding the same images before they are drawn each time on the screen.

The sound effects used are the starting duck's quack, the dying duck's groan, the squeal of the duck that is hit but not killed, and gunshot noise.

Feeling "comfortable" with quantum phenomena is not easy. Repeated exposure to such phenomena might be the only way to bring about

some degree of comfort. Such exposure is expected to be more effective at a younger age. Computer games like the one discussed here could be one way of exposing children to quantum ideas—this game is absorbing enough and children do want to play it. Whether it helps them grasp quantum mechanics more naturally at a later age is yet to be seen. ☞

### References

1. T. Biswas, "Group Theory (Symmetries) in a Computer Game," *Computers in Physics*, Vol. 12, No. 5, Sept./Oct. 1998, pp. 488–492.
2. Quantum Duck Hunt, Windows 3.1 and Windows 95/98; contact biswast@newpaltz.edu.
3. D.M. Cook, "Computers in the Lawrence Physics Curriculum," *Computers in Physics*, Vol. 11, No. 3, May/June 1997, pp. 240–245.
4. R. Ehrlich, M. Dworzecka, and W.M. MacDonald, "Computers in Physics Education," *Computers in Physics*, Vol. 6, No. 1, Jan./Feb. 1992, pp. 90–96.
5. "10th Anniversary CIP Educational Software Directory," *Computers in Physics*, Vol. 11, No. 1, Jan./Feb. 1997, pp. 49–72.
6. T. Biswas, "Nonlinear Effects in a Stretched String—A Numerical Computation," *Computers in Physics*, Vol. 8, No. 4, July/Aug. 1994, pp. 446–450.
7. H.G. Weller, "Assessing the Impact of Computer Based Learning in Science," *J. Research on Computing in Education*, Vol. 28, No. 4, Apr. 1996, pp. 461–484.
8. B. White, "Thinker Tools," *Cognition and Instruction*, Vol. 10, No. 1, Jan. 1993, pp. 1–100.
9. R. Shankar, *Principles of Quantum Mechanics*, Plenum Press, New York, 1980, pp. 159–163.
10. T. Biswas, *Quantum Mechanics—Concepts and Applications*, 1999, pp. 32–34; www.engr.newpaltz.edu/~biswast.

---

**Tarun Biswas** is an associate professor of physics at the State University of New York, New Paltz. He has worked in particle physics (alternative theories of strong interactions) relativity (special relativistic Newtonian gravity), optics, and the use of computers in physics education. He has a PhD in physics and an MS in computer science and is a member of the APS, AAPT, and Sigma Xi. Contact him at the Physics Dept., State Univ. of New York, New Paltz, New Paltz, NY 12561; biswast@matrix.newpaltz.edu.

**Submissions:** Send two copies, one word-processed file and one PostScript file, of articles and proposals to Francis Sullivan, Editor in Chief, *CISE*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; cise@computer.org. Submissions should not exceed 6,000 words and 10 references. All submissions are subject to editing for clarity, style, and space.

**Editorial:** Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *CISE* does not necessarily constitute endorsement by the IEEE, the AIP, or the IEEE Computer Society.

**Circulation:** *Computing in Science & Engineering* (ISSN 1521-9615) is published bimonthly by the AIP and the IEEE Computer Society. IEEE Headquarters, Three Park Ave., 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314, phone, +1 714 821 8380; IEEE Computer Society Headquarters, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903; AIP Circulation and Fulfillment Department, 1NO1, 2 Huntington Quadrangle, Melville, NY 11747-4502. Annual subscription rates for 2001: \$39 for Computer Society members, \$51 (print plus online) for AIP member society members, \$64 for members of other IEEE societies, and \$107 for members of professional organizations other than IEEE and AIP. Back issues cost \$10 for members, \$20 for nonmembers. This magazine is available on microfiche.

**Postmaster:** Send undelivered copies and address changes to Circulation Dept., *Computing in Science & Engineering*, PO Box 3014, Los Alamitos, CA 90720-1314. Periodicals postage paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 0605298. Printed in the USA.

**Copyright & reprint permission:** Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US copyright law for private use of patrons those articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Dr., Danvers, MA 01923. For other copying, reprint, or republication permission, write to Copyright and Permissions Dept., IEEE Publications Administration, 445 Hoes Ln., PO Box 1331, Piscataway, NJ 08855-1331. Copyright © 2001 by the Institute of Electrical and Electronics Engineers Inc. All rights reserved.