Maestoso: Triumphant and Heroic Sketch Recognition for Music Education

Laura Barreto
Sketch Recognition Lab
Vassar College
Poughkeepsie, NY 12604
labarreto@vassar.edu

Paul Taele
Sketch Recognition Lab
Texas A&M University
College Station, TX 77843
ptaele@cse.tamu.edu

Stephanie Valentine
Sketch Recognition Lab
Texas A&M University
College Station, TX 77843
steph.lynn.valentine@gmail.com

Tracy Hammond
Sketch Recognition Lab
Texas A&M University
College Station, TX 77843
hammond@cse.tamu.edu

***Abstract*—As an adult, it becomes difficult to find efficient music education applications. Alternatives include signing up for music classes at a local college or signing up for a Coursera course. While these choices prove to be successful in teaching people the basics in reading music, these options provide setbacks such as cost to take the course, grading based off of inconsistent peer reviews, and tedious methods of checking answers. Most music education opportunities include a structured class with peer or professor feedback. There are no music education applications that allow users to learn how to read and write music independently with automatic feedback. In order to circumvent this dilemma, we present Maestoso: a music education application that utilizes shape recognition algorithms. Maestoso is a pen based system that gives users feedback and guides users through a series of interactive lessons, challenges, and tutorials. This application allows many curious novice musicians to learn how to read and write without the need of a professor or tutor to check over his or her work in order to receive feedback. Maestoso provides additional methods of developing musical reading and writing skills. There has not been any sketch-recognition based contribution to music education therefore it is essential to augment the list of domains that have sketch recognition based educational interfaces. Ultimately, Maestoso is an application that catalyzes a new dimension of music education and simultaneously amplifies the domains of education sketch-recognition interfaces.**

I. Introduction

As an adult, it becomes difficult to find efficient music education applications. Learning how to read and write music without the structure and aid of a teacher-taught class can prove to be a daunting task. Alternatives include signing up for music classes at a local college or signing up for a Coursera course. Online classes have professors and peers that provide feedback to the user's work online. A Massive Open Online Course such as Coursera offers numerous introductory level musicianship courses that teach students the basics of music theory. It guides users from learning the staff, all the way to learning Neopolitan chords. Through MOOCs, the creator of the course creates quizzes and assignments that the students complete. While these choices prove to be successful in teaching people the basics in reading music, these options provide setbacks such as cost to take the course, grading based off of inconsistent peer reviews, and tedious methods of checking answers. For example, some music questions require picture responses which consists of writing their response by hand, taking a picture of the response, and then uploading their response. In order to provide feedback, peer reviewers and the professor has to download the uploaded image one by one. This method is monotonous, tedious, time consuming, as well as inefficient if the picture contains sloppy answers, bad lighting, blurry nature, or barely visible answers. Not only does the peer and professor go through every single submission, but the user has to wait for the peer or professor to finish grading every other submission in order to receive feedback. Despite these limitations, there are no adequate solutions. The only other music education opportunities are geared for children. Music Applications available to users are applications that are made for seasoned musicians who have had exposure to reading and writing music. These applications act as electronic music paper. The only way for users to be able to use the already available applications is to have a solid understanding of reading and writing music. There is a lack of music education applications for adults. If it has been possible to create educational interfaces for Japanese Kanji, mathematical problem solving, drawing faces, and for mechanical physics, then there should already be an educational interfaces devoted to teaching the art of reading music. In order to combat the lack of sketch-based music educational interfaces, we present Maestoso: a music education application that takes advantage of sketch recognition algorithms such as the $1 and Hausdorff algorithms. Maestoso is a pen based system that guides users through interactive lessons, challenges, and tutorials. Maestoso allows users to learn how to read and write music at their own pace as well as liberates users from waiting for feedback from a professor or peer. This grants autonomy to professionals and pre professional users in the process of building their musicianship skills. Maestoso requires no other equipment besides a computer. The drawing aspect of Maestoso can be improved with the use of a Wacom Bamboo tablet. We hypothesize that sketch recognition can be applied to music education in order to provide users with favorable and successful outcomes. Ultimately, the purpose of Maestoso is to catalyze a new dimension of music education and to augment the domains of educational sketch-recognition based interfaces. Maestoso's versatility will allow the user the option of either learning the foundations of music independently or alongside a structure class environment.

II. Related Work

A. Music Applications

Sketch based music interfaces include NotateMe, Music Notepad, MusicHand, and NoteFlight. These applications are

fantastic sources for musicians due to their present knowledge of how to write music. These applications would not be user friendly for some one who has no knowledge of how to read music. There are online resources such as Coursera, or websites such as musictheory.net or datadragon.com that will provide people young and old with the option to learn how to read music.

1) NotateMe: NotateMe [1] is described as a portable music composition application that can be purchased through the Apple App Store, or Google Play. It provides a similar experience to writing music with pen and paper except NotateMe also is able to provide instant playback, editing, and a printable version of the users composition at the end. The playback option will play the users composition back to the user including the sound for the instruments the user is writing for. On top of providing sketch notation options and playback options, it also provides users with a recognizable drag and drop option to do actions such as indicate pedal markings (when composing for piano), ornaments, and appoggiatura [2]. As a user composes by hand, the application generates a clean, computerized PDF version of the users composition. NotateMe includes the option to take pictures of scores and then playback the score.

2) Music Notepad: Developed by students and faculty at Brown University, Music Notepad [3] acts as an electronic sheet of music paper. It has the capability to create notational symbols, notes, rests, beams, accidentals, key signatures, pick instruments, and score playback. In addition, Music Notepad also has editing features that includes a stroke recognizer to delete elements from the composition. Users are required to draw notes, rests, beams, and accidentals. Yet for a key signature, a user would have to draw a point and then label the piece with the user's desired key. The user does not physically draw out the sharps or flats in his or her desired key signature. The final version of a users composition will resemble a paper and pencil composition.

3) Music Hand: Music Hand [4] is a music application developed by Brown University which utilizes an optimal character recognition based approach. Taubman describes Music Hand as a tool for the quick and easy inputting of music given that users already know how to write music. Ultimately, MusicHand is able to recognize clefs, accidentals, noteheads, note stems, note flags, and beaming between notes.

4) Note Flight: NoteFlight Crescendo allows teachers to create assignments and students to submit their answers through their NoteFlight accounts for commenting, grading, or editing by other students [1]. All the interactions between the user and the application are based on user clicks and selections. Note Flight enables users to save their work online and be able to access their composition at a later time and on a different device. NoteFlight allows users to learn hands-on by creating compositions and completing assignments, however, it does not utilize sketch-recognition to do so [5].

5) Coursera: Coursera [6] is a Massic Open Online Course otherwise known as MOOC. Coursera is by far one of the most useful tools to all people who want to expand their knowledge. It offers thousands of courses ranging from cooking, to computer science. The courses available vary in topic and in difficulty and level. Berklee College of Music offers multiple online courses through Coursera for beginner musicians trying to improve upon their musicianship skills or to learn beginner music notation. The downside to taking an introductory music course through Coursera is that quizzes and assignments typically include questions that are open ended, multiple choice, or picture based. A user would either have to write a paragraph about their understanding, pick between 2-3 multiple choice options, or take a picture of their written response and submit it for peer-review. For example, in the first assignment for the *Developing Your Musicianship* course, the following is what is required of the students:

> Write out the C major scale by hand.
> 1) Download and print the staff paper, or write a staff by hand.
> 2) Draw a treble clef at the beginning of the staff. Please view the drawing a treble clef video demonstration.
> 3) Write the notes of the C major scale using whole notes, half notes, or quarter notes.
> 4) Scan or take a picture of your staff with the scale and then upload it for evaluation. . . .

It takes four steps for the user to be able to complete one question out of the five questions for the assignment. It is a tedious process to answer one question. Though Coursera has been proven to produce strong student learning outcomes, the experience for the user would be vastly improved alongside a sketch recognition interface.

6) Music Education Websites: Along the process of searching for sample music lessons for Maestoso, musictheory.net/lessons [7] and datadragon.com [8] were discovered. MusicTheory offers multiple lessons for people beginning to learn how to read music. It takes users step by step in a PowerPoint format. Each page of the lesson has some information and an image to go along with the text. DataDragon is similar in that it provides block quotes that feature explanations of the multiple music terms necessary to know before reading music. These music websites, while useful, are limited in versatility. They are limited to text and images and do not feature any interaction between the student and lessons. This method of learning is inefficient as the student will not remember the material after only five minutes of reading an entire page. These websites are better used as reminders rather than as educators.

7) Applications for Kids: Music4Kids [9] is an application that is geared towards music education for children. The interface of the applications features colorful backgrounds, animals as noteheads, composing options using drag and drop, and game mode.

> Music4Kids by Olivier Romanetti allows a user to place musical notes on a staff to write songs, but these are not just any notes. These notes have faces of a bird, hamster, octopus, fox, and a ghost.

Music4Kids is a form of exposing children to music as opposed to actually teaching them how to read and write music. Throughout the application, kids are not taught how to draw notes, or how to draw clefs, or how to draw sharps or flats. There is another app by the same creator called Music4Babies that provides a similar experience for babies.

B. Sketch Recognition

Maestoso utilizes a combination of low-level $1 and Hausdorff algorithms in order to recognize the many aspects of musical notation. These are not the only algorithms available for sketch recognition. There has been more work done in low-level recognizers in order to distinguish between scribble intentions. In addition to the low-level recognizers, there are high-level recognizers such as PaleoSketch and LADDER.

1) $1 Recognition: $1 [10] is a low-level gesture recognizer that can recognize single stroke objects. The algorithm of $1 can be described in four steps. The first step is to resample the point path. The point path is made depending on the speed of the user's stroke. If the speed of the user's stroke is fast, the points will be more spaced out than if the user's stroke was made slowly. In order to make gestures recognized despite the speed of the stroke, it is necessary to resample the point path and make sure all points are spaced evenly. The next step is to rotate. This step is not needed for our algorithm due to the assumption that users will not draw music notes upside-down. After rotation, the gesture is scaled and translated to a reference square. The last step is the most important step as it acts out the recognition. The last step compares the user's gesture to a template gesture. The limitations of $1 includes that if a stored template is a circle or a square, the recognizer will not be able to tell the difference between a circle and an oval or a square and a rectangle without having to modify the code. In addition, $1 does not take time into account when recognizing thus time cannot be a part of the algorithm.

2) Hausdorff Recognition: The Hausdorff recognizer [11] uses Hausdorff distance in order to compare the template gesture and the user's gesture. When a user enters a multi-stroke gesture, the recognizer overlays the gesture on a template and compares the distance between points. If the distance between points are nearly zero, then the shapes are similar and the user drawn gesture should be recognized as the template. Hausdorff plays an important part in the making of Maestoso because there are a few music notational symbols that requires more than one stroke in order to be complete. Hausdorff allows users to create an object through multiple strokes.

3) Scribbled Intentions: Research has been done on distinguishing between sketched scribble look alikes [12]. This research has shown that the most efficient way to determine whether a shape is being filled in or scribbled out and deleted is to take into account density of the object as well as the intersections between the filling in and the bounding object. This is important to Maestoso due to the fact that quarter notes are similar to half notes except for the fact that the notehead is filled in.

4) PaleoSketch: Unlike $1, PaleoSketch [13], a low level recognizer, is able to tell the difference between circles and ovals, as well as arcs from curves. PaleoSketch is able to recognize lines, polylines, circles, ellipses, arcs, curves, spirals, and helixes. PaleoSketch is a useful tool when trying to recognize the difference between similar shapes such as two lines connected at a corner and a curve.

5) LADDER: LADDER [14] is a high-level recognition system that can be customized for many domains. This allows writers to recognize sketches without having to know how to code a recognition system. LADDER describes shapes and shape groups in order to create a versatile recognition system that can be used for multiple domains such as physics or mathematics.

C. Sketch Education Interfaces

The idea of using sketch-recognition algorithms in order to create an educational system is not an unfamiliar idea. There have been applications such as Hashigo, MathPad, Mechanix, and iCanDraw that use sketch recognition in order to create sketch-based applications that teach Japanese Kanji, mathematical problem solving, mechanical physics, and how to draw faces respectively.

1) Hashigo: Hashigo [15] was developed as a tool for Japanese Kanji learning. Though it is simple to look at the shape of a character in Japanese and copy the character, it is difficult to learn the correct written technique of the written character. In a classroom setting, it is difficult for teachers to monitor more than 10 students at a time and make sure that every single student has the correct written technique. Instructors sometimes require students to label each stroke even though it is an unnatural form of writing. To solve this problem, Hashigo not only recognizes the correct character, but also recognizes the stroke order in order to be able to determine if a student wrote the character technically correct on top of visually correct.

2) MathPad^2: MathPad^2 [16] is a mathematical sketching tool used for mathematical problem solving. This educational application can be used to create math sketches that allows students to associate math expressions with free-body diagrams. MathPad^2 offers animation options for the sketches made by students. The animation feature allows students to visually see the connection between the mathematical equations of a system and the system itself.

3) Mechanix: Mechanix [17] allows mechanical and civil engineer professors to create assignments through the application with a drawn answer. After the professors create their set of questions through Mechanix, students are able to log on to their account and answer the question. Sketch recognition compares the students sketch to the professors sketched answer. This saves the professors hours of grading. Instead of sitting through 100 assignments at a time to provide feedback, Mechanix allows professors to provide feedback as soon as the student checks their answer. Mechanix checks to make sure that a students answer includes a correctly configured truss, an axis, correct values and directions for all forces, and the same force as given by the professor.

4) iCanDraw: iCanDraw [18] is an application that provides user feedback to users learning how to draw. Users will freehand draw and iCanDraw provides users with tips on how to improve their drawing, suggestions on what to draw next, and step instructions. This application guides users through the basics of drawing and helps the user's drawing skills improve. After user studies, it was shown that the users' freehand drawings improved after using iCanDraw.

III. Implementation

Before creating the interface, we had a design stage where we created paper prototypes of our idea of Maestoso. Through

this process, we discovered the inconvenience of placing a ”Clear” button next to a ”Playback” button. The user could accidentally press ”Clear” and erase the answer instead of pressing ”Playback,” which will play audio of the user response. This allowed us to have an idea of what we wanted Maestoso to look like before coding. Figures 1 and 2 show the similarities between our paper prototype design and the final design.

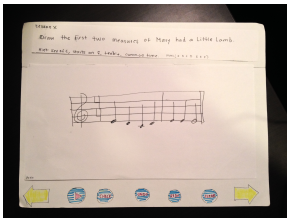

Figure 1. Paper Prototype

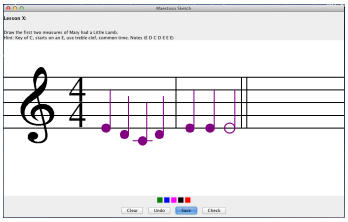

Figure 2. Final Prototype

Maestoso consists of four panels: a text panel, a draw panel, a results panel, and an interactions panel. The text panel is used to display the lessons, challenges, and questions. The draw panel is used for the user to draw their answer and interact with the program. The results panel displays ”CORRECT” or ”TRY AGAIN” after the user clicks the ”check” button. The interactions panel includes all the necessary buttons such as the color of the pen, a clear button, an undo button, save button to save the file and translate the drawing into an XML file, and a check button to check the answer as soon as the user is done drawing.

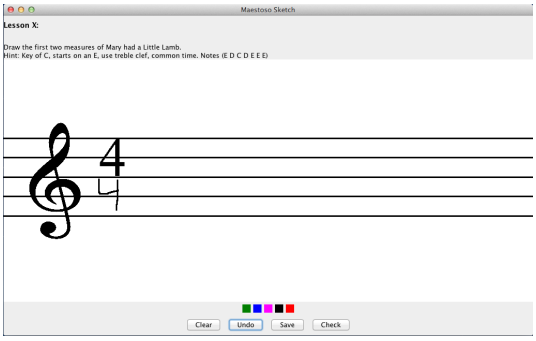

Figure 3. Top number is recognized, Bottom number is about to be recognized

In order to be able to recognize the user’s drawing, we used Hausdorff and $1 algorithms. These algorithms allowed us to match templates. For items that only require one stroke, we used $1. For items that can be drawn in multiple strokes, we used Hausdorff. We have action listeners to define what a stroke, shape, and sketch is. The stroke is the set of points from the moment the user presses with the pen to the moment the pen is released. The shape is the list of strokes until it is recognized by the program. The sketch is the final product. We first worked on recognizing a staff because a staff is the backbone of the music composition. Without the 5 lines and 4 spaces, its hard to tell what pitch a note is going to have. After the staff, we worked on the clefs because the clefs are an indicator of what range the composition takes place in. A quarter note by itself does not tell the complete story. However, a quarter note on a staff with a treble clef does. After clefs are recognized on Maestoso, the sketched clef is changed into an image of a clef for clarity. When the numbers of the time signature are recognized, they are turned into block numbers. Figure 3 shows this process.

In order to be able to later check the user response, we first had to have a method of comparing the recognized music shapes with an answer. In order to do so, it was imperative to create a method that saves the recognized user response as an XML file. The save() method called by the save action listener does exactly that. As a shape is recognized, the shape can then be classified into a note shape, key signature shape, time signature shape, or other shapes. After defining the recognized shape as one of these objects, they are then assigned to their corresponding elements of an Answer object which is converted to XML. Meanwhile, an Answer XML file is stored for each question.

Once we were able to save the user response, a Parser was created to read the XML files. The parser creates Answer objects from XML files as it reads the file. Because the Answer object is a new object created by us, we had to create a method that checks non-numeric equality between these objects. We called this new method isEqual() and took in the object, and two strings: similarities and differences. This method is able to detect differences found between the objects. It is able to print to the console the differences found. This feature will be useful when we work on not only returning whether or not the response is correct or incorrect, but what part of the response the user has to change. It also allowed us to test whether or not our equality method was working by printing the differences and similarities found between objects. After successfully creating the isEqual method for the Answer object and all the sub-objects in the Answer object, we had to find a way to compare two Answer objects. This method called compareDocuments() returns a String. If the Answer objects are equal, compareDocuments() returns a ”Correct”, if not, it would return ”try again”.

```
- <answer>
    <clef type="Treble"/>
    <keySignature/>
    <timeSignature top="4" bottom="4"/>
  - <measure>
      <atom type="QuarterNote" duration="1.0" position="10" accidental="NONE"/>
      <atom type="QuarterNote" duration="1.0" position="11" accidental="NONE"/>
      <atom type="QuarterNote" duration="1.0" position="12" accidental="NONE"/>
      <atom type="QuarterNote" duration="1.0" position="11" accidental="NONE"/>
    </measure>
  - <measure>
      <atom type="QuarterNote" duration="1.0" position="10" accidental="NONE"/>
      <atom type="QuarterNote" duration="1.0" position="10" accidental="NONE"/>
      <atom type="HalfNote" duration="2.0" position="10" accidental="NONE"/>
    </measure>
  </answer>
```

Figure 4. Example of XML Answer code

An example of a sample answer XML code is shown in Figure 4. After the user ”checks” the response, the XML code generated by the writer will parse both the user response XML and the answer XML and compare the Answer objects generated by the Parser. Any mismatch will deem the answer incorrect. A print statement will inform users which aspect of their response was incorrect.

Finally, our check button action listener called the method check(), so all we had left was to figure out how to show the user whether or not the answer was correct. In the check method, we compareDocuments() and if it returns the string

”Correct,” then check accesses the Results panel and changes the result label to ”CORRECT” in green. If the string does not return ”CORRECT,” then check would change the result label to ”TRY AGAIN” in red.

IV. Results

Maestoso is capable of recognizing the following musical shapes:

> *Staff, Treble Clef, Bass Clef, Key Signatures, Time Signatures, Sharp Accidentals, Flat Accidentals, Natural Accidentals, Beamed Eighth notes, Eighth notes with flags, Dotted Eighth Notes, Quarter Notes, Dotted Quarter Notes, Half Notes, Dotted Half Notes, Whole Notes, Dotted Whole notes, Eighth rests, Dotted Eighth rests, Quarter rests, Dotted Quarter rests, Half rests, Dotted Half rests, Whole rests, Dotted Whole rests, Single Bar lines, Double Bar lines.*

These make up the basic shapes that appear in introductory music lessons. Once the user clicks the ”save” button, Maestoso creates an XML file from the recognized shapes. After the file is outputted, the user has the option to ”check” their answer. The Parser generates an Answer object and successfully compares the parsed user response XML with the parsed answer XML. Once the files are compared, the user is notified by the accuracy of their response. Maestoso is able to display whether or not the answer is ”CORRECT” or if the user should ”TRY AGAIN.” These are promising results that can be used revolutionize music education interfaces. Figures 5 and 6 show the results panel.

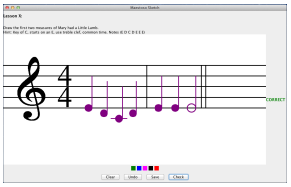

Figure 5. Correct User Response

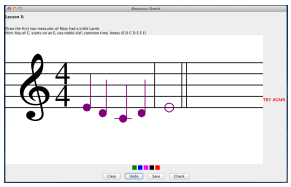

Figure 6. Incorrect User Response

V. Discussion

After ten weeks, we were able to create a functional prototype of Maestoso. The results we received were promising and if Maestoso is further developed, could prove to be revolutionizing for music education interfaces. Despite the available mediums to learn how to read and write music, there has been a complete void of sketch-based music education interfaces that allow users to get automatic feedback and interact directly with the interface. By using algorithms such as $1 and Hausdorff, we were able to recognize the necessary music shapes that are introduced in introductory musicianship courses. The novel idea this research represents is being able to check musical sketches and provide feedback. Music Education has been a domain that has not taken advantage of the advances made in sketch-recognition. Since it has been possible to create education interfaces for Japanese Kanji, mathematical problem solving, and drawing faces, then it is about time that the methodology behind creating the educational interfaces be applied to music education. Maestoso is a means of improving and aiding music education.

VI. Future Work

A. Interface

It is in our plans to create a cleaner looking interface with a customized appearance. We want to include a redo button along with the clear and undo. It is not necessary to have a redo button, but it is a huge convenience to users. In addition, we want to include a playback button that would allow users to play back and be able to hear answers and compositions. The capability of our isEqual() method allows us to pinpoint what exactly caused the user response to be deemed incorrect. It is important to not only notify users whether or not the response is correct, but also what aspect of their response needs another look. We want to be able to let the user know what part to rework in order to get the correct response. In addition, it would be crucial to notify users as the user is drawing whether or not the direction of the stems are correct or if there are too many or too few notes for a measure. An imperative change to be made is to make the circular shape of the note-heads into right slanted ellipse shaped note-heads due to the traditional method of music notation. The right slanted ellipse shaped note-heads help musicians naturally guide their eyes from left to right while reading music.

B. User Studies

We plan on doing a case study in which we observe the effectiveness of Maestoso by having two sample groups: one group learns how to read and write music through an online course, and the other group utilizes Maestoso. This will help us discover if Maestoso serves its purpose and is able to efficiently teach users how to read and write music. The user study will also aid us in discover what aspects of Maestoso could be improved. After the user study is completed, Maestoso will once again undergo reconstruction in order to implement changes suggested by users.

C. Availability

At the moment, Maestoso is only available for Java enabled web devices. It is in our plans to make Maestoso available for iOS and Android devices. This will allow Maestoso to become an application for iPads and Android tablets and allow music education to be an on-the-go experience.

D. Plugins

Due to the popularity of the growing MOOC, Coursera, we would like to have the option to create a plugin for the online music education courses. This would allow students to write their responses to answers directly through the Coursera website in order to get automatic feedback on their responses. Adding a plugin to Coursera would not only improve the student experience, but will also provide an easier alternative to printing out paper, writing the responses by hand, taking a picture or scanning response, and then uploading their response for peer review.

VII. Conclusion

Maestoso was created in order to facilitate independent study resources for adults. It provides an alternative to signing up for music classes at a local college or signing up for

a Coursera course. Even though these methods prove to be successful in teaching people the basics in reading music, Maestoso will help users all over be able to learn how to read and write music at the user's own pace and desire. This application is an alternative to traditional music education options such as college courses or MOOC courses. Hopefully in the future, Maestoso can not only prove that the famous quote used among musicians, "practice makes perfect," does not only apply to learning instruments, but also interacting with a program is beneficial to a student's success. Maestoso is the first step towards revolutionizing music education and was able to add to the list of domains with sketch-recognition based educational interfaces.

REFERENCES

[1] C. Criswell, "Software in the spotlight," *Teaching Music*, vol. 21, no. 5, pp. 32–39, 2014.

[2] "Notateme," http://www.neuratron.com/notateme.html.

[3] A. Forsberg, M. Dieterich, and R. Zeleznik, "The music notepad," in *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '98. New York, NY, USA: ACM, 1998, pp. 203–210. [Online]. Available: http://doi.acm.org/10.1145/288392.288608

[4] G. Taubman, "Musichand: A handwritten music recognition system."

[5] "Note flight," http://www.noteflight.com/info/k12.

[6] "Coursera," https://www.coursera.org/.

[7] "Music theory lessons," http://musictheory.net/lessons/.

[8] "Learning to read music," http://datadragon.com/education/reading/.

[9] "Music4kids," http://www.smartappsforkids.com/2013/12/featured-app-music4kids-learn-create-and-compose-music-through-play.html, 2013.

[10] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes," in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '07. New York, NY, USA: ACM, 2007, pp. 159–168. [Online]. Available: http://doi.acm.org/10.1145/1294211.1294238

[11] S. Valentine, M. Field, A. Smith, and T. Hammond, "A shape comparison technique for use in sketch-based tutoring systems," in *Proceedings of the 2011 Intelligent User Interfaces Workshop on Sketch Recognition (Palo Alto, CA, USA, 2011), IUI*, vol. 11, no. 5, 2011.

[12] K. Dahmen and T. Hammond, "Distinguishing between sketched scribble look alikes," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, ser. AAAI'08. AAAI Press, 2008, pp. 1790–1791. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620270.1620355

[13] B. Paulson and T. Hammond, "Paleosketch: Accurate primitive sketch recognition and beautification," in *Proceedings of the 13th International Conference on Intelligent User Interfaces*, ser. IUI '08. New York, NY, USA: ACM, 2008, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/1378773.1378775

[14] T. Hammond and R. Davis, "Ladder, a sketching language for user interface developers," in *ACM SIGGRAPH 2007 Courses*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1281500.1281546

[15] P. Taele and T. Hammond, "Hashigo: A next-generation sketch interactive system for japanese kanji," in *IAAI*, 2009.

[16] J. J. LaViola, Jr. and R. C. Zeleznik, "Mathpad2: A system for the creation and exploration of mathematical sketches," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 432–440, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015741

[17] S. Valentine, F. Vides, G. Lucchese, D. Turner, H.-h. Kim, W. Li, J. Linsey, and T. Hammond, "Mechanix: A sketch-based tutoring system for statics courses." 2012.

[18] D. Dixon, M. Prasad, and T. Hammond, "icandraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 897–906. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753459