SCORING AN ANIMATED NOTATION OPERA – THE DECIBEL SCORE PLAYER AND THE ROLE OF THE DIGITAL COPYIST IN 'SPEECHLESS'

Cat Hope
Sir Zelman Cowen School of Music
Monash University,
Melbourne, Australia.
cat.hope@monash.edu

Aaron Wyatt
Sir Zelman Cowen School of Music
Monash University,
Melbourne, Australia.
aaron.wyatt@monash.edu

Dan Thorpe
Electronic Music Unit
University of Adelaide,
Adelaide, Australia.
thorpe.daniel@gmail.com

ABSTRACT

This paper outlines the developments made to the Decibel ScorePlayer, the software that enables the delivery of the digital, graphic animated score of an hour long new opera by composer Cat Hope, entitled Speechless. The Decibel ScorePlayer is an iPad application that delivers the coordinated reading of graphic notation that was first created in 2012 and has been updated regularly ever since it was made available on the iTunes store the following year. Engaging the software to deliver the score for an hour long opera featuring a thirty piece orchestra, a thirty voice choir and four soloists saw considerable improvements made, contributing to a smooth running workshop period for the work. The workshop saw the score for the opera being updated, revised, added to and shared with different sections of musicians, vocalists, technicians and stage managers daily. This paper summarises the major additions to the score player software that came about as a result of this workshopping period and discusses procedures, developments and contributions engaged to facilitate the score updating process in a digital score for a large-scale work.

1. THE SCORE FOR SPEECHLESS

Composer Cat Hope has been using digital graphic scores to notate her music since 2008. While drafted with coloured pens and paper, the final graphic scores are created in the design program Illustrator, and exported as PDF for hardcopy or PNG files for the softcopy 'screen score' [1]. The development of Hope's composition practice has occurred alongside the development of the Decibel ScorePlayer application[1], with creative ideas for new works feeding the development of the player, and vice versa.

The Score Player application connects multiple devices playing the same score and keeps them accurately coordinated. It enables the viewing of independent parts and has functions such as pause, change of speed and the ability to scroll through the score to find particular points in the score.

The majority of Hope's scores use a 'scrolling screen score' format [1] where the graphic passes by a playhead, a vertical line that signals the point at which performers perform the score. While there are other models for the score player software, this is the one used for Speechless. To facilitate the workshopping period of the work, the programmer who wrote and continues to update the Decibel ScorePlayer (Wyatt) was engaged as the music director for the project, and a 'digital copyist' (Thorpe) worked with the director and composer on revisions and updates. The first of the two weeks saw a series of smaller workshops with soloists, the choir, and section leaders of the orchestra. The second week saw larger scale rehearsal with full orchestra and all performers.

The opera involves two community choirs (around 30 performers) from the local area, the Australian Bass Orchestra (a scratch orchestra of around 30 performers, all playing instruments that can perform notes below middle C [1]) led by four 'section leaders', and four vocal soloists. The work is made of three acts, an overture, and a short interlude section. Each instrumental group has its own colour in the part, as do the vocal soloists. The choir parts have similar colours to the vocal soloists, but in different opacities. The instruments sometimes have divisi parts, and some instructions presented on the score as text in english. The notation is proportional - pitch denoted by placement on the vertical axis, dynamics, and density by the thickness of the line.

The workshop period concluded with two performance showings of the work at the conclusion of around ten working days. The work was conducted by Wyatt, and featured some minimal staging, lighting, and sound reinforcement.

[1]https://itunes.apple.com/au/app/decibel-scoreplayer/id622591851

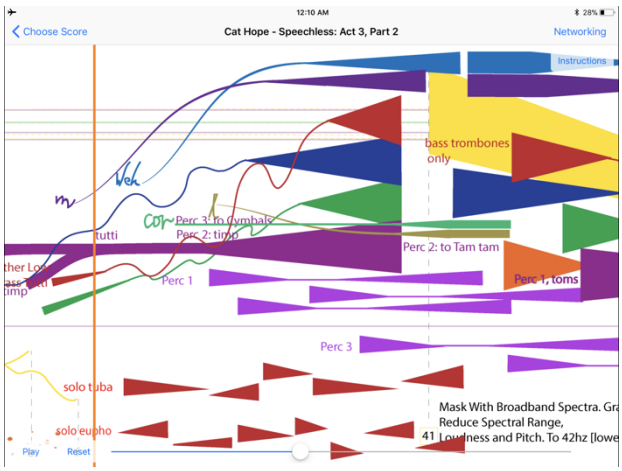

Figure 1. An excerpt from a section of the Speechless master score showing the red (brass), purple (percussion – in different shades of purple according to instrument), yellow (electronics), Orange (harp, piano and bass guitars), green (wind), blue (cello/double bass) and four vocal parts (violet, navy, brown and lime) and text instructions.

2. FACILITATING MULTIPLE UPDATES

To date, the largest number of iPads engaged simultaneously while using the Decibel ScorePlayer app had been 18, serving around 30 performers, in a performance of Hope's The Moment of Disappearance, at the performance venue Carriageworks in Sydney in 2014. The software worked smoothly in performance, although there were networking issues that arose in rehearsal that have since been resolved, mostly due to the heavily automated manner in which the ScorePlayer was making use of the Bonjour protocol at the time [2]. Loading the score onto that many iPads was time consuming and cumbersome, largely due to the proprietary, walled off nature of the iPad software. Each individual device had to be connected in turn via USB to a computer, with the score file transferred using the file sharing feature of iTunes. This was very time consuming for a one-off performance, but for a workshop environment where new versions of the score need to be uploaded to around 30 devices on at least a daily basis, this has the potential to become a major obstacle.

To help mitigate this problem in preparation for the Speechless workshops, a new option to check for score updates from a web server was added to the ScorePlayer software, with further refinements made over the course of rehearsals. It was also decided to bundle together the individual sections (acts, interlude, and overture) that made up the entirety of the opera, each appearing as a separate score in the player, within the one overarching score file to limit the number of downloads required, and to ensure that there was no chance that version discrepancies could occur between the different acts. While the first version of this file had to be transferred via iTunes as usual, subsequent versions of the score could be downloaded over the network from within the ScorePlayer itself. To achieve this capability, the opus.xml [3] file within the score file first needed two additional elements: one that defines the current version and one that points to a header containing update information.

```
<version>0.54</version>
<updateurl>
http://finn.psiborg.org/scores/Speechless
</updateurl>
```

The header itself, hosted on the update server, is a simple text file containing two comma separated values. The web address of the most recent revision of the score file and its version number:

```
http://finn.psiborg.org/scores/Speechless.dsz,0.55
```

When checking for updates, the ScorePlayer compares the current version of the score with the version given in the update header and if the remote score has a higher version number it gives the user the option to download the new score file. The code that it uses to extract and install the zipped score file from the server is the same code that is used to install a file installed locally via iTunes, so no additional checks are performed to verify that the server is supplying what it claims to be. While this level of trust in the server is acceptable when using a tightly controlled local network, as was the case in these workshops, some level of security checks will need to be added in the future if the feature becomes more widely adopted, especially if score updates are hosted online.

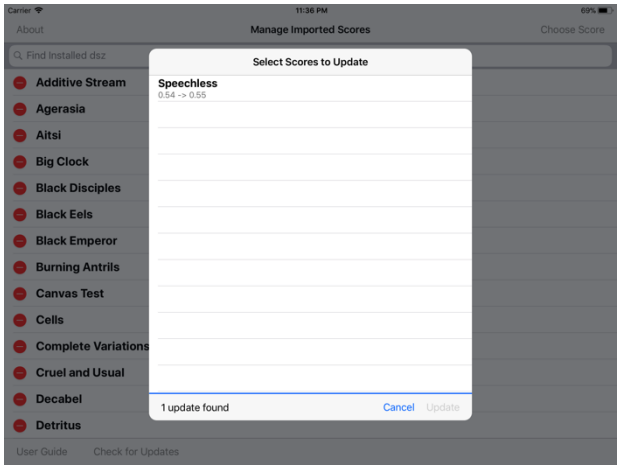

Figure 2. The update window in the ScorePlayer.

Further refinements to the update code included fixing some minor cosmetic issues (cleaning up the way some messages were displayed in the status bar) as well as fixing some more serious bugs. The code uses Apple's NSURLSession API [4] to download both the header files and the updated score files, and by default any HTTP requests are cached. This resulted in some of the iPads failing to see that updates were available if they had a recently cached version of an outdated header file. While this could be worked around by using Cache-Control directives [5] on the web server end, the more permanent fix was to change the caching policy on the iPad for our header download NSURLSession, as demonstrated in the code below:

```
//Create our initial URL session.

NSURLSessionConfiguration *headerSessionConfig =
    [NSURLSessionConfiguration defaultSessionConfiguration];

headerSessionConfig.timeoutIntervalForRequest = 2;

//Make sure we always download our headers and don't use any
    cached responses.

headerSessionConfig.requestCachePolicy =
    NSURLRequestReloadIgnoringCacheData;

headerSession = [NSURLSession
    sessionWithConfiguration:headerSessionConfig];
```

We also had issues at times with the response received by the iPad containing unexpected trailing characters which affected the version number. While it was unclear whether the issue was within the Apple API itself, our adoption of the API, or our web server, we were able to correct the problem by coding the ScorePlayer to ignore any header data beyond the first newline character, and to trim the received data to the length expected from the HTTP headers.

3. FURTHER SCOREPLAYER IMPROVEMENTS

As well as the code additions that allow for the updating of scores within the ScorePlayer, a number of other changes were put into place. These were principally to ensure the performance ran as smoothly as possible with our increased number of iPads on the network. As such, most of these changes were to the networking code, with the aim of increasing both stability and ease of use. These included simple changes such as a new alphabetical sorting of the list of connected iPads in the network connection screen, that made it much easier to see whether all of the devices were connected as expected and to quickly identify any that weren't. Naming and labelling the iPads by instrument also greatly helped with this process, as users could call out the name of the iPad, or the conductor could identify the iPad by instrument section, rather than having to go into the settings to see the assigned name of their own iPad.

Before the Speechless project, changing between scores required having to disconnect from and reconnect to the network. It was soon apparent that this would be a problem in a single long form work made up of different score sections that are sometimes required to flow easily in to one another. The change meant that the conductor was able to move the ensemble seamlessly from one section of the opera to the next without the need for any of the performers to interact with the iPads. The only way to achieve anything like this previously would have been to create one massive, single scrolling score that contained the entire opera. While doable, it would have made it difficult to quickly and easily change the duration and tempo of individual sections – an important facility in a workshop session. While files could be stretched or compressed in Illustrator (where the scores were created), this would have made the source image more difficult and unwieldy to work with, and consume considerable time.

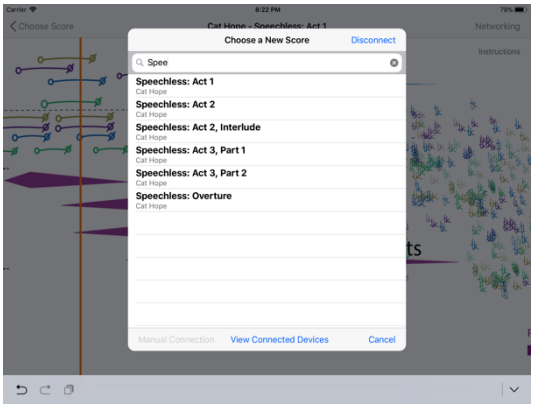

Figure 3. The score change window in the player.

To achieve this capability, a few new network commands needed to be added to the ScorePlayer, breaking compatibility with the previous version of protocol. The new network protocol (Decibel Networking Protocol v14) has been used in all versions of the ScorePlayer now since version 1.8.0, first released to the Apple App store on the 28th of June, 2017. Any previous versions of the ScorePlayer will no longer connect to these newer versions and must be updated. As with many of the existing ScorePlayer commands [6], these new commands can also be sent via Open Sound Control (OSC) from external devices, such as a laptop running MaxMSP, allowing further control and automation options if desired. These commands are outlined in Appendix A.

4. THE SCORE SERVER

The device used to host the update server was a first generation Raspberry Pi Model B [7], named 'finn', running the Raspbian Jessie distribution. As well as running Apache2 to provide our HTTP server, finn ran DHCP and DNS servers to provide network configuration and name resolution services for the iPads. The advantage of this approach is that it provided a single, cheap, and highly portable device that, once set up, can be easily deployed in conjunction with a wireless access point in any venue. While we used the ISC DHCP server and Bind for DNS during the workshop, subsequent smaller projects and performances have used a more lightweight Dnsmasq utility[2].

As well as providing a staging area for updated scores, finn provided us with the opportunity to automate aspects of the ScorePlayer score creation process. Each score has a master and several parts that are extracted from it (in this case strings, percussion, wind, brass, electronics, choir, soloists, bass guitars/piano/harp). In the design of the original Illustrator masters, the parts are ascribed to separate layers, that are exported separately to create parts in the ScorePlayer 'score file'. To make a score file for the Decibel ScorePlayer, the score png image/s must be combined with other data and turned into a specific file type with the filename extension 'dsz'. By connecting to the network and navigating to http://finn.psi-borg.org/scores/, the composer or copyist was able to

[2] http://www.thekelleys.org.uk/dnsmasq/doc.html

upload new versions of the png score images to the server. By setting large enough values in the php.ini [8] configuration file for **upload_max_filesize**, **max_file_uploads**, and **post_max_size**, the images for every part and every section of the score could be uploaded in a single, convenient transfer.

The PHP script that handled the upload only accepted files that were named as expected so that we didn't end up with a lot of unrelated materials in the upload directory, and so that the uploaded files worked properly with the score creation script. Since the ScorePlayer checked this by comparing the names of the incoming files to the names of the files already in the server's images directory, empty files initially needed to be created with the Linux *touch* command[3] before the first set of images could be uploaded. This check also meant that the person uploading the images could be quickly alerted to any error that might have occurred in the image naming process.

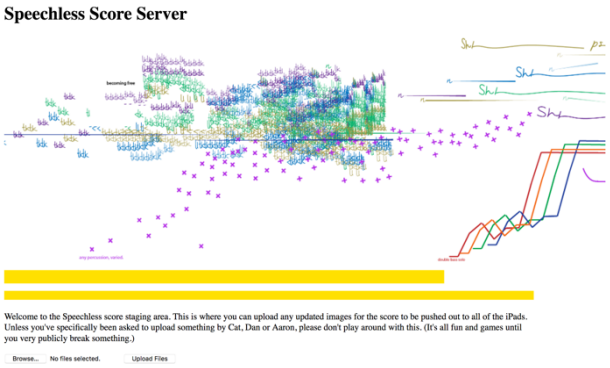

Figure 4. The score update web page hosted on finn.

Once the updated score image files had been uploaded, the score creation Python script, MakeDsz.py, could be run manually to create the new score file and header file. The only user interaction required with the script was the entry of the new version number, although in future even this could be automated if desired using a version number based on a timestamp. As well as editing the version number in the opus.xml file and writing the new header file, the script also split images that were too big for display in the ScorePlayer into equally sized tiles using the Wand[4] ImageMagick binding for Python. It then zipped the opus.xml file and all of the image resources together to create our finished score file.

Earlier versions of the script attempted to call the ImageMagick convert[5] command rather than using the Wand library. While this worked in tests on an Apple laptop, performance on the Raspberry Pi was very poor, with memory shortages proving a major issue. Using Wand fixed these issues, and allowed the script to run faster and more reliably. Manipulation of such large images is still relatively slow on the original generation Pi, however recent testing on a Raspberry Pi 3 Model B shows that the newer devices fare much better. Measured using the Linux time command, the score creation script took 2'35" to run on the newer device compared to 30'14" on the original – a marked improvement.

5. SCRIPTING ILLUSTRATOR

When it was discovered that Illustrator could be scripted using Javascript [9], we saw an opportunity to automate the exporting of png files from the original score image Illustrator files. As well as eliminating the potential for human error in the process (that occasionally resulted in files of the wrong dimensions), it also assisted in the creation of score images for individual parts. Performers found it useful to see the whole score, but often had issues seeing their own part within the gamut of instruments. In addition, their part was sometimes obscured by other parts. To assist the performers, parts were made for each section, where their part was shown at full opacity, while the rest of the orchestra was at 30% opacity. While this would have been possible to do manually, it would have been a time consuming and tedious process.

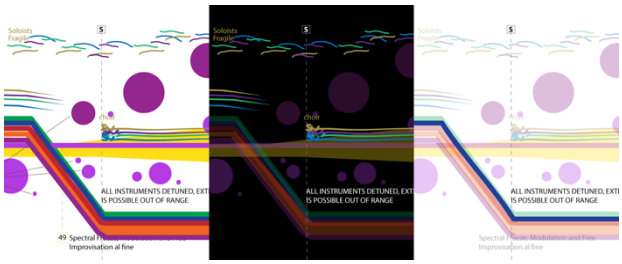

Figure 5. A side by side comparison of an excerpt from Speechless showing the full score to the left, the projected vocal part in the middle, and the blue cello/double bass part highlighted, with the rest of the parts at 30% opacity on the right.

It also provided the opportunity to fix another issue that arose from the performer workshops. The singers (both choir and soloists) were reading their parts from a projection behind the audience, and the score's white background was producing a lot of undesirable light bleed from the back of the hall into the main performance space. As well as lighting the space undesirably, it drew the audience attention to the score, rather than obscuring it, which was the original intention of projecting it there. The team also wanted to experiment with the idea that it may be easier to distinguish the singers individual parts from a distance when placed against a darker background, as the choir is broken into four parts at times, and there are four soloist parts. To add the black background manually in Illustrator would have required the selective colour inversion of all of the common elements of the score, such as the text used to signal rehearsal marks and general performance instructions, which were all in black, one element at a time. Automating the task was definitely the preferred option for accomplishing this.

The main body of the script loops through and sets the opacity for each of the layers as required, before exporting a png at the correct resolution for each part. In addition to the layers for each part, there is a common guide laye showing rehearsal marks to all players. These and

[3] http://man7.org/linux/man-pages/man1/touch.1.html
[4] http://docs.wand-py.org/en/0.4.4/
[5] https://www.imagemagick.org/script/convert.php

the layer for parts are set to 100% opacity, while the other layers are set to 30%. Conditional branches within the loop can be used to accommodate special cases. For example, the percussion and electronics parts contain three different colours in a single layer that need to be simultaneously set to 100% opacity. A subroutine is run for the vocal parts, that creates temporary duplicates of the guide and vocal layers with any colour that is almost black inverted, along with a black background layer fitted to the size of the Illustrator artboard. These are created and destroyed every time the script is run so that they precisely duplicate any changes that have occurred in the original working copies of the layers. For the purposes of the script, almost black was any object with an RGB value of less than 15,15,15 or a grayscale value of less than 30%.

Because of the nature of the object model employed by Illustrator's scripting engine, where layers and groups of objects can contain sublayers and subgroups, the inversion function had to act recursively. Each time it is called it iterates through the collections of path items, text frames, and compound path items that belong to the supplied group or layer, before being called again on any subgroups. While other types of objects can exist in a layer or group, the score for Speechless did not make use of them and so we did not include them in our script. The script should suffice, however, for any score that only makes use of vector graphics and text. And if colour inversion is not required, and only the layer opacity changes are needed, then it would work with little modification for a score that included these additional types of objects.

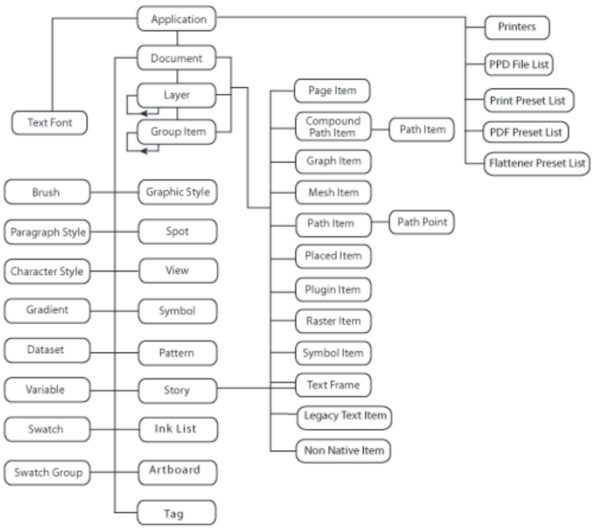

Figure 6. The object model used for scripting Illustrator[6] to generate the parts in the score. Layers and groups can contain sublayers and subgroups, as well as any number of objects in the adjacent column.

Since the script finds each layer based on the name assigned to it in Illustrator, alternative names can easily be substituted into the script. Additionally, a composer could simply choose to stick to a common naming convention for the Illustrator layers of their scores to make the script reusable. An array holds the list of layers that are iterated through in the main loop, while another variable holds an index to a special "GUIDES" layer that appears in every part. Since the script checks whether layers, including the guide layer, exist before performing any operations on them, it would be possible to fill the array with any possible layer name that you might want to use, and the script will export only parts corresponding to those layers that actually exist. It would also be possible to create special branches within the loop for specific layer names. For example, any layer whose names starts with "INVERT" could have its colours inverted in the way previously described.

The code excerpt below shows the start of the loop and the snippet of code that precedes it, where the guideIndex variable is set and where our array of layer names is set up. By checking if the result of getLayerIndexByName is equal to -1 we can find out whether the layer exists or not and can then either ignore it or operate on it as appropriate.

```
var guideIndex = this.getLayerIndexByName("GUIDES");

//Reshow our layers one at a time and export a png.
var parts = ["BLUE", "GREEN", "OLIVE", "ORANGE", "RED",
"YELLOW", "STAGING"];
for (var i = 0; i < parts.length; i++) {
   var layerIndex =
      this.getLayerIndexByName(parts[i]);
   if (layerIndex != -1) {

      ...
```

6. THE ROLE OF DIGITAL COPYIST

Outside of new software developments facilitating the ScorePlayer, a work of Speechless' scale and type also required a fresh approach to the scoring and rehearsal workflow. This was expedited by a unique relationship between the composer and copyist in the rehearsal cycle. Prima facie, the role of copyist in the digital, graphic, and animated score context [10] is not dissimilar to that in traditional notation: it requires an informed approach to refining and organising the composer's ideas, a knowledge of orchestration, and the ability to identify the needs of sections/performers to create adequate parts. In the case of Speechless, however, the copyist also needed to understand the composers' unique notational techniques, their approach to using the software (in this case, Illustrator) and the conventions of the ScorePlayer software. These aspects when combined with traditional approaches were informed and ultimately reshaped by the decidedly non-traditional delivery medium of the ScorePlayer itself.

Hope's notational practice and the ScorePlayer application allowed for several practical innovations that would otherwise be arduous in a more traditional context. In the context of Hope's notational practice, the clear advantage is the ability for players to read from a score while still having a clear and defined part through the opacity and colour coding discussed above. The ScorePlayer's ability to deliver essentially unlimited variations for groups of performers and was incredibly useful, and the copyist worked very closely with the programmer to facilitate the different requirements of individual sections of instruments. The use of vertical dashed lines to coordinate

[6] https://www.adobe.com/devnet/illustrator/scripting.html

player entries, and rehearsal marks as points of synchronisation were very useful in this automated environment, with the assistance of the conductor.

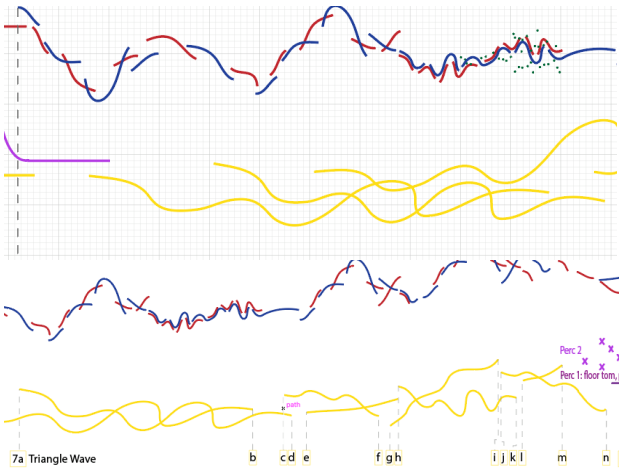

Figure 7. A screenshot of an early-stage, un-annotated section of Speechless, followed by a later draft of a similar gesture with additional cues and instrumentation marked in. Some cues – such as those on the yellow (electronics) part – came from the performers needs. Others, as deemed necessary by the copyist and composer.

The key challenges of the copyist role were the organisation required for layer-by-layer export, and the establishment of orchestration conventions in a decidedly unconventional context. A clear example of this was present in the foreground vs. background placement of musical materials. Hope's work uses proportional notation on both a horizontal and vertical axis, but also has a depth through the use of layers. In the context of a colour coded score, for example, this could mean an orange gesture appearing over a red gesture in one section, and then appearing below in another section. There needed to be a way of notating the depth, or complexity of multiple parts in the master, while enabling clear parts for the performers. Another example could be where there are five parts all playing at the same place, same dynamic on the vertical and horizontal axes - how can part be created that describes all the parts, but is accurate for each performer? Because instruments were sorted into layers on the master score, all of that instruments' gestures for that movement had a fixed, global depth in the part relative to other instruments. The copyist spent three days at the beginning of the workshop checking, re-organising and imposing a somewhat artificial hierarchy on months of work by the composer to simplify and ensure the parts were consistent and some level of depth was enabled on the master score.

Orchestration conventions were similarly challenging. A convention of indicating text instructions on a per-part and score-wide basis was required in a way that would not be interpreted as further graphic notation, and within the layer export hierarchy of the Illustrator script. A convention of using black, dashed lines and clear, bold typesetting within white boxes with black borders was established for score-wide text instructions. On a per-part basis (for example: divisi and tutti), colour coded text was used and black lines continued to indicate start times to contrast with the colour coded graphic notation. Score-wide instructions were the top of the layer hierarchy, and part instructions were in the same order as their respective part but always above the part to which they applied. In the case of the electronics parts, cue numbers were indicated in yellow-bordered boxes (similar to rehearsal marks), with dashed lines indicating start times, as seen in Figure 5.

An issue that was discovered in the performance workshops was the description of parts in the player. When the performer opens the score in the ScorePlayer on the iPad, they see the full score. Using an upward swipe motion, they move between parts. It didn't take long to realise that if the instrument didn't have any music in the first few minutes of the piece, they would be unable to know which was their part, unless they remembered how many swipes it took, which is a rather unreliable method. The copyist devised a method where a coloured triangle, that matches the colour of the part, shows in the lower corner of the score. This means that the master has a 'white' corner, concealing the other colours in the master, but enabling them to be seen in the parts. Figure 8 shows the red part, indicated by the red triangle in the lower left-hand corner. The opaque parts of voice 1 and percussionist 2 can be seen, yet the red part is yet to come into view. The coloured corner enables the instrumentalist to recognise their part. Also noted on Figure 6 is the removal of the horizontal swipe bar at the bottom of page, and the score information along the top and the orange playhead that can all be seen in Figure 10. These are other new functionalities of the ScorePlayer, developed for when scores are projected in a way where the audience does not need to see the timer, timeline, playhead or other iPad information. Also, a black background has been added at the start and end of each piece, to limit the screen luminosity at the beginning and end of movements.

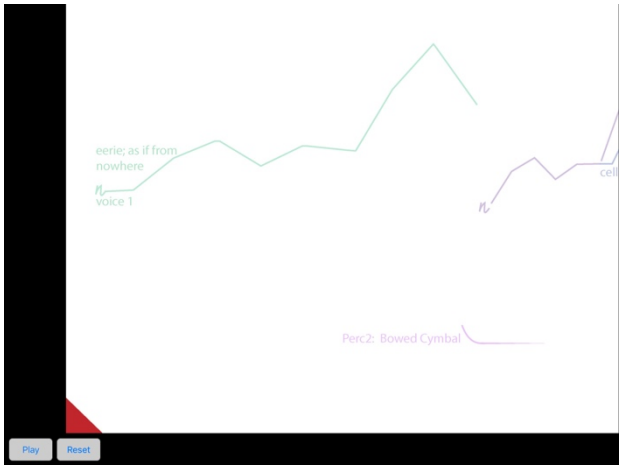

Figure 8. The start of Act 1 of Speechless, showing the red part, with the triangle in the corner indicating the correct part, as well as the removed playhead, timeline and other information from the top.

The workflow developed between the composer and copyist during the Speechless workshop period was unique for the workshopping of a significant, large scale, graphically notated, animated score. Once an agreed upon hierarchy had been established in the Illustrator versions of the score image files, an effective and efficient workflow between composer and copyist - and occasionally programmer - began to develop. Eventually, a very fast turnaround on corrections and changes was established. The composer would sit close to the conductor, following the score on her own, networked iPad and giving verbal notes to an assistant who would write on a Google Doc open on both the assistant and the copyist's computers. Using screenshots, notes and time codes, changes would be noted on the Google Doc, and seen by the copyist in real time. This meant that all changes to the score were happening as the piece unfolded, meaning that with the new Illustrator scripts automating part extraction, new parts could be ready and sent out as a score update to the iPads for the next rehearsal after a tea break. These kind of rapid changes would be impossible with typical paper or graphic scores.

SPEECHLESS SCORE UPDATES

DATE	NAME/PART	Score VS 1 Page	Score VS 2 Page	CHANGE
20-Jul	Cat Hope	overture	overture	Overturne additions - theremin. CH to do. Dan to check layers etc
		19	19	Add "as if from no-where" as marked
		20	20	Make wider space in green part, as marked
		21	21	Add blue vocal part as marked
		21	21	Blue wider, as marked
		21-22	21-22	All joins as marked
		22	22	New brown part as marked
		23	23	Joins as marked
		23	23	Wider gap in green part, as marked
		25	25	Return cyan to red
		27	27	Add text, "very low" as marked
		27	27	Add G as marked
		27-28	27-28	Add text to blue part
		27-28	27-28	Shorten vocals as marked

Figure 9. A screenshot of the Google Doc communicating changes from composer to copyist while rehearsal was taking place. A highlighting system was used to prioritise urgency.

Another important development from the project was the integration of stage management cues. A separate part was created for the stage manager to call lighting and sound cues during the showing performances. While these are currently quite simple, as seen in Figure 8, they will become quite complex, and even more useful in the full production of the work. They take advantage of the tight synchronisation of the Decibel ScorePlayer, and ensure that cues align exactly to the music at all times.

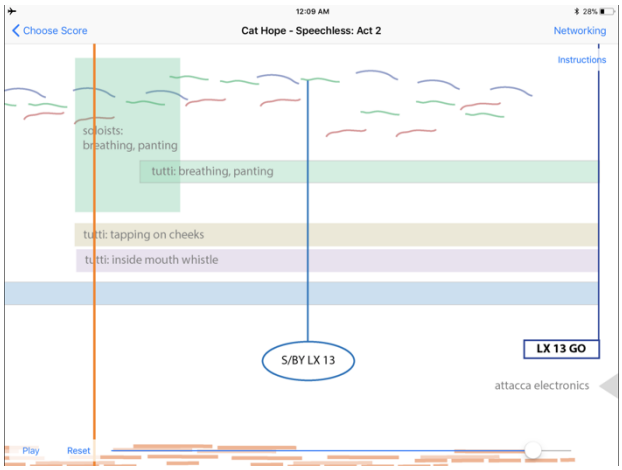

Figure 10. Screenshot of the iPad with the stage management part selected, showing lighting/sound standby and effect cues. Note how other parts are opaque.

The process of having a digital copyist for the Speechless workshop period has informed several potential future directions for the ScorePlayer in regards to the integration of notational media. The export algorithm could be revised to export parts by hex colour code, rather than layer. Assuming the composer retains the trend of using a set hex colour or range (for example 0x00FF69B4 to 0xFFFF69B4), this would allow for the layering of material in a more nuanced and compelling manner while retaining the convenience of batch export of parts; although the ability to quickly hide and show parts may be complicated by this. Overall, it is clear that this role within this project will continue to offer opportunities to develop practical digital graphic scoring methodologies, and methodologies for creating complex works with Decibel ScorePlayer as a delivery medium.

7. CONCLUSION

This paper has examined the workshop period of a new, animated notation opera Speechless and the new developments to the Decibel ScorePlayer that have come about as a result. These included new scripts to enable the extraction of parts from the Illustrator program into the ScorePlayer file format, new bulk update functions on the network, the use of a web server for score dissemination, new naming and labelling protocols for hardware and software, and new projection mode capacity.

The role of the 'digital copyist' demonstrated the ability for the copyist to bridge communication between the composer and programmer, and facilitate creative ideas into technical insights. This role contributed to the development of master score hierarchies, part identifiers and protocols. It also allowed for stage management instructions to be incorporated as a part in the score, and parts with theatre presentation taken into consideration were developed – a first for Decibel Score Player.

Speechless is the first opera written with an animated notation score, but at the time of writing, it is yet to be fully staged. The workshop period was a vital part of the works' development in the way it enabled further technical developments of the Decibel ScorePlayer in addition to the Speechless score, including its delivery to over 75 musicians and technicians in a series of rehearsals and showings.

8. REFERENCES

[1] Vickery, L "Screening the Score" in C. Hope (Ed.) *Audible Designs*, PICA Press, 2011.

[2] C. Hope, A. Wyatt, and L. Vickery, "The Decibel ScorePlayer: New developments and improved functionality," in *Proceedings of the International Computer Music Conference (ICMC'15)*, Denton, TX, USA, pp. 314–317, 2015.

[3] A. Wyatt and C. Hope, "Animated music notation on the iPad (or: Music stands just weren't designed to support laptops)," in *Proceedings of International Computer Music Conference (ICMC'13)*, Perth, Australia, pp. 201–207, 2013.

[4] NSURLSession - Foundation | Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/foundation/nsurlsession?language=objc

[5] Cache-Control - HTTP | MDN. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control.

[6] S. James, C. Hope, L. Vickery, A. Wyatt, B. Carey, X. Fu, and G. Hajdu, "Establishing connectivity between the existing networked music notation packages Quintet.net, Decibel ScorePlayer and MaxScore," in *Proceedings of the International Conference on Technologies for Music Notation and Representation*, A Coruña, Spain, 2017.

[7] eLinux.org, *RPi Hardware*. [Online]. Available: https://elinux.org/RPi_Hardware

[8] PHP, *List of php.ini directives*. [Online]. Available: http://php.net/manual/en/ini.list.php

[9] Adobe Systems Inc., *Adobe Illustrator CC 2017 Scripting Guide*. [Online]. Available: https://wwwimages2.adobe.com/content/dam/acom/en/devnet/illustrator/pdf/AI_ScriptGd_2017.pdf

[10] C. Hope, "New Scores for Electronic Music: The Possibilities of Animated Notation," *Computer Music Journal*, vol. 41, no. 3, pp. 21–35, 2017.

9. APPENDIX: NEW NETWORK COMMANDS

/Server/RegisterScores (*scoreName composerName scoreType version*)...

Sent to the server from the client iPads when they first connect. The client sends four arguments for each score installed on the device: the score's name, composer, type (for example "ScrollScore" for a scrolling score), and version number. (This is "0" if unspecified by the score file.) This is used by the server to keep a list of scores that are common to all of the connected devices. Only these scores are available for loading via a network command, so it is crucial that all devices have the same version of any score that you wish to load in this manner.

/Server/ScoreList (*scoreName composerName scoreType version*)...

Sent to the client iPads from the server whenever the list of common scores changes. (When one of the iPads connects or disconnects.) This ensures that the user interface avoids showing any scores that are not common to all of the devices, and so cannot be loaded via network command.

/Server/LoadRequest *scoreName composerName scoreType version*

Sent to the server from an iPad client or an external device to request a change of score. This is the command that is invoked by selecting a new score in the score change window of the ScorePlayer.

/Server/RequestRejected

/Server/RequestOK

Sent from the server to an external or iPad client in response to a LoadRequest message. The most likely reason for a negative response for an iPad is that a new device without the selected score just joined the network. (There is only a very small window in which this can happen before the list of common scores is updated network wide.) The iPad client simply shows a dialogue box announcing this. For an external, the most likely reason is that one of the parameters didn't exactly match the expected parameters for the score.

/Score/Load *scoreName composerName scoreType version*

Once a successful LoadRequest has been made, the server sends this command to all of the iPad clients. This prompts the new score to be loaded on all of the devices, and blocks any control messages (play, seek, reset) from being sent until all iPads have finished loading the new score, or until a five second timeout has been reached. It also prevents new clients from connecting during this period.

/Server/LoadOK

Sent by the iPad clients to the server to let it know that they have finished loading the new score.

/Server/GetScoreList

Can be used by externals to retrieve a list of available scores. This prompts the server to respond with a /Server/ScoreList message which is usually only broadcast to the iPad clients.