Application of Progressive Discovery as a Teaching Aid Using MathDisk

KUMAR, Ajit
ajit72@gmail.com
Department of Mathematics,
Institute of Chemical Technology,
Mumbai - 400 019, India.

LEE, Sang-Gu[1]
sglee@skku.edu
Department of Mathematics,
Sungkyunkwan University,
Suwon, 440-746, Korea.

SHARFUDEEN Ashraf
thahir@mathdisk.com
MathDisk Technologies,
242 Peters Road Royapettah, Chennai, India

Abstract

The pedagogical benefits of using computer aided dynamic geometry software system in classrooms are well understood among educators. MathDisk, the latest entrant into this fray, has all the essential aspects one could expect from an Interactive Mathematical Software. MathDisk also goes beyond the conventional feature set that defines this class of software and brings in proven learning techniques found in other disciplines in mathematics teaching. The main aim of this paper is to describe one such technique called "progressive disclosure", a concept extensively introduced in User Interface Design to improve the readability and accessibility of software. Progressive discovery or disclosure is an informational presentation pattern, where we seek to focus the attention of the audience to the point that you we are making, rather than presenting them with too much distracting information. The concept itself is by no means an innovation, however MathDisk provides a simple and systematic approach to apply this concept to teaching mathematics. This paper illustrates this concept using two concrete examples, (both in 2D and 3D), the tools available within MathDisk to achieve progressive discovery, and how to effectively apply this as a teaching aid. Before we embark on our main goal, we give a brief introduction of MathDisk.

[1] Corresponding author

1. A brief introduction to MathDisk

MathDisk [1], which is designed specifically for pedagogical purposes, can help students to experiment and explore mathematics, both online and offline. What distinguishes MathDisk from the rest of the numerous other math tools currently available is its approach and philosophy. In almost all other math tools, trying to do anything beyond graphing a simple 2D function requires writing code using the tool's own programming language. MathDisk, on the other hand, allows the users to express equations of any type, just as they see it in their textbooks. Thus, there is no artificial layer that stands between the user and the native math expressions. Students never feel any disconnect, as they can instantly recognize and correlate the content with their textbook materials. The equal emphasis to both the symbolic and visual representation of mathematics makes MathDisk an ideal tool to create online interactive textbooks for teaching mathematics. MathDisk also uses "integrated rigid body dynamics," which can help students understand the abstract nature of mathematical structures using simulated physical objects. MathDisk can help teachers to build mathematical and physical models and simulations to improve the cognitive abilities of students in mathematics and physics. Advanced users of MathDisk can also use scripting based on the syntax of processing languages to produce amazing mathematical and physical models, enabling MathDisk to become truly open ended and infinitely extensible. More details on scripting in MathDisk can be obtained from MathDisk Scripting website [6]. MathDisk also allows users to share their individual resources and their entire working space over the web, a key feature in today's interconnected world.

2. What is progressive discovery?

Anyone who has seen a PowerPoint presentation or any other slide show tool (viz. Beamer, Impress, Prezi, Google Presentation, SlideRocket) where bullet points or images appear in a certain sequence as the presenter presses enter is already familiar with "progressive discovery" [3]. It could just be three or four bullet points, but still presenters resort to showing them one by one, because we have long since realized that too much information at the same time clutters thinking and disturbs focus. Using this same principle, a given mathematical model can be progressively revealed in a step-by-step fashion using MathDisk. Each step could display or animate a portion of the model in a certain logical order, eventually revealing the entire model.

The first generation of mathematical software allowed users to plot and to visualize mathematical constructs. Then, the ability to animate the visual representation of the math constructs was introduced, which was further followed by the ability to interact directly with the visual itself. Today, by employing the concept of "progressive disclosure," MathDisk adds another key learning technique and dimension to the sphere of technology enabled math education. The following sections use two examples: (i) the Gram-Schmidt Orthogonalization Process [4] and (ii) the Nine-Points Circle [2] to demonstrate the effectiveness of using progressive disclosure as a learning aid in enhancing the geometric and conceptual understanding of these concepts. MathDisk also allows users to add text and audio with any models.

If the student were to use interactive dynamic software, he/she can change the orientation and the length of the vector to see how the theorem holds true for different conditions and convince

himself/herself of the validity of the theorem. However, the interplay between fundamental vector operations, such as vector addition, dot product and cross product, that underline this theorem are not immediately oblivious to the observer unless we label or annotate each part of the image separately. Even if we were to do this, the order in which these vector operations should be applied could not be made apparent, which is equally critical in understanding the theorem. The "Progressive Script" built into MathDisk helps us to deconstruct each operation and to progressively reveal the entire model without compromising the mathematical integrity of the theorem.

Example 1. (Nine-Point Circle)

The first example we deal with, is a nine point circle [2]. For example, one can create a nine point circle using the dynamic features of MathDisk, similar to GeoGebra [5] as shown in Figure 1.

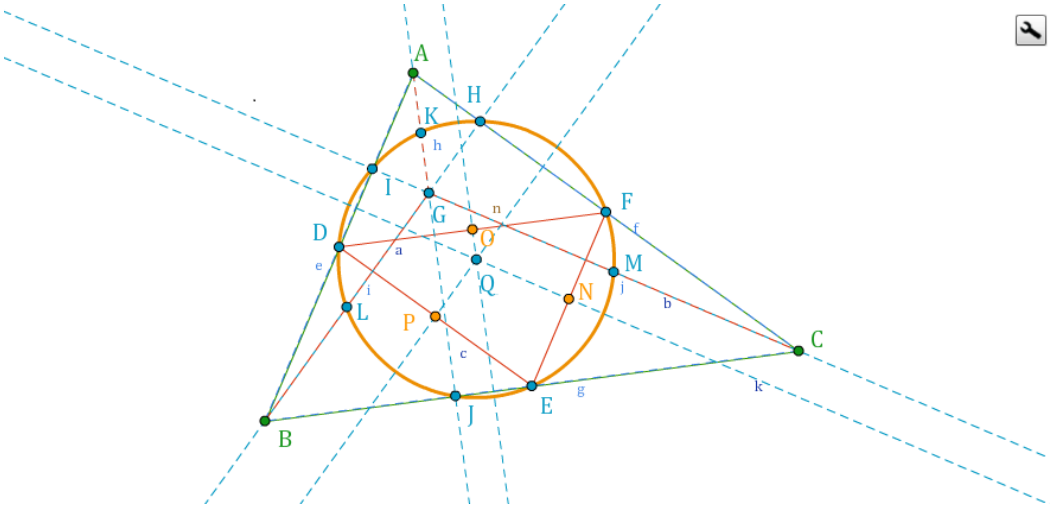

Figure 1: Nine Point Circle.

Once this model is built, we can use the sequence script of MathDisk to show, hide, and draw each object in whichever order we want and finally, the model can be played and shown to students. For example, one can get a final model as shown in Figure 2.

The final video output of the nine point circle can be seen from following link
http://www.mathdisk.com/pub/sheik.mohammed/worksheets/Nine_Point_Circle
The complete script of the step by step nine point circle is also included in the appendix A1.

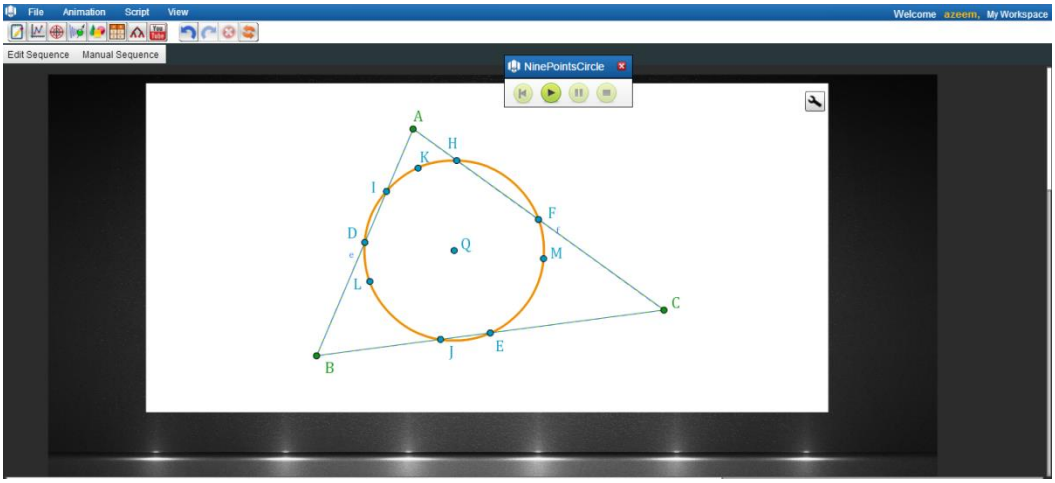

Figure 2: Nine point circle model using MathDisk.

Example 2. (Gram-Schmidt Orthogonalization)

The second example of progressive discovery in MathDisk is the Gram-Schmidt orthogonalization [4] of a set of three linearly independent vectors in the space. Figure 3 shows the model created for orthogonalization of two linearly independent vectors. Figure 4 shows the model created for orthogonalization of three linearly independent vectors. Once this model is created in MathDisk, each step in the orthogonalization process can be shown by playing this animation. Thus, students can see the geometry behind the underlying concepts.

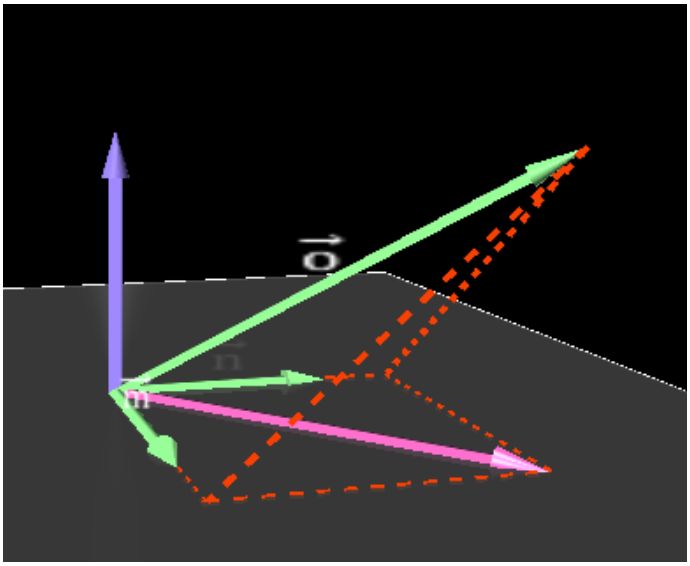

Figure 3: Gram-Schmidt orthogonalization with two vectors.

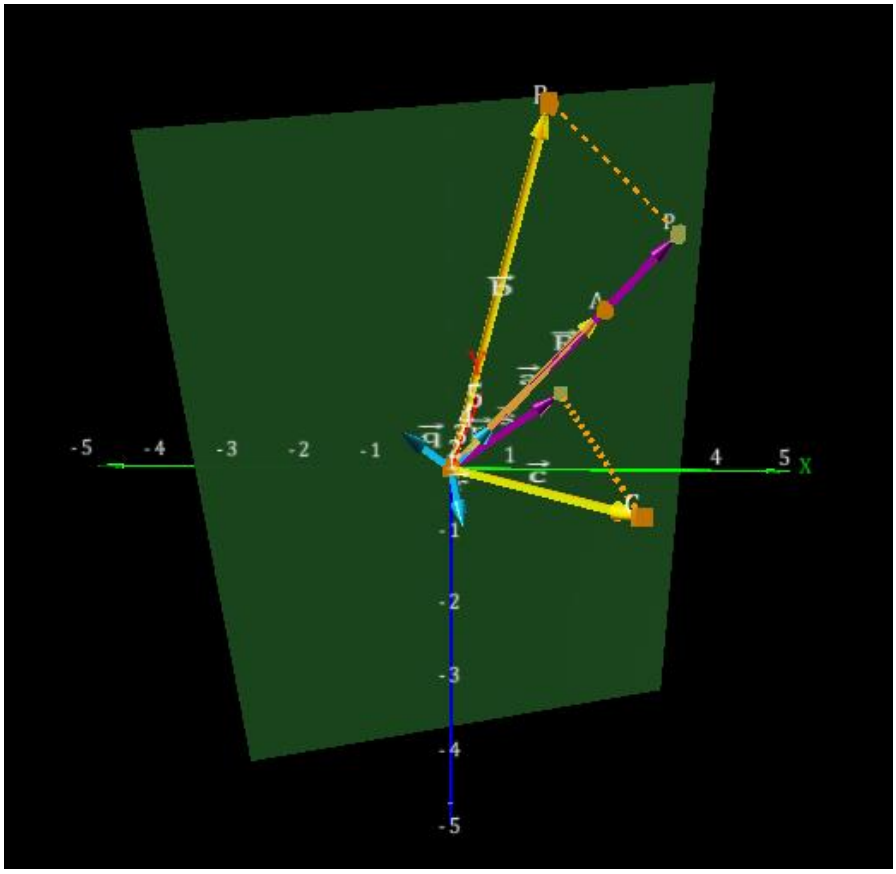

Figure 4: Gram-Schmidt orthogonalization with three vectors.

Use the following link to play the progressive discovery of Gram-Schmidt orthogonalization process.

http://www.mathdisk.com/pub/sheik.mohammed/worksheets/Gram_Schmidt_using_Sequence

The complete script of the Gram-Schmidt orthogonalization is also included in the appendix A2.

3. Syntax and structure of progressive script

A wide range of progressive effects required to reveal or highlight a given mathematical model is condensed into a set of a few basic commands that can be applied in different combinations. The basic structure of the command is captured in the following EBNF (Extended Backus Naur Form) railroad diagram (Fig.5).

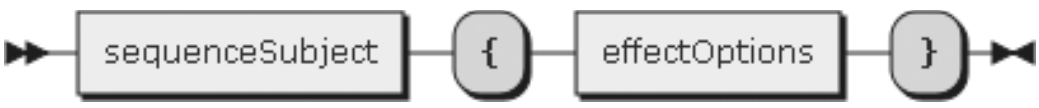

Figure 5: EBNF railroad diagram.

The "sequence subject" is the type of the element that we want to apply an effect to in order to highlight or reveal an interesting phenomenon. The sequence subject is further classified into the following categories as shown below in Figure 6.

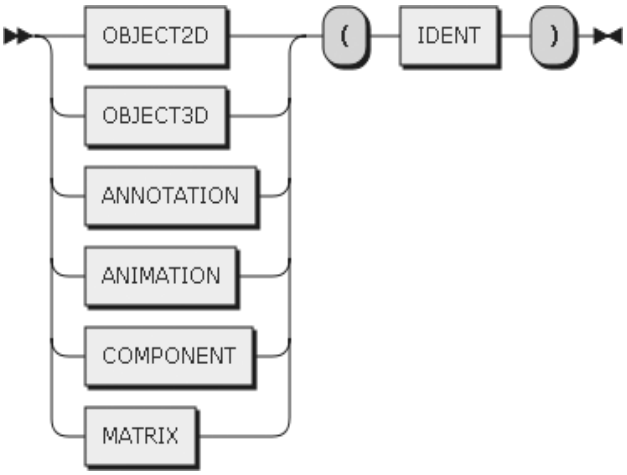

Figure 6: Classification of sequence subject.

Examples of OBJECT2D are 2D graphical primitives such as Point, Vector, Line, Circle, etc. In order to implement a progressive discovery model, user can apply effects, such as changing colors, interpolating between two points, and animating certain properties like font style, line thickness etc. as shown in the Figure 7. MathDisk abstracts the entire set of operations required to employ progressive disclosure into the following options.

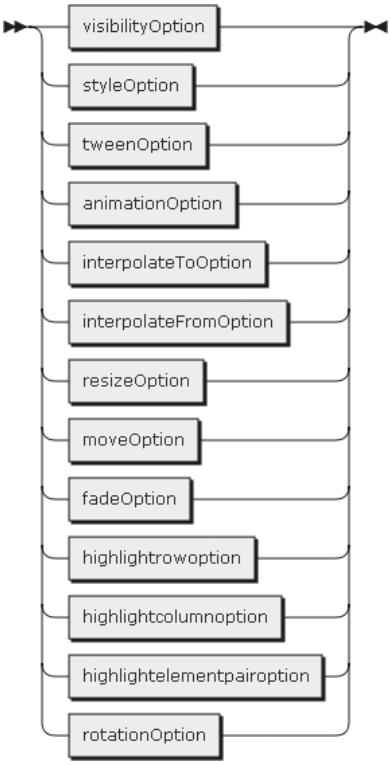

Figure 7: Graphical Primitives

One can click on any primitives in Figure 7 (for example, InterpolateToOption) in order to see further options and how to use them. Figure 8 shows this process specifically for InterpolateToOption.

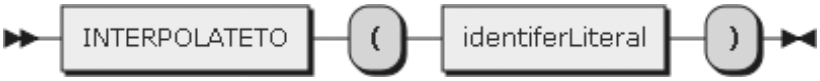

Figure 8: InterpolateTo Option example syntax.

If there are two 3D points, namely A and B, and if A needs be interpolated from its position to B, the required syntax would be as shown in Figure 9.

```
object3d(A){interpolateto(B)};
```

Figure 9: Sequence script for interpolation of point A from A to B.

Similarly, in order to animate or draw a 2D vector, the following commands as shown in Figure 10 are used.

```
object2d(V_m){tween()};
```

Figure 10: Sequence script for drawing a 2d vector.

As we can notice, Object2D is the subject type, and the value V_m is the notation used within MathDisk for referring to a vector m. The tween effect is one of many options that instruct the vector to animate itself. In particular, user can see how the vector is drawn between two points. A series of such statements can be written which will then be made to execute in the same order as they are written.

Though all lists of commands follow the same pattern set, beginners still do not have to memorize any syntax or structure, as MathDisk provides a very convenient way to generate this script. For example, when we select any element using the mouse such as a line, vector or point, it displays a context bar item called Sequence Script, and opening it shows the different commands the elements can be subjected to during progressive discovery. Users only have to copy the command and change certain parameters based on their needs.

4. Is this a macro?

Some tools allow users to replay the set of actions that leads to the creation of the model. However, in most cases, the steps used in constructing the model might not be listed in the appropriate order

in which we want the students to understand the model. So, just by simply replaying the actions as a macro does not serve this objective. MathDisk goes a step beyond this and allows a worksheet created by a user to be progressively discovered in a different order (Using Sequence Script) by another user as he sees it and which fits into the given context. MathDisk uses Progressive discovery/disclosure to demystify complex mathematical concepts. The consistent usage of this technique reinforces the basic constructs and helps to demonstrate that the underlying concepts behind many advanced topics are essentially an interplay of same fundamental mathematical operations. Application of Progressive

5. Conclusions

Discovery not only improves students' comprehension, but in the long run, it also increases their retention capability. This unique feature of MathDisk is very useful tool and can make mathematics teaching more interactive.

References

[1] MathDisk, www.mathdisk.com (accessed in July, 2014)
[2] Brand L. "The Eight-Point Circle and the Nine-Point Circle", Amer. Math. Monthly 51, 84-85, 1944
[3] Andrew T. Progressive disclosure, (accessed in July, 2014)
http://www.artsjournal.com/artfulmanager/main/progressive_disclosure.php
[4] Kumaresan S. Linear Algebra-A Gemometric Approach (p.181), Prentice Hall of India, 2010
[5] GeoGebra, www.geogebra.org (accessed in July, 2014)
[6] MathDisk Scripting, http://scripting.mathdisk.com/

Appendix

A1. Nine-Point Circle Scripts:

```
group[component(Image24){fade(fadein,0)},
component(Graphsheet2d23){resize(TopToBottom,0)},
component(Annotation9){show()}];
component(Annotation9){hide()};
component(Annotation7){move(leftToRight,0)};
group[object2d(A){show()},object2d(B){show()},
object2d(C){show()}];
object2d(Poly1){tween()};
component(Annotation7){hide()};
component(Annotation8){move(leftToRight,0)};
object2d(D){show()};
object2d(E){show()};
object2d(F){show()};
component(Annotation8){hide()};
component(Annotation11){move(leftToRight,0)};
object2d(a){tween()};
object2d(b){tween()};
object2d(c){tween()};
component(Annotation11){hide()};
component(Annotation12){move(leftToRight,0)};
object2d(G){show()};
object2d(H){show()};
object2d(I){show()};
object2d(J){show()};
 component(Annotation12){hide()};
component(Annotation13){move(leftToRight,0)};
object2d(a){hide()};
object2d(b){hide()};
object2d(c){hide()};
group[object2d(h){tween()},object2d(i){tween()},
object2d(j){tween()}];
component(Annotation13){hide()};
component(Annotation14){move(leftToRight,0)};
object2d(K){show()};
object2d(L){show()};

object2d(M){show()};
component(Annotation14){hide()};
component(Annotation15){move(leftToRight,0)};
object2d(n){tween()};
component(Annotation15){hide()};
```

```
component(Annotation16){show()};
group[object2d(h){hide()},object2d(i){hide()},
object2d(j){hide()},object2d(G){hide()}];
object2d(n){hide()};
component(Annotation17){move(leftToRight,0)};
object2d(Poly2){tween()};
object2d(N){show()};
object2d(O){show()};
object2d(P){show()};
object2d(k){tween()};
object2d(l){tween()};
object2d(m){tween()};
object2d(Q){show()};
group[object2d(Poly2){hide()},
object2d(N){hide()},
object2d(O){hide()},
object2d(P){hide()},
object2d(k){hide()},
object2d(l){hide()},
object2d(m){hide()}];
group[component(Annotation16){hide()},
component(Annotation17){hide()}];
component(Annotation18){move(leftToRight,0)};

object2d(n){tween()};
object2d(Q){hide()};
component(Annotation18){hide};
component(Annotation19){fade(fadein,0)};
```

A2. Gram-Schmidt orthogonal process script

```
group[object3d(O){show()},
component(LabelAnnotation130){fade(fadein,0)}];
group[object3d(A){show()},
component(LabelAnnotation124){move(TopToBottom,50)}];
group[object3d(B){show()},
component(LabelAnnotation126){move(leftToRight,400)}];
group[object3d(C){show()},
component(LabelAnnotation128){move(leftToRight,400)}];
component(Graphsheet3d68){rotatey(-20)};
component(Graphsheet3d68){rotatey(20)};
object3d(Va){tween()};
object3d(Vb){tween()};
object3d(Vc){tween()};
```

```
group[component(Graphsheet3d68){rotatex(-45)},
component(LabelAnnotation130){hide()},
component(LabelAnnotation124){hide()},
component(LabelAnnotation126){hide()},
component(LabelAnnotation128){hide()}];
group[object3d(h){tween()},
object3d(VP){tween()},

component(LabelAnnotation115){move(TopToBottom,0)},
component(Annotation151){fade(fadein,0)}];
group[object3d(Vv){tween()},
component(LabelAnnotation117){move(TopToBottom,100)},
component(Annotation151){hide()}];
group[object3d(Plane_2){tween()},
component(Graphsheet3d68){rotatex(-10)}];
group[object3d(i){tween()},

object3d(Vs){tween()}];

group[object3d(Vw){tween()},
component(LabelAnnotation119){move(leftToRight,0)},
component(LabelAnnotation115){hide()}
];
group[object3d(Vp){tween()},
component(LabelAnnotation119){hide()},
component(LabelAnnotation143){show()},

component(LabelAnnotation117){hide()}];
group[component(Graphsheet3d68){zoom(10)},
object3d(i){hide()}];

group[object3d(Vq){tween()},
annotation(LabelAnnotation146){show()}];
group[object3d(Vr){tween()},
annotation(LabelAnnotation147){show()}];
object3d(Vw){hide()};

object3d(Vv){hide()};
component(ExpressionGroup148){fade(fadein,0)};
object3d(i){show()};
group[component(LabelAnnotation143){hide()},
annotation(LabelAnnotation146){hide()},
annotation(LabelAnnotation147){hide()}];
component(Graphsheet3d68){zoom(-10)};
component(Graphsheet3d68){rotatex(65)}];
```