LEARNING BY VISUALIZING

Nadia Benakli
NBenakli@citytech.cuny.edu

Ashwin Satyanarayana
ASatyanarayana@citytech.cuny.edu

Satyanand Singh
SSingh@citytech.cuny.edu

Arnavaz Taraporevala
ATaraporevala@citytech.cuny.edu

NYC College of Technology
300 Jay Street Brooklyn, NY 11201

Abstract: Technology is universal and omnipresent. It has reshaped the way we function. One of the best ways to incorporate technology in the classroom is as a tool to facilitate learning by using it as a visual aid. This is a type of teaching and learning style in which ideas, concepts, data and other information are associated with images and techniques. In this paper we will discuss some of the tools and present and analyze a number of examples to demonstrate how the use of technology in the classroom and visualization can improve Math and Computer Science learning.

Keywords: Visualization, Riemann Sums, CAS, Programming, Database, Networking

Introduction

It is without doubt that teaching is undergoing a metamorphosis as technology is ubiquitous in every facet of instruction. Technology has led to the emergence of several tools that assist students to learn and understand abstract concepts using visualization. As instructors we use technology, with an emphasis on visualization, to allow students to test conjectures and add legitimacy to their guesses. Armed with this knowledge, students are motivated to come up with theoretical proofs to support their conjectures. Students gain a better understanding of the concepts and develop self-confidence in the discipline. According to the Visual Teaching Alliance [1], "*Approximately 65% of the population are visual learners*", "*The brain processes visual information 60,000 times faster than text*", "*Visual aids in the classroom improve learning by up to 400%*".

Visualization software emerged in the late 1980's for the purpose of creating and interactively exploring graphical representations of concepts. Many experimental studies designed to substantiate the educational effectiveness of such visualization technology [2]. In this paper we discuss how Visualization can help students in Math and Computer Science by giving 3 examples from each department.

A. Computer Science

Motivation

Korhonen and Malmi [3] describe a visualization system that presents novice programmers with graphical representations of algorithms which requires them to manipulate these representations

in order to simulate the algorithm. Teaching fundamental concepts in a first programming course using graphics and visualization in an OO environment, will provide sufficient programming experience, with stronger meaningful context for helping students to visualized object oriented concepts.

Over the last two decades, researchers have developed and used several visualization tools and technologies, such as: LOGO, GROK, KAREL, XTANGO and BALSA, Turing's World and JFLAP, TOONTALK, and ALICE.

In this paper we will look at using Visualization in 3 pillars of computer science (which are 3 primary modules for undergraduates taking courses in Computer Science at CityTech) which are as follows:

1. Programming
2. Databases
3. Networking

1. Programming

Programming has become the paradigm of choice for software development in industry as well as academia. Instructors of introductory programming classes are faced with the challenge of helping novice programmers learn to design, build, and debug computer programs [4]. In a large class, the challenge is often overwhelming, and teachers find themselves questioning why some novice programmers struggled in the process of learning to program. The students face problems when they try to put the pieces together, identify what constructs to use and how to coordinate those constructs, and determine what went wrong when things do not work [5].

Interviews and discussions with some of the New York City College of Technology department's faculty, who were or are currently teaching programming courses, revealed many problems facing the faculty in teaching concepts of programming. One of the major shortcomings of programming environments is the lack of visualization mechanisms [6]. Using 3D animations for program visualization offers computer science instructors an approach to introduce fundamental concepts to novice programmers [7].

In this example we use a tool called Alice, which is a 3D Interactive Graphics Programming Environment built by the Stage 3 Research Group at Carnegie Mellon University under the direction of Randy Pausch. A goal of the Alice project is to make it easy for novices to develop interesting 3D environments and to explore the new medium of interactive 3D graphics [8].

Alice provides an environment where students can create virtual worlds on their computers and populate them with some really interesting 3D objects in a creative sense, and use/modify these objects and write programs to generate animation. By writing simple scripts, students can control object appearance and behavior. In fact, it is not necessary to type lines of code at all; by using the smart editor students can drag and drop the instructions that make up their programs [9].

Alice provides several built-in action commands. In general, actions can be subdivided into two categories: those that tell an object to perform a motion and those that change the physical nature

of an object. Motion commands include moving objects within the world (e.g. **Move**), rotating them about their 3-D axes (e.g. **Turn** and **Roll**), and pointing at other objects (e.g. **PointAt**). Commands that change the physical nature of objects include object destruction (**Destroy**), dynamic object creation (e.g., **AddObject**), object resizing (**Resize**), and making objects visible/invisible (e.g. **Hide** and **Show**).

While it is beyond the scope of this paper to discuss all of Alice's action commands (the commands listed above are a subset), we discuss the **Turn** action to illustrate details. Turning is allowed in 4 directions: Forward, Back, Right, and Left. In the Turn command, it is only necessary to specify which object is to be turned, the direction it is to be turned, and how much it is to be turned. Figure 1 illustrates turning along one of the rotational axes.

Figure 1: Turn action in Alice for Helicopter example

Named Instructions:
It is possible in Alice to name a sequence of instructions. The concept of a named instruction is similar to the procedure concept in many other programming languages (or a function that just performs side effects, not returning anything). While in traditional programming languages, it may not be clear why a group of statements should be blocked into a function/procedure, in Alice it makes intuitive sense. By collecting the 10-20 Move and Turn instructions it takes to make a bunny hop, and naming this entire sequence of instructions *Hop*, it becomes clear to the student that

DoInOrder(Hop,Hop)

causes the bunny to hop twice as shown in Figure 2. Again, the animation aspect of Alice allows the student to immediately visualize the functionality of program constructs.

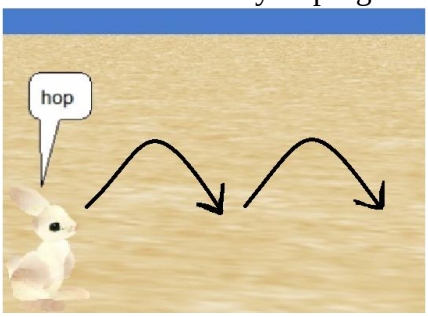


Figure 2: Bunny hop function

Functions:
Functions in Alice are supported through the underlying Python language. In Alice, functions are primarily used in the implementation of recursion/looping, and in the implementation of interactions via events. Also, they can be useful in computation. For example, the *howmany* function, illustrated below, calculates how many turns a ball will make based on its diameter and the distance it will travel.

def *howmany (dist, diam):*
return *dist / (3.14 * diam)*

Decisions:
As with functions, decisions are supported through the underlying Python language.Because of the visual feedback in Alice, students are able to see immediately the results of a decision statement. For example, in the statement:

if *cat.**DistanceTo**(Fish) < 1.0:*
DoInOrder*(*
*cat.**Move**(**Up**, 0.5),*
*cat.**Move**(**Down**, 0.5))*

if the distance between the cat and the Fish is less than one unit, the student sees the cat jump up and down as.

Figure 3: Decision making in programming using animations

While this seems simple enough, it becomes more complex in worlds containing several objects. Each object has its own orientation that may or may not be aligned with another object in the same scene. As objects move around in the world, their spatial orientation with respect to other objects may change. Alice provides good support for aligning the orientation of objects and allows an object to be positioned "AsSeenBy" another object within the world, or to be "placed" on top of another object.

We believe that Alice provides some real benefits in teaching concepts of problem solving in the particular way used in computer programming as well as in teaching the fundamental programming constructs. A major benefit is the high level of student interest and involvement. The ability to make changes in program code and, within seconds, observe the effect on their animation contributed to sustaining that interest. Students were enthusiastic, voluntarily devoting

much extra time to projects they considered fun and worthwhile. Based on student evaluations, it seems that students developed a justifiable sense of self-confidence in their programming skills.

2. Databases

A *database* is structured collection of data. Thus, card indices, printed catalogues of archaeological artifacts and telephone directories are all examples of databases. Databases *may* be stored on a computer and examined using a program. These programs are often called `databases', but more strictly are *database management systems* (DMS). Just as a card index or catalogue has to be constructed carefully in order to be useful, so must a database on a computer. Similarly, just as there are many ways that a printed catalogue can be organized, there are many ways, or models, by which a computerized database may be organized.

Almost all computer science undergraduate programs require the students to take database courses. We have noticed that students mostly face difficulties when designing databases. We have seen that this problem can be solved when the students are given the opportunity to visualize the database that they are going to design. In this paper we talk about one such Visualization tool called Tableau.

Tableau Software is a database tool that provides design of databases, visual analysis and reporting. Although there are several tools available to help users efficiently and easily create pivot tables or cross-tabulations, Tableau allows you to take it one step further: visualize the cross-tabulations in real time. With Tableau, you can drill-down database tables, visually.

Tableau's products allow students to plow through large dimensional databases quickly. As we discovered, it quickly generates graphs even against large database tables. By giving an analyst the ability to spot trends and relationships visually, information buried in the data becomes more readily available.

Tableau's products came out of R&D projects of Stanford University professor Pat Hanrahan. Hanrahan was a founding employee of Pixar and was the chief architect of RenderMan--a graphics protocol widely used in the film industry. Led by Hanrahan and Chris Stolte, the technologists at Tableau developed VizQLTM (Visual Query Language), a declarative language that allows a user to interact with a database and get graphical/visual results.

Once you connect to a data source, Tableau automatically divides the fields of your data source into dimensions or measures. In our example, we connected to a MySQL table consisting of some fake weekly book sales data, and joined to another table (using the ISBN), which contained attributes of a book: the programming language, DBMS, operating system, and where it lies in a product hierarchy (two database fields representing the two levels in a hierarchy).

In Tableau, you perform analysis by dragging fields into "shelves," which are places in the user interface where the fields being used are displayed. We found this paradigm to be very intuitive and quickly adopted to this mode of analysis. Cross-tabs and pivot tables are easy to generate in Tableau. In our example involving the join between two tables, we found that Tableau generated such tables quickly (see Figure 3).

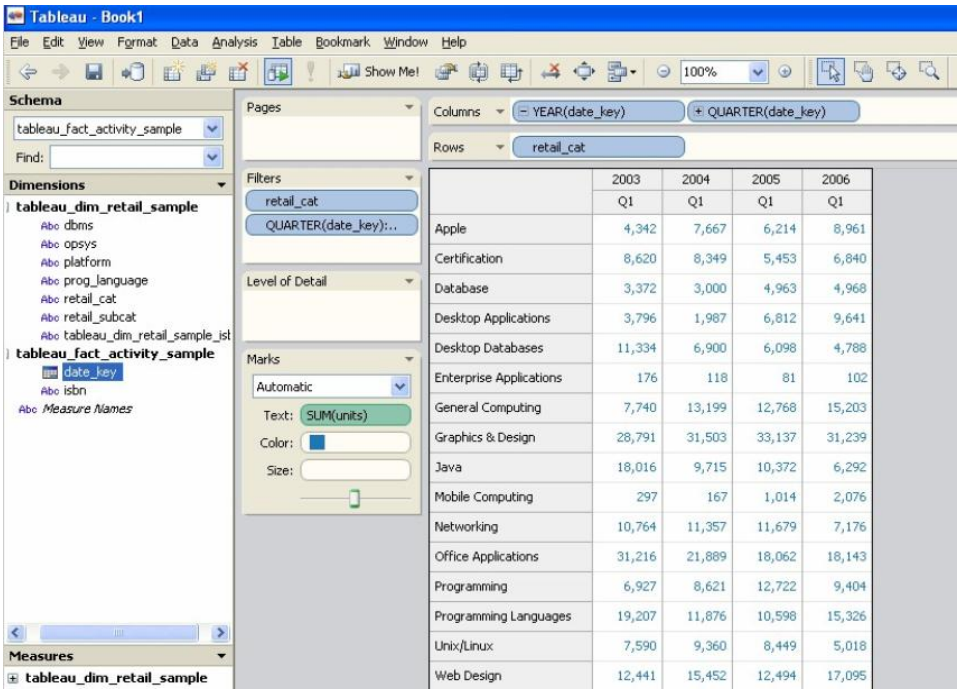


Figure 3: Tableau showing a pivot table

Cross-tabs and pivot tables are interesting, but it is frequently difficult to discern patterns when staring at numerical information. Processing large quantities of information in graphical form is easier. Tableau allows you to start analyzing the same information visually, and it generates graphs as quickly as it generates tables (see Figure 4).

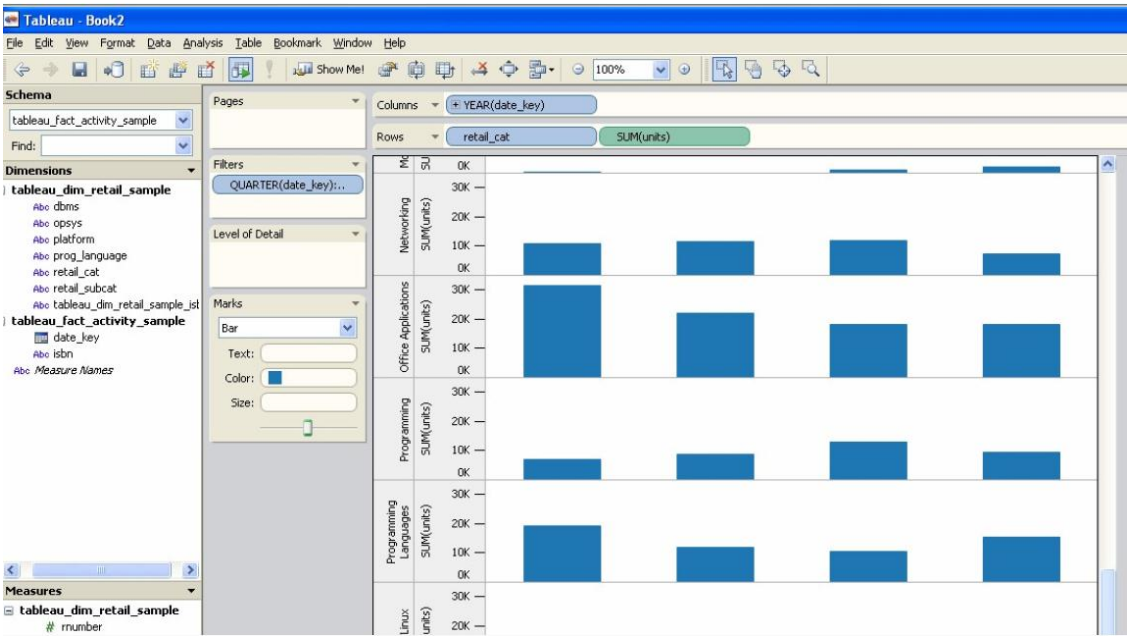


Figure 4. Tableau showing graphs for comparison

Animation also allows you to process large dimensions in an efficient manner. Setting up an animation takes only two mouse-clicks. You can also take advantage of a new feature called dynamic calculations. In our example, we quickly plotted the year-over-year percentage growth by category, thus allowing us to compare categories using the same scale. Note that Tableau allows you to do these dynamic calculations in a few steps; you don't have to perform these complex calculations in advance in your database (see Figure 5).

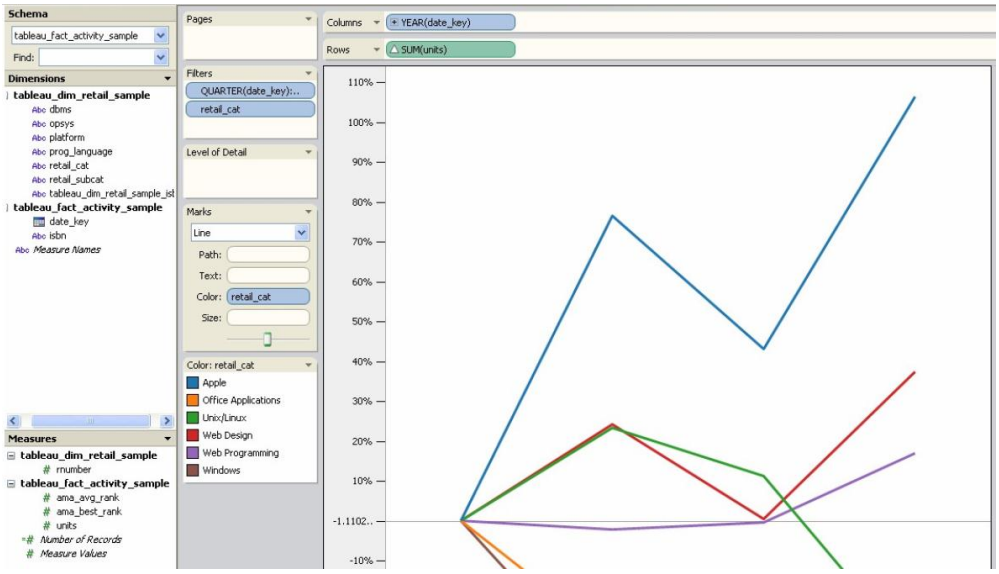


Figure 5. Setting up an animation in Tableau

We have seen that using Tableau at school helps students visualize and understand database systems better than traditional approaches. It takes minutes to install, is easy to learn, and definitely fills a need. The animation feature is well suited for viewing large categories and time-series data.

3. Networking

A network is any collection of independent computers that communicate with one another over a shared medium. A computer network is a collection of two or more connected computers. When these computers are joined in a network, people can share files and peripherals such as modems, printers, tape backup drives, or CD-ROM drives.

As Networking is closely related to Graphs, it is best for students to understand concepts in networking by visualizing the concepts. Students struggle to understand the concept of networking just from text. A graph representing the information about the relations among nodes can be a very efficient way of describing a social structure. A good drawing of a graph can immediately suggest some of the most important features of overall network structure. Are all the nodes connected? Are there many or few ties among the actors? Are there sub-groups or local "clusters" of actors that are tied to one another, but not to other groups? Are there some actors with many ties, and some with few?

In this paper, we will look at some commonly used techniques for visualizing networks using **NetDraw** (version 4.14, which is distributed along with UCINET). Using colors and shapes are

useful ways of conveying information about what "type" of actor each node is. Figures 6 and 7 provide examples. The data here describe the exchange of information among ten organizations that were involved in the local political economy of social welfare services in a Midwestern city (from a study by David Knoke; the data are one of the data sets distributed with UCINET). In Figure 6, NetDraw has been used to render a directed graph of the data. This is done by opening ***Netdraw>File>Open>UCInet dataset>Network***, and locating the data file. NetDraw produces a basic graph that you can then edit.

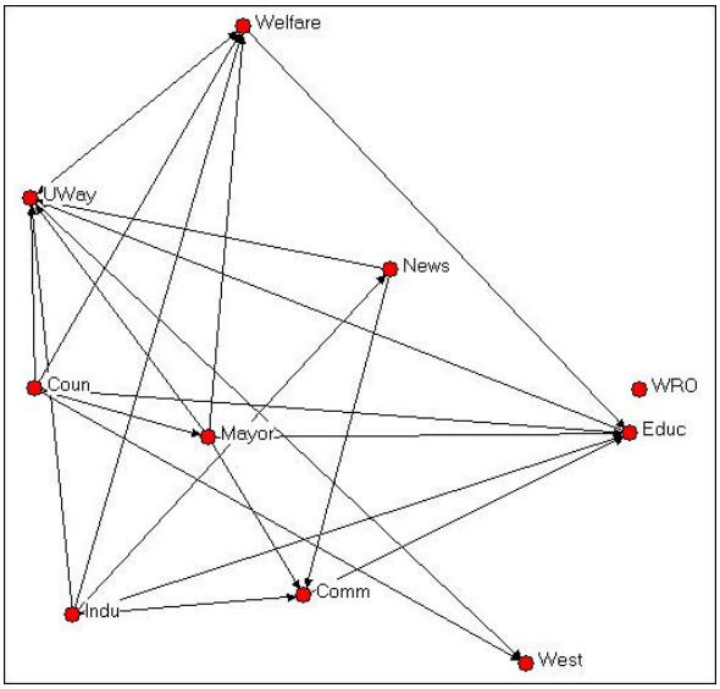


Figure 6: Knoke Information exchange network

Institutional theory might suggest that information exchange among organizations of the same "type" would be more common than information exchange between organizations of different types. Some of the organizations here are governmental (Welfare, Coun, Educ, Mayor, Indu), some are non-governmental (UWay, News, WRO, Comm, West).

In Netdraw, we used the ***Transform>mode attribute*** editor to assign a score of "1" to each node if it was governmental, and "0" if it was not. We then used ***Properties>nodes>color>attribute-based*** to select the government attribute, and assign the color red to government organizations, and blue to non-government organizations. You could also create an attribute data file in UCINET using the same nodes as the network data file, and creating one or more columns of attributes. ***NetDraw>File>Open>UCInet dataset>Attribute data*** can then be used to open the attributes, along with the network, in NetDraw.

In Netdraw, we used the ***Transform>mode attribute*** editor to create a new column to hold information about whether each organization was a "generalist" or a "specialist." We assigned the score of "1" to "generalists" (e.g. the Newspaper, Mayor) and a score of "0" to "specialists (e.g. the Welfare Rights Organization). We then used ***Properties>nodes>shape>attribute-based***

to assign the shape "square" to generalists and "circle" to specialists. The result of these operations is shown in figure 7.

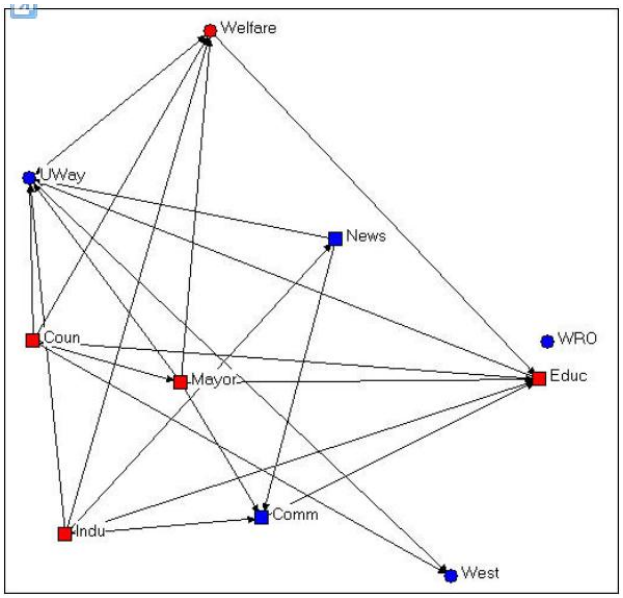


Figure 7: Knoke information exchange with government/non-government and specialist/generalist codes

A visual inspection of the diagram with the two attributes highlighted by node color and shape is much more informative about the hypotheses of differential rates of connection among red/blue and among circle/square. It doesn't look like this diagram is very supportive of either of our hypotheses.

Identifying types of nodes according to their attributes can be useful to point out characteristics of actors that are based on their position in the graph. Consider the example in the next figure 8.

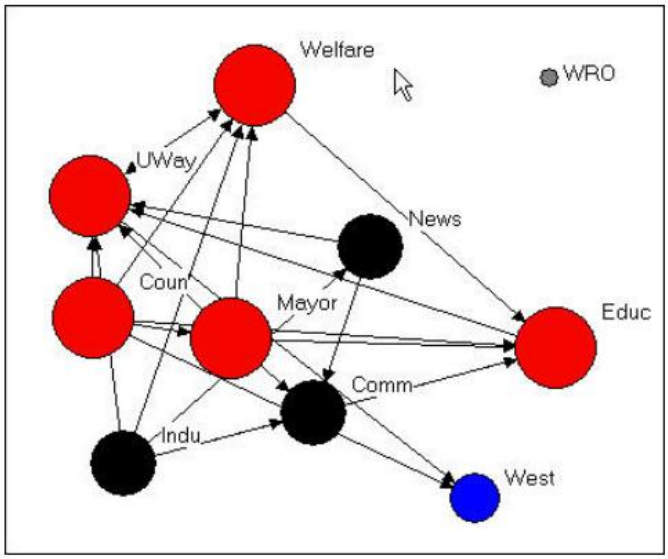


Figure 8: Knoke information exchange with k-cores

This figure was created by using the ***Analysis>K-core*** tool that is built into NetDraw. A K-core is a set of nodes that are more closely connected to one another than they are to nodes in other K-cores. That is, a K-core is one definition of a "group" or "sub-structure" in a graph.

Figure 8 shows four sub-groups, which are colored to identify which nodes are members of which group (the "West" group and the "WRO" group each contain only a single node). In addition, the size of the nodes in each K-core are proportional to the size of the K-core. The largest group contains government members (Mayor, County Government, Board of Education), as well as the main public (Welfare) and private (United Way) welfare agencies. A second group, colored in black, groups together the newspaper, chamber of commerce, and industrial development agency.

NetDraw is also very helpful for dealing with the complexity of social network data, which may involve many actors, many ties, and several types of ties. Hiding, highlighting, and locating parts of the data can be a big help in making sense of the data Finally, working with drawings can be a lot of fun, and a bit of an outlet for the creative side. A really good graphic can also be far more effective in sharing your insights than any number of words.

B. Mathematics

Motivation

Computer Algebra Systems (CAS) are powerful tools that facilitate instruction in the classroom. In this paper, we are using *Maple* software. The examples presented here can easily be adapted with *Matlab*, or *Mathematica* for example. Graphing and animating two and three dimensional figures bring to life concepts such as continuity, areas under curves, volumes and surface areas of various objects, and direction fields [10]. Graphs, especially in three dimensions, are not very easy to draw by hand. Students can graph such functions using a CAS and rotate them to get different facets of these objects.

1. Definite Integrals for functions of one variable

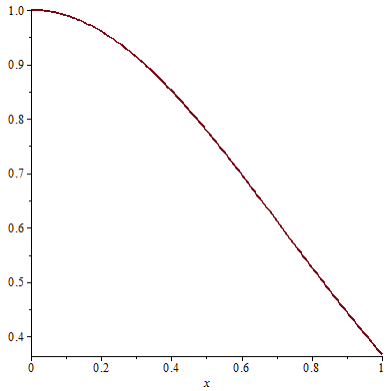
The definite integral of a nonnegative function (of one variable) over a closed interval is defined as the area bounded by the curve, the *x*-axis and the end points. The area can be obtained by constructing rectangles and finding the area of these rectangles (Riemann Sums) [11]. Students are able to see what happens as the number of rectangles increases. This concept can be extended for functions that are not necessarily nonnegative. In practice, students will use the Fundamental Theorem of Calculus and basic rules of integration to find the values of definite integrals when it is possible. We will illustrate using the midpoint Riemann sum how to calculate definite integrals [11].

Consider the function $f(x) = e^{-x^2}$. We will first define and sketch the graph of this function using *Maple* commands.

$f := x \to \mathrm{e}^{-x^2}$

$$x \to \mathrm{e}^{-x^2}$$

$plot(f(x), x = 0..1)$



We will use *Maple* to evaluate the integral.

$\int_0^1 f(x)dx$

$$\frac{1}{2}\,\mathrm{erf}(1)\sqrt{\pi}$$

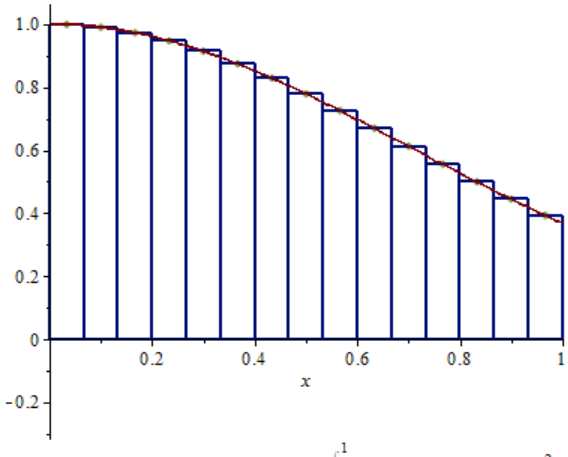
The *evalf* command is now used to obtain a decimal approximation of the integral above. The % symbol refers to the previous output

$evalf(\%)$

$$0.7468241330$$

So an approximate value of the integral is 0.7468241330. We will now show how to use *Maple* to generate a visual image of a midpoint Riemann sum approximation of the definite integral $\int_0^1 e^{-x^2}dx$. Next we will generate visual representations of the midpoint Riemann sum approximations using 15 equally spaced rectangles. Note that the height of the rectangle is the value of the function at the midpoint of the rectangle.

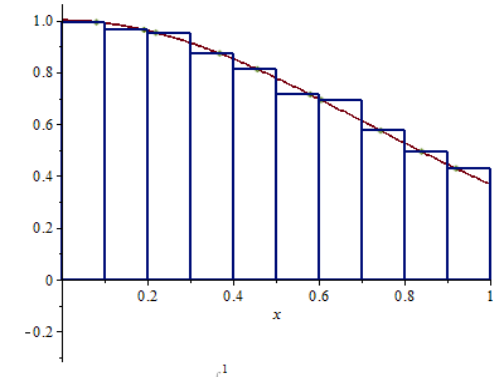
$ApproximateInt(f(x), x = 0..1, method = midpoint, output = plot, partition = 15)$



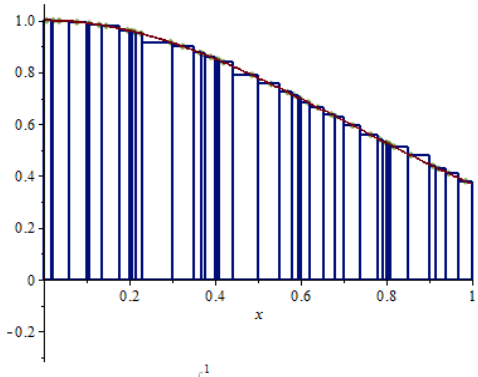
A midpoint Riemann sum approximation of $\int_0^1 f(x)\,dx$, where $f(x) = e^{-x^2}$ and the partition is uniform. The approximate value of the integral is 0.7469604197. Number of subintervals used: 15.

We now use the animation command to show what happens when the number of rectangles is increased. The animation can easily be implemented in any CAS. The graphs below show three screen shots of the output when the animation option is used. We are using randomly spaced rectangles here.

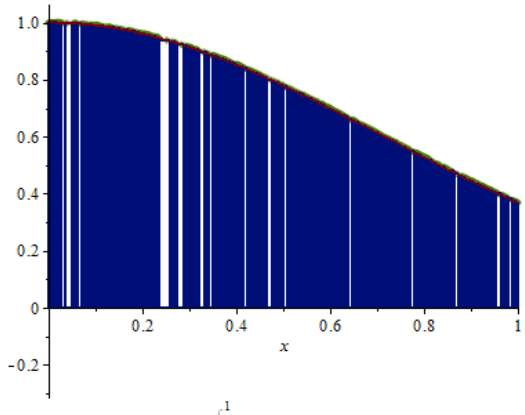
$$ApproximateInt\left(f\left(x\right), x=0..1, method=midpoint, output=animation, iterations=7, refinement=random\right)$$



An animated approximation of $\int_0^1 f(x)\,dx$ with randomly selected points, where $f(x) = e^{-x^2}$ and the partition is uniform. The approximate value of the integral is 0.7501355879. Number of subintervals used: 10.



An animated approximation of $\int_0^1 f(x)\,dx$ with randomly selected points, where $f(x) = e^{-x^2}$ and the partition is uniform. The approximate value of the integral is 0.7460347480. Number of subintervals used: 40.



An animated approximation of $\int_0^1 f(x)\,dx$ with randomly selected points, where $f(x) = e^{-x^2}$ and the partition is uniform. The approximate value of the integral is 0.7467620452. Number of subintervals used: 640.

Next, we will estimate this integral by using the limit of midpoint Riemann sums.

$$M := n \to ApproximateInt\left(f\left(x\right), x=0..1, method=midpoint, partition=n\right)$$

$$n \to Student:-Calculus1:-ApproximateInt\left(f\left(x\right), x=0..1, method=midpoint, partition=n\right)$$

$$\lim_{n\to\infty} M\left(n\right)$$

$$\frac{1}{2}\,\mathrm{erf}\left(1\right)\sqrt{\pi}$$

$$evalf\left(\%\right)$$

$$0.7468241330$$

It follows that the approximate value of the integral using midpoint Riemann sums is 0.7468241330.

Using this feature we not only evaluate an integral that a student in an Integral Calculus class cannot easily integrate but also visualize that integral as the area under the curve $y = e^{-x^2}$, above the *x*-axis and between *x* = 0, and *x* = 1.

2. Definite Integrals for functions of two variables

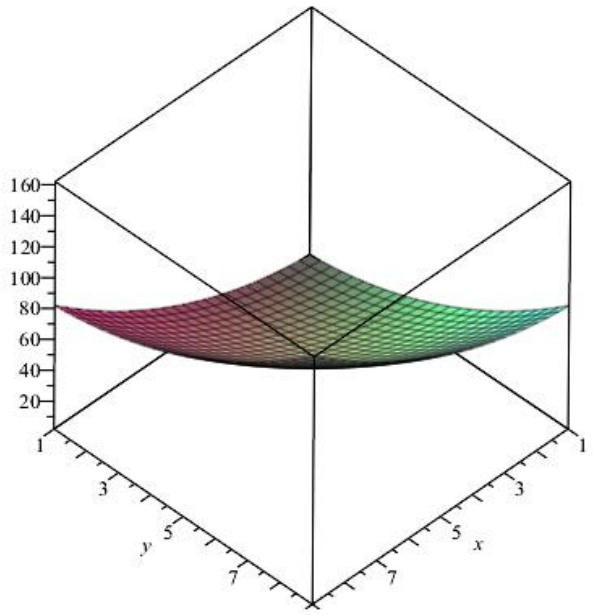
The definite integral of a nonnegative function (of two variables) over a closed rectangle is defined as the volume bounded by the surface, the *xy*-plane and the end points of the rectangle. The volume can be obtained by constructing rectangular blocks and finding the volume of these blocks (Riemann Sums) [11]. Students are able to see what happens as the number of blocks increases. This concept can be extended for functions that are not necessarily nonnegative. We will illustrate using the midpoint Riemann sum how to calculate definite double integrals [11].

Consider the function $f(x,y) = x^2 + y^2$. As in the previous example, we will first define and sketch the graph of this function using *Maple* commands.

$f := (x,y) \to x^2 + y^2$

$$(x,y) \to x^2 + y^2$$

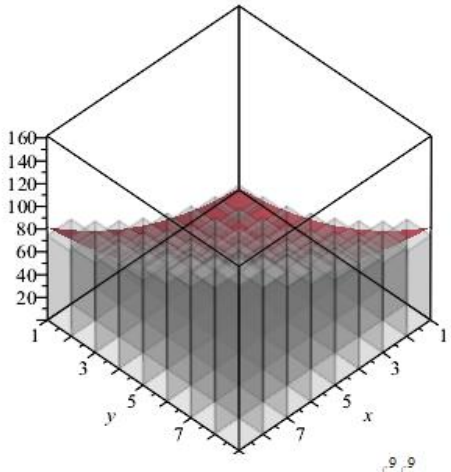
$plot3d(f(x,y), x = 1..9, y = 1..9, axes = box)$



Now we will write *Maple* code to generate an animation of midpoint Riemann sum approximations of the double integral $\int_1^9 \int_1^9 (x^2 + y^2) dx dy$ to generate visual representations of the midpoint Riemann sum approximations. In order to do so we need to activate the $with(Student[MultivariateCalculus])$ library.

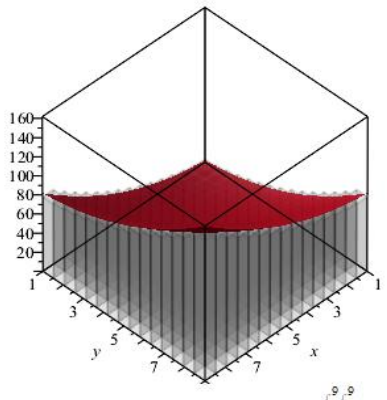
$$with\left(Student\left[MultivariateCalculus\right]\right):$$

$$ApproximateInt\left(f\left(x,y\right),x=1..9,y=1..9,method=midpoint,output=plot,\right.$$
$$\left.partition=\left[8,8\right],axes=box,prismoptions=\left[color=grey\right]\right)$$

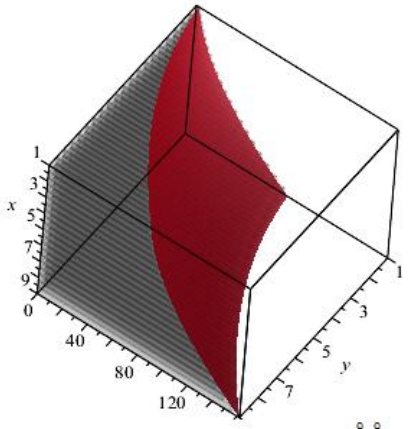


Using a midpoint Riemann sum, an approximation of $\int_1^9\int_1^9 f(x,y)\,dy\,dx$,
where $f(x,y)=x^2+y^2$. Actual value: 3882.7. Approximate value: 3872.0. Grid: 8 × 8

$$ApproximateInt\left(f\left(x,y\right),x=1..9,y=1..9,method=midpoint,output=animation,\right.$$
$$\left.frames=15..20,partition=\left[8,8\right],axes=box,prismoptions=\left[color=grey\right]\right)$$



Using a midpoint Riemann sum, an approximation of $\int_1^9\int_1^9 f(x,y)\,dy\,dx$,
where $f(x,y)=x^2+y^2$. Actual value: 3882.7. Approximate value: 3879.6. Grid: 15 × 15



Using a midpoint Riemann sum, an approximation of $\int_1^9\int_1^9 f(x,y)\,dy\,dx$,
where $f(x,y)=x^2+y^2$. Actual value: 3882.7. Approximate value: 3881.0. Grid: 20 × 20

The images are obtained using different frames and Rotation plot option of *Maple*. Note that we have used regular sized rectangular blocs. Next, we will use the *Maple* command below to find an approximate value of the integral

$$ApproximateInt\left(f\left(x,y\right),x=1..9,y=1..9,method=midpoint,partition=[8,8]\right)$$

3872.000000

Hence, an approximate value of the integral $\int_1^9\int_1^9\left(x^2+y^2\right)dxdy$ is 3872.000000. Next we will find the value of the integral.

We recall that by definition

$$\int_1^9\int_1^9\left(x^2+y^2\right)dxdy=\lim_{n\to\infty}\left(\lim_{m\to\infty}\sum_{j=1}^{n}\sum_{i=1}^{m}f\left(xibar,yjbar\right)\left(\frac{9-1}{m}\right)\left(\frac{9-1}{n}\right)\right)$$

where $xibar=1+\frac{(9-1)(i-0.5)}{m}$ and $yjbar=1+\frac{(9-1)(j-0.5)}{n}$. We will now evaluate the limit using *Maple.*

$m:='m':n:='n':xibar=1+\frac{(9-1)(i-0.5)}{m}:yjbar:=1+\frac{(9-1)(j-0.5)}{n}:$

$$\lim_{n\to\infty}\left(\lim_{m\to\infty}\sum_{j=1}^{n}\sum_{i=1}^{m}f\left(xibar,yjbar\right)\left(\frac{9-1}{m}\right)\left(\frac{9-1}{n}\right)\right)$$

$$3882.666666$$

Therefore $\int_1^9\int_1^9\left(x^2+y^2\right)dxdy = 3882.666666$

To find the value of $\int_1^9\int_1^9\left(x^2+y^2\right)dxdy$ using the *Maple* command *int* to evaluate integrals

$int\left(f\left(x,y\right),\left[x=1..9,y=1..9\right]\right)$

$$\frac{11648}{3}$$

$evalf\left(\%\right)$

$$3882.666667$$

Thus, as it was expected the output of the *Maple* command *int* gives the same number as using the limits of midpoint Riemann sums.

Notice that students can use the images generated by *Maple* to understand why the integral $\int_1^9\int_1^9\left(x^2+y^2\right)dxdy$ depicts the volume under the surface $f\left(x,y\right)$ over the square $\left[1,9\right]\times\left[1,9\right]$ is also a volume.

3. Bizarre Functions

In a calculus class, students are introduced to the concept of continuity. They learn how to graph functions by hand and with a graphing calculator. They understand how to recognize points of discontinuity (if any) on graphs as the holes, breaks or jumps. Most of the time, students will be able to achieve this task. But there are "exotic functions" [12] which do not fit the traditional mold and could present a challenge to the students. We will give an example of such a function. We will describe a function that is continuous at each irrational number in the interval $\left(0,1\right)$ and discontinuous at each rational number in $\left(0,1\right)$.

Consider the function:

$$f(x)=\begin{cases}\frac{1}{m} & \text{if } x=\frac{n}{m} \text{ is rational in lowest terms.}\\ 0 & \text{if } x \text{ is irrational}\end{cases}$$

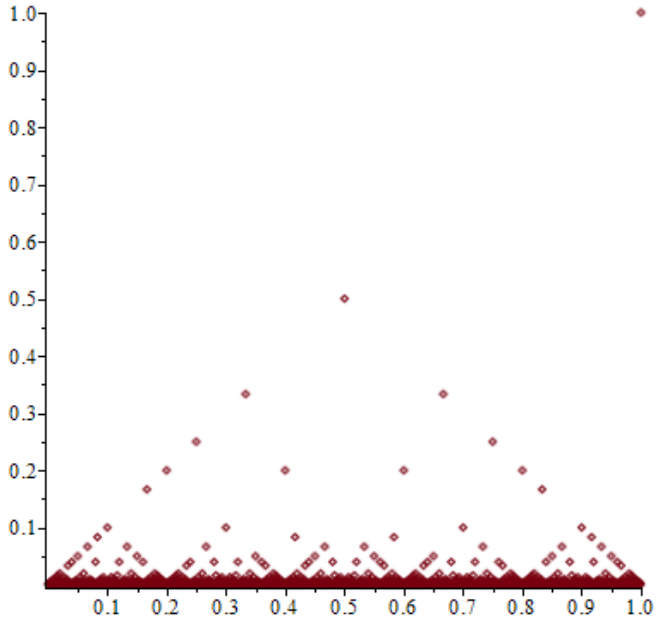
Of course it would be impossible to sketch the graph of the function f by hand. On the other hand, using *Maple*, students can have a very good image of the graph of this bizarre function. Below are the *Maple* code and a *Maple* representation of the function f using special sequences.

$with(plots):$

for m **from** 1 to 1500 do $r := seq\left(\left[\left[\frac{n}{m}, \frac{1}{denom\left(\frac{n}{m}\right)}\right], n=1..m\right]\right)$ **end do**:

$p := plot([r], style = point):$

$display(p)$



We will now prove the claim that the function f is continuous at each irrational number in the interval $(0,1)$ and discontinuous at each rational number in $(0,1)$.

(a) We will first show that $f(x)$ is discontinuous at all rational numbers in the interval $(0,1)$

Let x_0 be a rational number in the interval $(0,1)$. Let $\langle x_n \rangle_{n=1}^{\infty}$ be a sequence of irrational numbers that converges to x_0 then $f(x_n)=0$ for all n, which means that $\lim_{n\to\infty} f(x_n)=0$. It is however clear that $f(x_0)>0$ since x_0 is rational and therefore

$f(x_n) \neq f(x_0)$ which implies that $f(x)$ is discontinuous at x_0 and more to the point it is discontinuous at all rational numbers in the interval $(0,1)$.

(b) We will now establish that $f(x)$ is continuous at all irrationals. Let x_1 be an irrational number in the interval $(0,1)$. For any $\varepsilon > 0$, there is a k such that $\frac{1}{k} < \varepsilon$ (Archimedean property). Since there are only a finite number of rationals in $(0,1)$ with denominator smaller than k, we can find a real number $\delta > 0$ such that all the (reduced) rationals in the interval $(x_1 - \delta, x_1 + \delta)$ have a denominator $\geq k$. It then follows that if x is in $(0,1)$ and $|x - x_1| < \delta$ then $|f(x) - f(x_1)| \leq \frac{1}{k} < \varepsilon$. This implies that f is continuous at the irrational number x_1 and in fact at every irrational number in the interval $(0,1)$.

From the above example students get a graphical view of an abstract function and are able to make valid conclusions on its continuity by visualization.

Conclusion

Technology provides tools that can enhance teaching but does not give all the answers and must be used carefully. In particular, technology has built-in limitations; it can also be a distraction if it is difficult to learn or involves extensive time commitment. Currently classes are given extended lab time to facilitate the learning of technology. In addition, the technology used by most colleges has a shallow learning curve and is constantly improving. The examples presented in our paper demonstrate the power of some of the technological tools we have used to facilitate student learning. Based on our experiences, we find that visualization in one, two, and three dimensions, both from a mathematical and programming perspective, plays a critical role in enhancing students understanding and their abilities to produce better quality work. This also equips students with the requisite tools to function well in industry.

References

[1] Visual Teaching Alliance, http://www.visualteachingalliance.com/

[2] Naps, T.F., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. and Velázquez-Iturbide, J.A. (2002) Exploring the Role of Visualization and Engagement in Computer Science Education, Working group reports from ACM SIGCSE/ITiCSE on Innovation and technology in computer science education.

[3] Korhonen, A. and Malmi, L. (2000) Algorithm Simulation with Automatic Assessment, Proc. 5th ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education.

[4] Dann, W., Cooper, S. and Pausch, R. (2000) Making the Connection: Programming with Animated Small World, Proc. 5th ACM SIGCSE/SIGCUE ITiCSE Conf. on Innovation and technology in computer science education .

[5] Cooper, S., Dann, W. and Pausch, R. (2000) ALICE: A 3-D Tool For Introductory Programming, Journal of Computing Sciences in Colleges, 15.

[6] Rosenberg, J. and Kölling, M. (1997) Testing Object-Oriented Programs: Making it Simple, Proc. 28th ACM SIGCSE technical symposium on Computer science education.

[7] Dann, W., Cooper, S. and Pausch, R. (2001) Using Visualization to Teach Novices Recursion, Proc. 6th ACM SIGCSE Conf. on Innovation and technology in computer science education.

[8] Cooper, S., Dann, W. and Pausch, R. (2003) Using Animated 3D Graphics To Prepare Novices for CS1, Computer Science Education, Routledge, part of the Taylor & Francis Group, 13, 3-30.

[9] Cooper, S., Dann, W. and Pausch, R. (2005) Learning to Program with ALICE (United States of America: Pearson Prentice Hall).
[10] Taraporevala, A., Benakli, N., and Singh, S. (2012) Visualizing Calculus by Way of Maple (McGraw-Hill)
[11] Rogawski, J (2011) Calculus: Early Transcendentals,, 2nd ed (W.H.Freeman)
[12] Rudin, W. (1976) Principles of Mathematical Analysis, 3rd ed (McGraw-Hill)