the number of times the composer would need to recharge batteries on each of the devices during the composition stage. Although HappyBrackets does not enable one to run more than one virtual device of the computer, it is possible to simulate multiple devices by layering sketches on top of one another, creating a pure logical multiplicity. Instead of sending a single sketch to thirteen different DIADs during the composition phase of the work, the required sketches were sent multiple times to simulator—three sketches that utilised IMUs and ten for static sonification—thus simulating the entire multiplicity [6]. Global scoped controls facilitated communication between all sketch instances, with no changes required to any of the code when changing to sketches running on the networked devices. Moreover, it was possible to add or remove more DIADs to the multiplicity without requiring any address mapping changes.

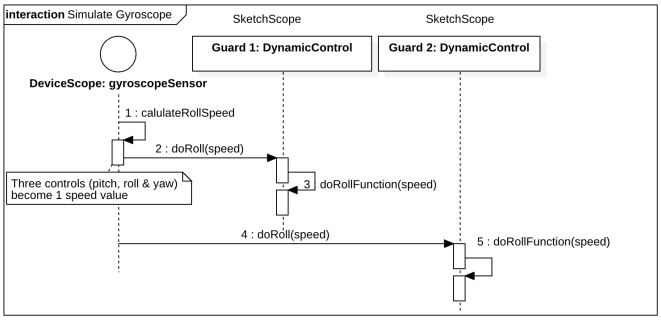

Fig. 14. Device and Sketch ControlScope

Fig. 15. Device and Sketch ControlScope

Simulating the three balls would require three sets of gyroscopes, however, the virtual device simulator only has one set of gyroscope controls. It is possible, however, to simulate these as separate controls for each device by using the gyroscope average values through a *SketchScope* control. Sensor simulator controls would have a *DeviceScope*, in that the values are transmitted to all sketches on the device. The *SketchScope* control only transmits its value to other DynamicControls in that same sketch. This effectively enables one to individually simulate an independent IMU for each sketch on the virtual device by connecting the *DeviceScope* control to the *SketchScope* control, shown in Figure. 14. Fig. 15 shows how the three interactive DIADs were simulated in the HappyBrackets GUI.

C. Complex Hand Movement Interface

The final work we present using our Dynamic Control API was for collecting complex hand and finger movements during embroidery. The primary focus of the research was to capture the complex hand gestures of traditional crafts, which often take a lifetime to acquire, to gain new perspectives on complex hand skills and ways of sharing the traditional wisdom embodied in these practices through development of a digital audio-visual interface [8]. The capture technique involved collecting synchronised multi-modal data consisting of three PixiCam cameras, a LeapMotion detector, and raw video capture. Each PixiCam captured continuous frames of the embroiderer's individual finger positions effected through the use of coloured nail polish on their hands, mapping various colour combinations to fingers. The three PixiCams, facilitating three axes, were all connected to the one Raspberry Pi through an I2C bus and grouped as a timestamped three dimensional frame. The LeapMotion detector uses a stereo infra-red camera to capture a three dimensional image of the person's hands, and using the LeapMotion software, able to determine pitch, roll and yaw, as well as finger extensions and positions. Similarly, the LeapMotion data was collated and timestamped as frames of data. At no time with the PixiCam or LeapMotion would it be possible to predetermine how much data would be in any frame, shown in Fig. 16.

Fig. 16. Embroidery Data Frames

The Raspberry Pi had to poll all three PixiCam devices and send the data to the capturing computer approximately every 20ms. Instead of using a global control to send the data, coupled global and target scope pair were used in a publish-subscribe pattern [37, p. 293]. The computer required to collate all the data subscribes by sending a global text message called "SendPixy" with it's device name, shown in the top section of Fig. 17. In the bottom section of the code, the device connected to PixiCam receives the subscription message, adds the device name as a target to the `pixyBlockSender` *ClassObject-Control* with control name of "pixycam.PixyBlock". The code simply has to call `setValue` with the PixyCam data, located in the variable `pixyMessage`. In the top section of the code, the control named "pixycam.PixyBlock" receives the code and

actions it. Using the Target Scope control prevents the Pi from receiving messages from itself, only directing them to where they are required.

```
// in Data collator
TextControl pixyRequester = new TextControl(this, "SendPixy", "").setControlScope(ControlScope.GLOBAL);
pixyRequester.setValue(Device.getDeviceName());

// Type classObjectControl to generate this code
new ClassObjectControl(this, "pixycam.PixyBlock", PixyBlocksMessage.class) {
    @Override
    public void valueChanged(Object object_val) {...}

}.setControlScope(ControlScope.TARGET);// End DynamicControl pixcamBlockReceiver code
```

```
// In Pi connected to PixiCam
PixyBlocksMessage pixyMessage = new PixyBlocksMessage(NUM_PIXY_CAMS);

ClassObjectControl pixyBlockSender = new ClassObjectControl(this, "pixycam.PixyBlock",
        PixyBlocksMessage.class).setControlScope(ControlScope.TARGET);

new TextControl(this, "SendPixy", "") {
    @Override
    public void valueChanged(String control_val) {
        pixyBlockSender.addControlTarget(control_val);
    }
}.setControlScope(ControlScope.GLOBAL);// End DynamicControl textControl code

// this will not send until a target has been added
// This is in a loop where pixyMessage is filled
    pixyBlockSender.setValue(pixyMessage);
```

Fig. 17. PixiCam messages sent via TargetScope control

The data is joined with LeapMotion frames as JSON data and stored to disk. Our intention is to synchronise the data with the raw video and then enter into a machine learning algorithm to learn various stitching patters. With the API, there is no requirement to encode or decode the complex data structure as it is completed in the lower layers, outside the scope of what the creative coder is required to do.

VI. Conclusion

We presented an alternate paradigm of sharing parameters throughout a multiplicity whereby variables are treated objectively rather than functionally. The existing models of defining parameters were based on a flat addressing scheme, and although were suitable to procedural or functional programming paradigms, we treated parameters as objects that can have scope, aliases, and could be manipulated based on their logical placement rather than their physical location. Instead of moving outside the OSC specification, we have altered the way we view and implement OSC messages in our software, while still maintaining OSC 1.0 compatibility when communicating between devices over the network. The ability to provide different scopes for parameters or functions could be potentially employed to other programs. For example, the send and receive pair in Max allows passing of messages between patches without requiring patch cables [59]. They currently facilitate sending messages globally based on a single name, although it can be restricted to a single device through a generated ID using Live [60]. Opening two patches in Max that have the same send and receive names could cause unintentional cross communication between the two. The facility to assign a level of scope and association would prevent this type of cross-talk, while allowing greater flexibility in mapping.

The ability to send and/or access complex data types as complete entities rather than lists of parameters promotes data abstraction and encapsulation, allowing greater flexibility through modular architecture as underlying data structures can change during the lifestyle or evolution of a computer based composition. Additionally, the facility to define data accessibility, and the ability to reuse human readable names based on a variable's scope is a common feature of most programming languages. This paradigm has been extended in that scoping a variable can be dynamically bound or addressed to specific objects, class types, devices or globally on an entire network.

Finally, we presented three works that utilised this new mechanism of parameter sharing throughout the multiplicity, showing how implementing control scope and unified data enabled composers to think about their parameters as logical entities, agnostic to the network, and treated as though they existed on the same device.

Acknowledgment

We acknowledge the work of Patricia Flanagan and Saurabh Srivastava for their work in embroidery capture project. We also acknowledge the support of the Australian Research Council through their Linkage Grant (LP180100151).

References

[1] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet, "Internet of musical things: Vision and challenges," *IEEE Access*, vol. 6, pp. 61994–62017, 2018.

[2] A. Fraietta, O. Bown, S. Ferguson, S. Gillespie, and L. Bray, "Rapid composition for networked devices: HappyBrackets," *Computer Music Journal*, vol. 43, no. 2-3, pp. 89–108, 2020.

[3] O. Bown and S. Ferguson, "Understanding media multiplicities," *Entertainment Computing*, vol. 25, pp. 62–70, 2018.

[4] W. Stallings, *Operating systems: internals and design principles*, Prentice Hall, Upper Saddle River, NJ, 2009.

[5] U. Vahalia, *UNIX internals: the new frontiers*, Prentice Hall, Upper Saddle River, N.J., 1996.

[6] A. Fraietta, "Transient relics: Temporal tangents to an ancient virtual pilgrimage," in *Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, New York, NY, USA, 2020, TEI '20, ACM.

[7] A. Fraietta and O. Bown, "Creating a sonified spacecraft game using HappyBrackets and Stellarium," in *Proceedings of the 17th Linux Audio Conference (LAC-19)*. CCRMA, Stanford University, USA, 2019, pp. 1–7.

[8] P. J. Flanagan and A. Fraietta, "Tracing the intangible: The curious gestures of crafts' cultural heritage," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, New York, NY, USA, 2019, UbiComp/ISWC '19 Adjunct, p. 49–52, Association for Computing Machinery.

[9] O. Bown and S. Ferguson, "Creative media+ the internet of things= media multiplicities," *Leonardo*, vol. 51, no. 1, pp. 53–54, 2018.

[10] A. Fraietta, "The smart controller workbench," in *Proceedings of the 2005 conference on New interfaces for musical expression*, Vancouver, BC, Canada, 2005, University of British Columbia, pp. 46–49.

[11] O. Bown, L. Loke, S. Ferguson, and D. Reinhardt, "Distributed interactive audio devices: Creative strategies and audience responses to novel musical interaction scenarios," in *International Symposium on Electronic Art*. ISEA, 2015, pp. 604–607.

[12] O. Bown and S. Ferguson, "A musical game of bowls using the diads," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2016, pp. 371–372.

[13] J. Axelson, *Serial port complete: COM Ports, USB virtual COM ports, and ports for embedded systems*, Lakeview Research, 2007.

[14] P. Doornbusch, "Early hardware and early ideas in computer music: Their development and their current forms," in *The Oxford Handbook of Computer Music*. Oxford University Press, New York, NY, USA, 2009.

[15] J. Pressing, *Synthesizer performance and real-time techniques*, AR Editions, Inc., 1992.

[16] M. Wright and A. Freed, "Opensound control: a new protocol for communicating with sound synthesizers," in *Proceedings: International Computer Music Conference 1997, Thessaloniki, Hellas, 25-30 september 1997*. The International Computer Music Association, 1997, pp. 101–104.

[17] A. Freed and A. Schmeder, "Features and future of open sound control version 1.1 for nime," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Pittsburgh, PA, United States, 2009, pp. 116–120.

[18] "Libmapper motivation," Web: http://libmapper.github.io/about.html.

[19] T. Place, T. Lossius, A. R. Jensenius, and N. Peters, "Addressing classes by differentiating values and properties in OSC," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Genoa, Italy, 2008, pp. 181–184.

[20] M. Wright, A. Freed, and A. Momeni, "Opensound control: State of the art 2003," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003, pp. 153–159.

[21] J. Malloch, S. Sinclair, and M. M. Wanderley, "Libmapper:(a library for connecting things)," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2013, pp. 3087–3090.

[22] I. Bergstrom and J. Llobera, "OSC-namespace and OSC-state: Schemata for describing the namespace and state of OSC-enabled systems," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, London, United Kingdom, June 2014, pp. 311–314, Goldsmiths, University of London.

[23] A. Hunt, M. M. Wanderley, and M. Paradis, "The importance of parameter mapping in electronic instrument design," *Journal of New Music Research*, vol. 32, no. 4, pp. 429–440, 2003.

[24] J. Malloch, S. Sinclair, and M. M. Wanderley, "A Network-Based Framework for Collaborative Development and Performance of Digital Musical Instruments," in *Computer Music Modeling and Retrieval. Sense of Sounds*, R. Kronland-Martinet, S. Ystad, and K. Jensen, Eds., Berlin, Heidelberg, 2008, pp. 401–425, Springer Berlin Heidelberg.

[25] F. Nazir and A. Seneviratne, "Towards mobility enabled protocol stack for future wireless networks," *Ubiquitous Computing and Communication Journal*, vol. 2, no. 4, pp. 65–79, 2007.

[26] A. Fraietta, "Open sound control: Constraints and limitations," in *Proceedings of International Conference on New Musical Interfaces for Music Expression (NIME-2008)*. Genova, Italy, Juns 2008, pp. 19–23.

[27] "Dot mapper: Use pointers for internal mappings," Web: https://groups.google.com/forum/?utm_source=footer#!msg/dot_mapper/0Qst4QCCJ5g/mzj3g6neAwAJ.

[28] J. Kleimola and P. J. McGlynn, "Improving the efficiency of open sound control with compressed address strings," in *Proceedings of the 8th Sound and Music Computing Conference (SMC)*, 2011.

[29] R. B. Dannenberg and Z. Chi, "O2: Rethinking open sound control," in *Proceedings of the International Computer Music Conference*, 2016, p. 494.

[30] J. Kleimola et al., *Nonlinear abstract sound synthesis algorithms*, Aalto University, 2013.

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, MIT press, 2009.

[32] I. Chivers and J. Sleightholme, *An Introduction to Algorithms and the Big O Notation*, pp. 359–364, Springer International Publishing, Cham, 2015.

[33] B. W. Kernighan and D. M. Ritchie, *The C programming language*, 2006.

[34] B. Smith, *Object-Oriented Programming*, pp. 1–25, Apress, Berkeley, CA, 2011.

[35] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML)," *World Wide Web Journal*, vol. 2, no. 4, pp. 27–66, 1997.

[36] T. Bray et al., "The javascript object notation (json) data interchange format," 2014.

[37] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Pearson Education, Reading, MA, 1994.

[38] D. M. Huber, *The MIDI manual: a practical guide to MIDI in the project studio*, Focal Press, Waltham, MA, USA, 2012.

[39] O. Bown, A. Fraietta, S. Ferguson, L. Loke, and L. Bray, "Facilitating creative exploratory search with multiple networked audio devices using happybrackets," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Porto Alegre, Brazil, June 2019, pp. 286–291, UFRGS.

[40] L. Loke, A. Fraietta, P. Crawley, A. Winston, and M. Leete, "Sonic sk8er," Performance, Aug 2019.

[41] M. Blume, M. Rainey, and J. Reppy, "Calling variadic functions from a strongly-typed language," in *Proceedings of the 2008 ACM SIGPLAN workshop on ML*. ACM, 2008, pp. 47–58.

[42] M. Fowler, "Patterns of enterprise application architecture," 2003.

[43] D. Caromel, L. Henrio, and M. Leyton, "Type safe algorithmic skeletons," in *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, 2008, pp. 45–53.

[44] M. I. Cole, *Algorithmic skeletons: structured management of parallel computation*, Pitman London, 1989.

[45] M. A. Ellis and B. Stroustrup, *The annotated C++ reference manual*, Addison-Wesley, 1990.

[46] B. Meyer, "Overloading vs. object technology," *Journal of Object Oriented Programming*, vol. 14, no. 4, pp. 3–7, 2001.

[47] M. Wright, R. J. Cassidy, and M. Zbyszynski, "Audio and gesture latency measurements on linux and osx.," in *ICMC*, 2004.

[48] N. Lago and F. Kon, "The quest for low latency.," in *ICMC*, 2004.

[49] R. H. Jack, T. Stockman, and A. McPherson, "Effect of latency on performer interaction and subjective quality assessment of a digital musical instrument," in *Proceedings of the audio mostly 2016*, pp. 116–123. 2016.

[50] R. Goonatilake and R. A. Bachnak, "Modeling latency in a network distribution," *Network and Communication Technologies*, vol. 1, no. 2, pp. 1, 2012.

[51] L. Cheng and C.-L. Wang, "Network performance isolation for latency-sensitive cloud applications," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1073–1084, 2013.

[52] L. Cheng, C.-L. Wang, and S. Di, "Defeating network jitter for virtual machines," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE, 2011, pp. 65–72.

[53] X. Guo, J. Dai, L. Li, Z. Lv, and P. R. Chandra, "Latency hiding in multi-threading and multi-processing of network applications," in *16th International Conference on Parallel Architecture and Compilation Techniques (PACT 2007)*. IEEE, 2007, pp. 270–279.

[54] D. J. Barrett, L. A. Clarke, P. L. Tarr, and A. E. Wise, "A framework for event-based software integration," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 5, no. 4, pp. 378–421, 1996.

[55] W. W. Milner, "A broken metaphor in Java," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 76–77, 2010.

[56] J. Rumbaugh, "Relations as semantic constructs in an object-oriented language," in *Conference proceedings on Object-oriented programming systems, languages and applications*, 1987, pp. 466–481.

[57] P. America, "Inheritance and subtyping in a parallel object-oriented language," in *European Conference on Object-Oriented Programming*. Springer, 1987, pp. 234–242.

[58] AMEI and MMA, "Common rules for MIDI CI property exchange Version 1.0," Web: https://www.midi.org/midi2, 2020.

[59] M. Puckette, "Combining event and signal processing in the max graphical programming environment," *Computer music journal*, vol. 15, no. 3, pp. 68–77, 1991.

[60] C. '74, "send: Send messages without patch cords," Web: https://docs.cycling74.com/max8/refpages/send.