
Using Question Mark Perception v3 for testing mathematics

Daniel Nichols
Brunel University

daniel.nichols@talk21.com



Martin Greenhow
Brunel University

martin.greenhow@brunel.ac.uk



Acknowledgements

This work has been supported by the LTSN Maths Stats & OR Network and by the Learning Technology Development Unit at Brunel University.

Professor David Hewitt of Monash University, Australia and Dominic Smith and Dalvinder Jathol of Brunel University provided valuable input into the development of the questions styles.

A very full description of Perception can be found at Question Mark's Web site; reviews by Dempster (1998), Strickland (1999), Paterson (2001) may also be seen on the Web, and a good set of "driving instructions" has been produced by the Learning Technology Development Unit (LTDU) at the University of Hertfordshire.

This report therefore focuses on various extensions that we have made at Brunel to enhance the functionality of Perception for testing of mathematics and statistics. A long-term aim is to replace and enhance the Mathletics suite of online objective tests, see Greenhow (2002). Whilst Mathletics was (and still is) a very successful workhorse, it was written in QM Designer, meaning that questions are hard-wired in a Windows-based proprietary format. Perception offers a much more future-proofed and interoperable authoring system (written within QTI IMS specifications, see Sclater and Low (2002) that can deliver questions via Windows or the Web, possibly linked to a VLE (Virtual Learning Environment) such as Blackboard or WebCT. This means that answer files from a Perception-authored test can also be managed by the VLE (so that a lecturer could assess by using a standard set of Perception tests without using Perception itself).

These are, of course, considerable advantages, but for assessing mathematics the real advantage of using Perception probably lies in its extensibility. Rather than directly translating Mathletics into a set of hard-wired questions as offered by the standard Perception templates and question wizard, what is needed is to enhance the functionality of the questions by writing *questions styles* that are "realised" at runtime by dropping random parameters into the questions stem, key, distracters and feedback. It is important to realise that questions can be algebraically identical, but pedagogically different (for example, there may be additional problems with negative or fractional parameters). It is therefore necessary to underpin the writing of good question styles by giving (see Greenhow(2000)):

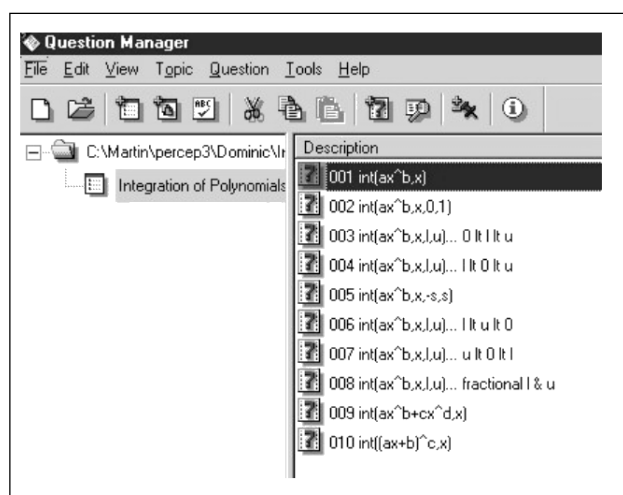
- a clear specification of the tested skill; for example is the question about doing the integration, or applying the limits?
- a clear specification of any assumed skills; does the student need to know how to simplify logs, for example?
- an algebraically codified set of mal-rules leading to the question distracters.
- good feedback resulting from the knowledge that if a student has selected a particular distracter, he/she has probably used the particular mal-rule that generated it.

As seen below, the random parameters are used within MathML. So far we have developed this for multi-choice, multi-response and numerical input questions. Perception adds in WebEQ to the browser (Internet Explorer) to interpret the MathML syntax. In contrast to using graphics (often gifs are used within www pages) for equations, the code can be copied and edited to form other questions. The approach can also be used for diagrams using scalable vector graphics (SVG) syntax; again this interpreted plain text can include random parameters so that the diagram changes at runtime.

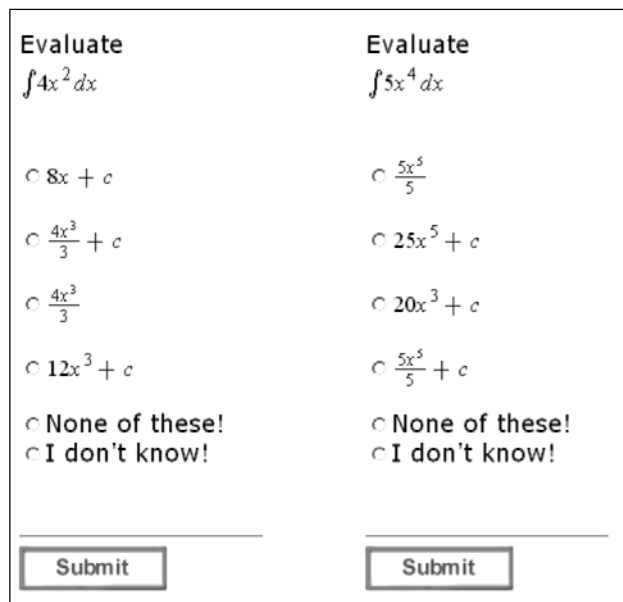
This work is not finished by any means, but the *questions styles* are likely to be of value to others writing assessments in mathematics and tests involving mathematics.

MathML and random parameters

The following screen shot shows a possible breakdown of the integration of ax^b for various orderings of positive or negative lower and upper limits (the HTML tag identifier "<" cannot be used to represent "less than" so "lt" is used instead). This illustrates what is meant by a question style; each question is designed to test different skills; eg integrating from $-s$ to s tests whether students can spot the use of symmetry in this simple example.



For the first question style we have the following realisations:



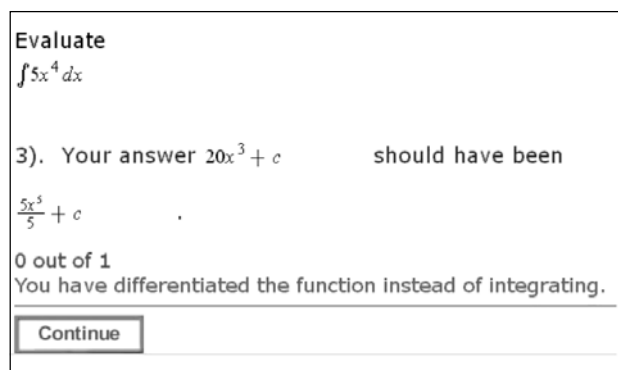
The random parameters of this question are generated by the following statements within the Question Mark Language (QML) code:

```
A = Math.ceil(5*Math.random()); // 1-5
B = 1 + Math.ceil(3*Math.random()); // 2-4
if (A==1) {A=""}; // suppresses display of 1x
```

and these are used within, for example, the stem as:

```
e1="<PARAM NAME='eq'
VALUE='<math><mrow><mrow><mo>&int;</
mo><mrow><mi>" +A+"</mi><msup><mrow><mi>x</mi></
mrow><mrow><mi>" +B+"</mi></mrow></msup></
mrow><mo>d</mo><mrow><mi>x</mi></mrow></mrow></
mrow></math>'></APPLET>"
```

The syntax here is an edited form of MathML, as generated by a high-level authoring package such as WebEQ, MathType etc. Similar syntax generates the key and distracters (according to the chosen mal-rules); the random parameters are taken through to the feedback screen; a student who is confused about differentiation/integration for the second realisation above would see the following screen:



Note that in this example, we have highlighted the problem of display that can occur when using random parameters. Whilst $5x^5/5$ is not incorrect, it would have been better as simply x^5 . We have implemented scripts (in SCRIPTS.FORMAT – see below) to detect and correct for such things as uncanceled fractions, or unconventional display of constants and symbols, as in $x + -1$, $1x$ or x^1 . Also note that for longer feedback, it might be better to have pop-up windows; for example a "Hint" button could give the general theory window, whilst a button entitled "This Example" would show that theory applied to the actual question. We are currently developing this.

Scripts for functions and display

A question may be authored using the wizard, or by editing the underlying QML syntax, as in the above section. Perception then combines this QML file with various formatting files to generate HTML code itself. The file SCRIPTS.FORMAT can therefore contain user-defined code for mathematical or display functions. Whilst some functions, such as Math.random() above, are built into Java and JavaScript, it is useful to extend this eg the factorial function; `getfact(9)` would return 362880 by implementing:

```

ALL_QUESTIONS=
<script>
function getfact(x) {
  if (x <= 1)
    return 1;
  return x * getfact(x-1);
}
</script>

```

Scalable Vector Graphics

Scalable vector graphics (SVG) are images generated from xml coding. These graphics can be produced using a high-level authoring package such as WebDraw and stored as separate files, but the main advantage is that the code can be included in any Web page or Perception question.

The code below shows the code for a simple rectangle:

```

<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/
  DTD/svg10.dtd">
<svg width="300" height="300">
  <rect x="67" y="78" width="189" height="108"
    style="fill:none;stroke:rgb(0,0,0);stroke-
    width:4"/>
</svg>

```

The svg coding firstly specifies the size of the canvas in which to place the desired object, in this case 300 by 300 pixels. Next the code details information needed to generate the rectangle, both its position and its style. The rectangle above has its top left corner at (67, 78) the width and height of this shape is 189 and 108 pixels respectively. The final line of script defines how the rectangle appears on screen, the border colour is black and of width 4, the shape has no fill colour.

It is possible to introduce randomly generated integers into these vector based graphics, for instance in the simple case of a rectangle the width and height of the shape can be defined by random numbers.

Taking the use of random numbers a step further, scalable vector graphics can be utilised within questions in order to provide accurate diagrams. Two such cases are in the creation of diagrams for statistics and trigonometric questions.

Taking advantage of the data produced from random data tables (these call a user-defined function create_unique_random_array from SCRIPTS.FORMAT that produces a random length table of unique random numbers) the information can be passed into svg scripting to produce a chart of the data. By scaling and positioning the data points, a scatter diagram can be drawn showing the data from the random data table.

The axis for the graphs are drawn by two lines, each of the data points use the random data and are plotted on the graph. The amount of data in the tables is randomly drawn from an array, this takes sets of values up to 12 (x, y) pairs.

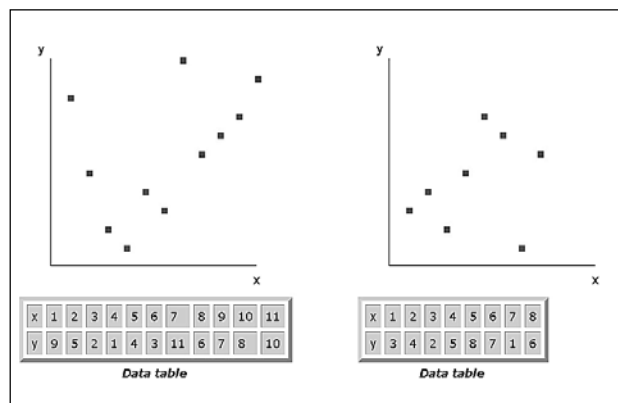
```

i = 1;
while (i < narray) {
  var obji = new_rect1(px1+arraya[i]*30 , py2-
  arrayb[i]*30, 5, 5);
  array.push( obji );
  canvas.appendChild( obji );
  i++;
}

```

For each item in the array of data (1, 2, ... i) the data is taken from the table and then a small square is drawn on the plot. Each square (using the basic rectangle coding) is placed at a specific x, y coordinate.

The data tables and scatter diagrams below show realisations of the random data and plots.



The trigonometric questions below again use svg coding. A simple question may entail a student to calculate the height of a tower, the only data provided being an angle of projection and the distance between the projection point and the base of the tower. Each question, generated at runtime, is produced by random integers generating varying distances and heights of the tower.

The script begins by drawing a triangle from three randomly determined points, (x1, y1), (x2, y2), (x3, y3).

```

var px1 = 360-5*Math.ceil(10*Math.random());
// x1 coordinate
var px2 = px1; // x2 coordinate (same as x1)
var px3 = px1-24*Math.ceil(10*Math.random());
// x3 coordinate (x1 - value)

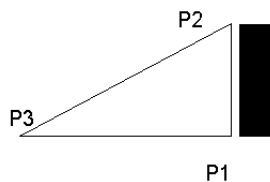
var py1 = 360-5*Math.ceil(10*Math.random());
// y1 coordinate
var py2 = py1-30*Math.ceil(10*Math.random());
// y2 coordinate (y1 - value)
var py3 = py1; // y3 coordinate (same as y1)

```

The next few lines specify positional information of the tower.

```
var x1 = px2+15; // 15 pixels from x2
var y1 = py2;    // same as y2
var w1 = 50;    // width = 50 pixels
var h1 = py1-py2; // height = y1-y2
```

Using this information we have a basic view of the question.



On the diagram there needs to be specific text for the user, an angle at P3 and the length of P3 to P1. The angle (in degrees) is calculated by the scripting below.

```
var anglep2 = (180 * Math.atan((py1-py2)/
(px1-px3))) / Math.PI;
```

The labels of angle and length can then be attached to the picture again using svg coding. The rounded angle (obj5), calculated above, is placed at coordinate (px3-40, py3). The distance between projection point and tower base is placed halfway between these two points.

```
var obj5 = new_textangle (px3-40, py3,
Math.round(anglep2) );
array.push( obj5 );
canvas.appendChild( obj5 );
```

```
var obj6 = new_text ((px1-px3)/2 + px3,
py3+30, px1- px3, "middle");
array.push( obj6 );
canvas.appendChild( obj6 );
```

The images opposite show two realisations of the same trigonometric question.

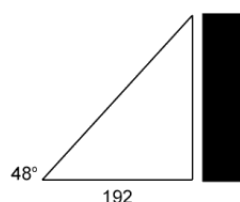
There are many possible applications of using SVG in randomised mathematical questions, for example in geometry, mechanics etc. In statistics the application may be even more useful; for example, runtime graphics like histograms and pie charts could also be displayed using SVG in a similar way to the above scatter diagrams.

Conclusion

The basic structure of Perception make it ideal for secure testing of many subjects; for tests involving the use of mathematics, the above approach to structuring questions and the MathML and SVG extensions should enhance Perceptions functionality in a very useful way.

References

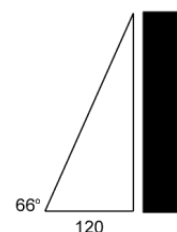
- Dempster J(1998) *Web-based assessment software: Fit for purpose or squeeze to fit?*
<http://www.warwick.ac.uk/ETS/interactions/vol2no3/dempster.htm>
- Greenhow M(2000) *Setting objective tests in mathematics with QM Designer* LTSN Maths, Stats and OR Newsletter, Feb 2000
<http://ltsn.mathstore.ac.uk/newsletter/feb2000/pdf/qmdesqn.pdf>
- Greenhow M(2002) *Answer files – what more do they reveal?* <http://ltsn.mathstore.ac.uk/articles/maths-caa-series/jan2002/index.shtml>
- LTDU *Question Mark Perception: Tutorials and Issues of Best Practice* <http://www.herts.ac.uk/ltdu/tutorials/qmark/index.html>
- Paterson J(2001) *Question Mark Perception v3*
<http://www.scrolla.hw.ac.uk/QMreport.doc>
- Question Mark Perception
<http://www.questionmark.com/uk/perception/>
- Slater N and Low B(2002), *IMS Question and Test Interoperability: An Idiots Guide*: CETIS Assessment Special Interest Group, Version 0.5, March 2002 available from <http://www.scrolla.hw.ac.uk/news.html>. See also <http://www.cetis.ac.uk/>
- Strickland P (1999) *Review of Question Mark Perception*, Maths&Stats Aug 99
<http://www.bham.ac.uk/ctimath/reviews/aug99/perception.pdf>



How high is the tower?
(Select the result with the nearest height)

- 229
- 208
- 223
- 224
- None of these!
- I don't know!

Submit



How high is the tower?
(Select the result with the nearest height)

- 295
- 286
- 265
- 290
- None of these!
- I don't know!

Submit