InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media

Clarice Hilton
Nicola Plant
c.hilton@gold.ac.uk
n.plant@gold.ac.uk
Goldsmiths University of London
London, UK

Rebecca Fiebrink
Phoenix Perry
University of the Arts London
London, UK

Carlos Gonzalez Diaz
University of York
York, UK

Ruth Gibson
Coventry University
London, UK

Bruno Martelli
Studio Gibson/Martelli
London, UK

Michael Zbyszyński
Marco Gillies
m.gillies@gold.ac.uk
Goldsmiths University of London
London, UK

ABSTRACT

Interactive Machine Learning offers a method for designing movement interaction that supports creators in implementing even complex movement designs in their immersive applications by simply performing them with their bodies. We introduce a new tool, InteractML, and an accompanying ideation method, which makes movement interaction design faster, adaptable and accessible to creators of varying experience and backgrounds, such as artists, dancers and independent game developers. The tool is specifically tailored to non-experts as creators configure and train machine learning models via a node-based graph and VR interface, requiring minimal programming. We aim to democratise machine learning for movement interaction to be used in the development of a range of creative and immersive applications.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design**; **Human computer interaction (HCI)**.

KEYWORDS

machine learning, movement interaction, creative virtual reality, artists, dancers

ACM Reference Format:
Clarice Hilton, Nicola Plant, Rebecca Fiebrink, Phoenix Perry, Carlos Gonzalez Diaz, Ruth Gibson, Bruno Martelli, Michael Zbyszyński, and Marco Gillies. 2021. InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media. In *VRST '21: ACM Symposium on Virtual Reality Software and Technology, Dec 08–10, 2021, Osaka, Japan.* ACM, New York, NY, USA, 10 pages.

VRST '21, Dec 08–10, 2021, Osaka, Japan

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

1 INTRODUCTION

Movement sensor technologies, such as hand and body tracking, are becoming more widely used in immersive creative work as they have become more affordable, reliable and with an infrastructure that is accessible to creative practitioners such as artists and dancers [1, 22, 23, 30, 34]. There is a demand for tools and ideation approaches that enable the design and implementation of movement interaction for immersive environments.

Although movements that exhibit simple and straightforward mappings between movement and output in the virtual environment, such as grabbing and moving virtual objects, can be designed and implemented with relative ease, designing more complex movements are not well supported by current tools. For artists and dancers to create movement-based interaction artworks they must rely on a developer to program a way for their system to understand a particular movement[29]. Movements that are more performative or expressive cannot easily be defined mathematically and are hard or not possible to represent in programmed rules, requiring expert

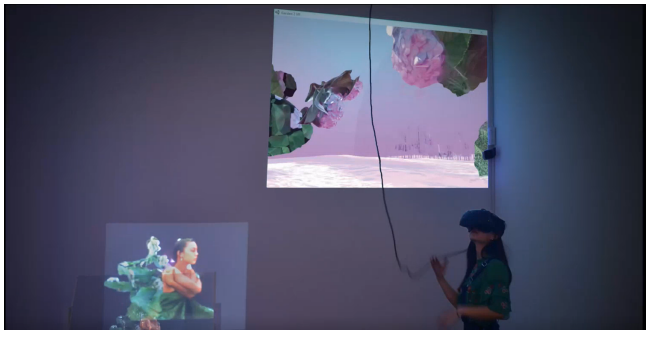

Figure 1: Artist working with immersive media

knowledge and often can only be achieved with machine learning. Interactive Machine Learning (IML) [14], enables creators to implement movement designs simply by performing them; it also provides quick feedback to enable a rapid iterative design workflow.

This work presents InteractML [1], a tool that allows creators to use IML to recognise and implement complex movement interaction designs [7, 16, 27–29]. The development of the tool aims to support creative practitioners and independent developers in designing movement interaction for immersive media, by building tools that allow design by moving and do not require artists or performers to have a background in computer programming. InteractML has been designed with a User Centered Design Process based on workshops and longer residencies with artists, dancers and game designers (as described in section 3.1). InteractML has been presented in demos and workshops at several conferences [7, 16, 27–29]. This paper gives a in more depth description of the design of the tool, describes the methodology of how the tool has been developed and what incites have been learnt through case studies of using the tool than has been presented in previous short papers.

2 BACKGROUND

2.1 Movement Interaction in VR

Movement interaction is increasingly sought after in VR development across many different fields and industries. Designing movement interactions for immersive applications that are intuitive, effective and easy to accomplish is a relevant endeavour as interaction styles move towards bodily-based ways of interacting and as most VR systems come with movement interface devices as standard.

Movement interaction has been shown to increase the sense of presence in VR applications, the sense of immersion depending on sensorimotor contingencies in a virtual world correlated to those experienced in the real world [33]. Movement interaction can also increase users' focus and attention [35]. [20] states that as our perceptuo-motor system is fundamental in an experience of cognitive reality, movement interaction in a virtual world is important to reproducing the cognitive and emotional engagement the real world. [19] describes 'reality-based interaction styles' as interactions in a virtual world mirroring the interaction we experience in the real world. Movement interaction is successful as they replicate the way we know to interact with the real world, such as interacting with our environment, manipulating objects, our bodies and other people.

2.2 Examples of Movement Interaction in Immersive Creative Work

In the creative domain, VR experiences that incorporate movement interaction can be found in exhibitions, theatres and museums. Although movement interaction is commonplace in popular VR games and applications such as Beat Saber and TiltBrush, creative work that utilises movement interfaces outside of head tracking remain novel and are produced primarily by teams of artists with a computing background or by professional creative studios.

Mutator VR: Vortex (2016) applies machine learning to generate abstract worlds in VR. Through movement interactivity users can mix between smooth and sharp rhythmic changes in the dynamics and characteristics of abstract shapes and audio in the virtual environment. The work was created by artist William Latham alongside mathematicians and programmers Stephen Todd, Lance Putnam and Peter Todd [21]. *Dream* (2021) is an online live performance based on Shakespeare's A Midsummer Night's Dream, created by the The Royal Shakespeare Company in collaboration with Manchester International Festival, creative studio Marshmallow Laser Feast and the Philharmonia Orchestra. The virtual play saw actors performing in motion capture suits where their movements and voices were animated by virtual avatars in the virtual world that was live-streamed to audiences. The production used Gestrument technology (Nordin, 2020) to allow performers to control pre-recorded musical phrases with their movements in real-time. *We Live in an Ocean of Air* (2019), another work created by Marshmallow Laser Feast is a multi-person, multi-sensory VR artwork that uses a tracking system to show the movement of audiences' hands and heads with red dots in the virtual space. The piece sees audiences explore a virtual forest evolving into a more abstract, interacting with moving particles representing the inside of a giant sequoia tree. Rachel Rossin's VR work *Stalking the Trace* (2019) enables the audiences to use their bodies to manipulate the shifting of time in scenes of explosions, using forward and backward motion to scrub forward and back in time and perspective.

3 METHODOLOGY

3.1 User-Centric Design Process

Development of the tool followed an iterative user-centric design process including our target users at every stage of the development. Over a two-year period we ran four in-person one-day workshops and three five-day online hackathons bringing together artists, dancers, developers and researchers working in immersive media, of a range of experience levels. These included short sessions that aimed to facilitate an introduction to InteractML and guidance on how to use it when developing their creative work. Over each workshop/hackathon, participants were tasked with producing a simple prototype of a creative application that used InteractML to implement movement interaction. We also developed a movement interaction ideation practise to compliment the IML workflow of performing movements to train models (see Section 3.2) that was explored at the beginning of each workshop/hackathon so participants could use it when designing movement for their prototypes.

We also invited artists to participate in three artist residencies that took place over 2 months so we could get a better sense of how creators worked with the tool and ideation method over a longer period. The artists developed a piece of work which involved movement interaction using the tool. The residency started with a 5 day hackathon building the beginning prototype. This was followed by 7 weeks of weekly discussions of how they were using the tool and any support they needed, as well as an active Discord where they could ask questions and share their progress. This allowed us to observe the learning process of the tool and how the use of the tool develops over time, giving in depth insight into how the tool might be used in creative practice and best for its design to

[1] https://interactml.com/ and https://github.com/Interactml/iml-unity

support this. We also mentored other artists from our workshops to develop their ideas from the workshop/hackathon into their creative practises.

All workshops sessions and post-workshop interviews with participants were video-recorded. Interviews were conducted individually or in groups as convenient to participants. For the residencies, interviews and support sessions with artists were conducted weekly, these were either individually or in groups on alternate weeks. A demonstration and discussion of the artists work created during the residency was presented at the end of the residency and post-residency interviews were conducted individually. All sessions and interviews were video recorded. All resulting video material was transcribed. In addition, gestural and movement data annotated as this was key to this research. The videos were coded directly and analysed using an open thematic analysis approach[3].

For our user-research we wanted participants from a dance and artistic practice who were engaged with creating virtual reality in Unity. For our workshops we aimed to recruit a range of skill level of programming and Unity experience (43 total, 24 experienced in Unity, 18 experienced in machine learning). These workshops were run online using a combination of Microsoft Teams, Rec Room and Discord. To recruit for the workshops we targeted: InteractMLs online community (Discord, Twitter, Facebook and mailing lists), IML and embodied interaction research communities from contacts made from presenting at conferences and several XR industry groups known to the researchers. We were able to connect with users across the globe with a range of programming and Unity experience. For the artist residencies we targeted users with less experience of programming and Unity as this is who the tool is targeted at (4 total, 1 experienced in unity and machine learning.

3.2 Movement Interaction Design Methodologies

In response to immersive media creators using bodily or movement-based interaction styles in their applications, there are several design approaches that embrace the affective and embodied dimensions in their processes. The soma-design approach suggests a framework that encourages a slower process that is actively reflective of a lived embodied experience. The approach also encourages movement interaction designers to rely on their first-person tacit understanding of moving and interacting [18, 32]. These methodologies advocate developing movement interaction designs through physical 'bodystorming' or 'embodied sketching' by designing by doing and moving [15, 17, 31] at an early stage of the design process. Designing by moving enables designers to reflect on the changing experience of movement over time, through this reflection the activities supports cycles of reflection and refinement. An important feature of soma-design is iterative testing and feedback of sociodigital material, where a creator experiences the correspondence between their embodied action and the system's response—this feedback is important while creators are still in the design process so the pairing between movement and system response can be properly explored. However, current practices involve a sharp change when moving from ideation to implementation. While at the ideation stage, designers can focus on moving, implementation involves sitting down at a computer and coding. This breaks the embodied thinking that was a key part of the ideation process. InteractML aims to use machine learning as a way to continue the embodied, movement based process into the implementation phase, by training learning algorithms with examples of movement.

To compliment the design by moving process that is a key feature of IML, we adopt the embodied sketching design approach as an ideation practise when introducing InteractML to users. Following from [6] the method we share to participants uses an adapted form of the critical incident technique; a procedure that elicits designers to recall recently lived memories from their everyday lives to apply as input for design [11]. Here, the method takes its form as a 'movement incident', where participants are instructed to remember, from the past few days, an atypical situation in which a memorable movement contributed. This guides the designer towards their own lived experience, in line with [25] approach of focusing, based on invoking an awareness of the felt qualities of embodied experience. We offer this method alongside the IML tool to bridge the design process from ideation to implementation, allowing for quick cycles between embodied exploration and system feedback from the immersive application [29].

3.3 Interactive Machine Learning

Interactive machine learning (IML) is a technique originally proposed by Fails and Olsen [8] that allows people to quickly create and refine supervised machine learning models. In the approach to IML used in existing movement and real-time interaction design systems such as Wekinator [10], a user first records a set of training examples, where each consists of an example input to the machine learning model (e.g., a person's movement) paired with the desired model output (e.g., a label denoting the category of movement, or a real-numbered value used to control some aspect of the computer's behaviour). The system then trains a supervised learning algorithm on these examples, producing a trained model which is a function capable of outputting new values in response to new inputs (e.g., producing new labels in response to new movements in real-time). Critically, IML systems enable users to iteratively evaluate this trained model (e.g., by experimenting with it in real-time on new inputs) and to modify it by changing the set of training examples. For instance, if a model does not accurately label certain variations of a given movement category, the user can add further examples of this variation to the training set in order to improve the model's accuracy. Or, a user could add examples of new categories of movements to the training set to make the model's behaviour more complex.

IML systems often employ algorithms such as shallow neural networks or nearest-neighbour classifiers which do not require large numbers of training examples or long training times [10, 12, 36]. This means that the process of recording examples and training a model can be very fast (e.g., training might take a few seconds or less on a dataset containing anywhere from a handful to a few hundred examples). Consequently, IML users can rapidly iterate, quickly adjusting models to fix mistakes or explore new design possibilities.

Wekinator [10] was the first tool to employ IML to support the creation of movement-based interfaces. It was originally developed to aid in the creation of movement-sound mappings in the design

of new digital musical instruments, though it has since been used in designing interactive art and games projects as well. However, it would take significant adaption to work with any VR creation software. Other toolkits have been designed to support experienced developers in employing IML in specific application domains (e.g., GRT [12] and ml.Lib [5] are designed for music programmers) and/or programming environments (e.g., RapidMix and RapidLib [36] support C++ and JavaScript development).

4 INTERACTML TOOL

InteractML is a node-based IML tool for designing movement interactions in the Unity 3D game engine [7, 16, 27–29]. Users can train, modify and run machine learning models in real time, using real-time gesture demonstrations to create new training examples and test trained models. InteractML can be used with any input sensor or numerical data. Users can use transform data from any virtual object in Unity and choose from the built-in movement features or take data directly from a script into an InteractML graph. This means that users can train the algorithm on the movement of the headset, controllers, finger positions or any sensor information brought into Unity. They can then use the model's output in any script in Unity, for example to trigger an animation, control the colour spectrum or any other behaviour desired. Whilst InteractML is hardware agnostic, it has been designed for use with immersive technologies, specifically VR. There is a VR module compatible with Unity's interaction toolkit including a VR interface for easy movement design in headset (see Fig 4).

InteractML is a unique tool for designing movement interaction by enabling

- **Embodied design:** Users develop their movement interfaces through performing the movements, harnessing the tacit knowledge of movement in the body.
- **Fast iterative process** for designing movements: Users can quickly test different movements with different outputs, with the ability to quickly tweak and change the movement and its effect simply by adjusting the training examples. It allows for the experience of interaction to be embedded in the process of design itself, creating a smoother testing and development process.
- **Low barrier to entry** to enable people without expert knowledge of machine learning to be able to design movement interactions themselves.
- **Versatility:** It is extremely customisable with regard to the nature of motions that can be modeled, the types of data that can be used, the ways that model outputs can be used in a design and how the code base can be built upon
- **VR Interface** allows for users to work independently in a VR headset to control models

4.1 Machine Learning

The machine learning back-end is the RapidLib C++ [36] library. RapidLib's implementation mirrors that of Wekinator [10], whose current choice of algorithms and default parameterisations have been informed by more than a decade of use by creative practitioners, ensuring that they work well off-the-shelf for many IML tasks involving small training sets and human motion modeling tasks. InteractML currently implements three types of machine learning algorithm: classification, regression and dynamic time warping. Classification models output one of a finite set of labels for each input. This is suitable, for instance, for human pose identification, where a certain pose will create a certain output (e.g., recognising a "thumbs-up" using the rotations of finger joints). Classification employs the k-nearest-neighbour algorithm, which can work well in IML contexts when a designer creates a small training dataset and uses a data representation that is relatively free from irrelevant data and noise (e.g., as is the case when using skeleton data for body or hand tracking) [9]. Dynamic time warping models also output one of a finite set of labels, but unlike classification, they take into account movement over a recent period of time. This algorithm is best suited for dynamic movements which trigger a certain behaviour, for recognising a movement from a prayer pose to hands stretched into the sky, then triggering a flower animation to grow. Regression models output a continuous, unbounded value in response to an input representing (like classification) data captured from a single moment in time. Regression can be used for continuously controlling something as a person moves between two (or more) poses; for instance, RGB values of an object could change continuously as a person moves their hand from knee to shoulder level. In InteractML, regression is implemented using a multi-layer perceptron neural network with one hidden layer; such an algorithm does not place undue restrictions on the shape of the model function to be learned (e.g., functions can be nonlinear) yet can often be trained effectively on as few as a dozen examples.

4.2 InteractML Process

InteractML enables users to interactively create and modify node graphs to match their desired use case. This includes choosing the algorithm, configuring the data capture (including choosing sensor inputs and configuring *movement features*—i.e., deciding which values to compute from these inputs), and specifying how the model outputs will be used (e.g., what effect they have). Once they have set up the graph, users can quickly record a training set by demonstrating a set of example movements and pairing these with the desired corresponding model outputs, train the model, and interactively test it by performing gestures and observing whether the model outputs change in the desired manner. Users can adapt the computation graph and training data to reach their desired result. This process is inspired by Fiebrink's wekinator[10] and is visualised in figure (See Fig. 2).

Below, we describe in more detail the process a user might employ to build a working system. For concreteness, we ground each step in an example application and process, in which a user wishes to control a particle system according to the speed of their waving hand.

4.2.1 *Configure InteractML Node Graph.*

- **Choose sensor inputs:** The user specifies the game object or value from the script which provides data related to the movement of interest. In our given example, the user might initially choose the left VR controller game object.

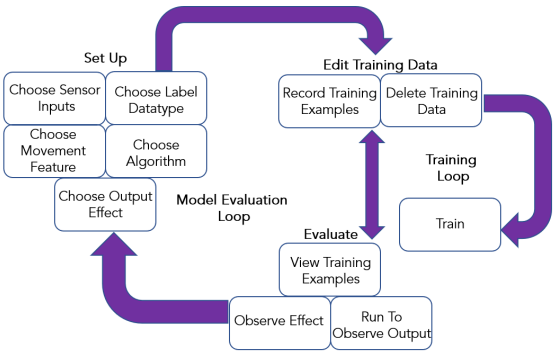

Figure 2: InteractML interactive machine learning process

- **Choose Movement Feature:** The user specifies which data they think are most relevant to the movement (i.e., the "features" to be sent as inputs to the model). In our example, the user might choose rotation and the velocity of rotation.
- **Choose Algorithm:** The user selects the most appropriate algorithm (classification, regression, DTW). Here, the user might choose classification.
- **Choose Label Datatype:** The user specifies the datatype that should be output by the graph. For the particle system, the user may choose a Float datatype to match the particle emission.
- **Configure Output Effect** The user configures what the movement will control in the virtual environment. In this case it will control the rate at which particles are emitted in the particle system.

4.2.2 *Edit Training Data.*

- **Record Training Examples:** The user decides on a pose or movement and its label then records an example of this, paired with the desired model output. In the wave example, imagine the user first records a very fast wave with the label 100, a medium wave with the label 50 and a slow wave with the label 1.
- **Delete Training Data:** The user can optionally delete and edit training data. For example, in the initial training the user may accidentally record the wrong label for an example and choose to delete the example.

4.2.3 Train. The user then tells the system to train, building the model from the training data.

4.2.4 *Evaluate.*

- **Run to Check Output:** The user can then run the model, executing the gesture or pose to check if the correct output is seen. In our example, the user would wave at different speeds to see if the output is correct. If the user has followed the steps above to set up and train the model, the user might observe fast waves generally output 100, medium wave generally output 50, and slow waves generally output 1.

Figure 3: InteractML graph for final clapping example set up

- **Observe Effect:** The user explores how the output of the model affects the behaviour in the rest of the system, and what this experience is like. For the waving example, the user would observe that the emission rate of the particle system jumps suddenly between 1, 50, and 100 as their waving speed changes. Perhaps this isn't what they had imagined; they might therefore choose to edit the graph by choosing regression rather than classification. They could then retrain on the data and run the modified model, observing that the outcome is the desired effect of a smooth transition between the particle emission rates. However, for certain waves the output might still be unexpected.
- **View Training Examples** The user can inspect the training data to see what values have been recorded. In the wave example, they might notice that the hand rotation values do not seem to be related to the speed of their wave; rather, only the velocity feature is relevant to this modeling task. The user then might decide to reconfigure the movement features in the graph then re-record their movements.

4.3 Node-Based Interface

InteractML is a node-based system allowing users to manipulate pre-programmed modules that make up a machine learning model. These nodes are connected together inside the InteractML window in Unity which gives a visual representation of each node. Node-based programming can enable non-programmers to understand the basic concepts of a system so that they can successfully utilise its functionality. It is particularly beneficial in creative applications where it has been shown to empower creative thinking [4]. For these reasons is it a prolific design choice in tools for creative practitioners. For instance, Troikatronix Isadora software[2] is a node-based tool used by dancers and other practitioners for real-time manipulation of digital media in performance. TouchDesigner—used by artists, performers and creative coders—is a node-based tool for real time interactive multimedia content.

Node-based programming also allows for flow-based design which helps users understand the logic of how data moves through the system. A flow-based node system can help create a mental model of how a system is working [24]. This is important in IML where the user needs to have an understanding of how movement sensors, machine learning algorithms, and training examples interconnect to create a model which will give their desired result. All without needing to grasp the complexities of the machine learning

[2]https://troikatronix.com

algorithms. This design allows for beginners to create simple interactions easily whilst including the functionality and flexibility to create bespoke, complex machine learning processes. A user can create a working model with as few as seven nodes. InteractML's modular structure is unique for IML, giving users the ability to create very specific machine learning configurations to match their individual need. This design supports users to create a working understanding of how the system works allowing for experimentation. (See Fig. 3)

4.3.1 Game Object Node. (See A in Fig. 3) represents a game object in the scene, this for example would be a VR Controller. From this node users can extract the position and/or rotation of that object, allowing for users to easily access input movement data from objects in their application.

4.3.2 Movement Feature Nodes. (See B in Fig. 3) extract data from a game object to be used as the input for defining a movement or pose. There are five Movement Feature Nodes: position, rotation Euler, rotation quaternion, distance to first input and velocity.

4.3.3 Data Type Nodes. (See C in Fig. 3) are used to set the type and value of an output label, to observe the output value of a system when running, or any point where the user wants to directly input data or observe data values. There are six types of Data Type Nodes that allow for full customisability– such as, a float (floating point value) or a Vector3 (a Unity specific type of array array storing 3 values– used for example when working with positional x,y,z values).

4.3.4 Teach The Machine Nodes. (See D in Fig. 3) is where users set up the data structure and training pairs, record training examples, edit training data and view training data. There are two types of Teach the Machine Node: "single" for training classification or regression, and "series" for training DTW.

4.3.5 Machine Learning System Nodes. (See E in Fig. 3) nodes are where the user controls the machine learning model, where they can connect training sets and run the model. There are three types of Machine Learning Node, one for each algorithm: classification, regression and DTW.

4.3.6 Script Node. (See F in Fig. 3) is a representation of any script that the user wants to either send data to from the graph, or retrieve data from to use in the graph. This allows data to easily transfer in or out of the graph and for users to easily add custom nodes into the graph.

4.3.7 Custom Control Nodes. enable users to control the functionality of the nodes using inputs such as a VR controller button or a keyboard button. For example to record examples pull the left controller trigger or to run to model press the left controller grip. They enable the user to easily customise how it is triggered in development or the final built experience.

4.4 Data

Users can import, export and edit their movement data from a JSON file with full control over when and how this is done. This data will exist only locally to the Unity project or executable unless the user chooses to share this data elsewhere. This sets InteractML apart from other free machine learning tools as users are in total control of their data.

5 VR INTERFACE

Feedback from participants building prototypes with InteractML during early workshop sessions (see section 3.1 highlighted the importance of offering a way to control events in the system with a VR interface. There were several reasons for this: without a VR interface or custom controls (see Section 4.3.7), the training process can be unwieldy and slow, as users would have to navigate to the graph view to initiate the data recording, put on the headset and grab the controllers, then perform the movement, and then remove the headset to navigate to the graph to stop recording data. Not only was this cumbersome, but it resulted in the recordings having extra data that was not part of their movement design. Participants sometimes worked in pairs to solve this problem, with one participant operating the graph whilst the other performed the movements. However, this option was not possible when creators worked alone.

This problem could not be solved simply by configuring the custom controls for triggering each node. Often participants would peer under their headset to see the graph, looking for information about the system's state, for instance to see whether the tool was receiving input from the sensors, whether they were recording data, or whether their model was trained and running. This demonstrates that it is also important to provide an indication of system state within the VR user interface when a user is performing movements to train the system and when running models in the virtual environment.

Nevertheless, creating a VR user interface to control IML presents several challenges. It is neither practical nor useful to create a replica of the whole graph in the headset, nor provide the ability to set up the graph itself in VR. Rather, users mostly expressed desire to control the events in the Teach the Machine and Machine Learning System nodes. Designing a VR interface for this specifically raises the following questions:"

- **How does the VR interface mirror and support the visual understanding that has been created from the node-based interface?**
- **What, specifically, is most important for users to be able to control in headset?**
- **How do users select which graph and nodes to control?** Users would need a way to tell the difference between each graph and node with a mapping of how this relates to their setup.
- **How is the state of the graph communicated to the user?** There are many different aspects of system state that could be communicated to the users, for instance the number of training examples, whether the graph is is currently recording examples, whether the model is trained, whether the model is running, how many examples the model is trained on, etc.

5.1 Design

In the VR interface, each graph present in the scene was represented by an InteractML icon appearing in the virtual environment (see

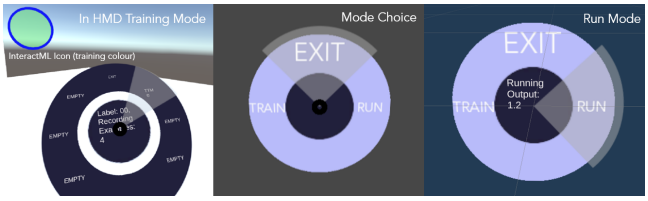

Figure 4: InteractML VR interface in HMD train mode, mode choice and run mode

figure 4, left). To select a graph the user points and clicks on the icon using the joystick on the VR controller to bring up a radial menu (See Fig. 4). Users can either choose to go into "train" mode or "run" mode. In train mode, the user has access to all the teach the machine nodes in the graph. By selecting a teach the machine node (i.e., the training set), users can record or delete examples by clicking VR controller buttons (these are customisable from a universal input setup node in the graph). Once users have recorded examples, they can go back to run mode to test the model that has just been trained from those examples. The user can quickly and easily switch between the modes, recording training data then quickly moving to run mode to evaluate the models ability to recognise the movement.

The state of the system is reflected both in the colour of the icon and the centre of the radial menu. This is important as the radial menu is connected to the controller meaning it may be out of headset view in some movements. On the radial menu the state of the model appears as text. In train mode the number of training examples is displayed as well as whether it is currently recording. In run mode whether it is currently running and the current output is displayed. The icon changes colour in reflection of whether there is a node recording, model trained or model currently running. (See Fig. 4)

5.2 Recommendations

Participants from later workshop sessions where the tool included the VR interface felt positive that being able to record, delete movement examples and evaluate models in the virtual environment streamlined their design process, as it was less interrupted by removing or putting on the VR headset. Users made less mistakes as they could easily see when they were recording, how many samples were recorded or when the model was running and so on. Users did request to be able to edit the label associated with the movement example they wished to record— as this would further streamline the process so an entire training set could be recorded without leaving the virtual environment.

6 CASE STUDIES OF CREATIVE WORK BUILT USING INTERACTML

InteractML has subsequently been used in a diverse set of creative work for VR. The following projects show how artists working with the tool can use a range of input sensors set-ups, for different types of movements and interacting with different elements within a virtual environment.

Figure 5: Artist working with an IMU movement sensor attached to ankle bells. Image courtesy of the artist. Bushra Burge, 2021

6.1 Performance in VR: Detecting Dance Phrases from Wearable Sensors

Bushra Burge was interested in using wearable sensors stitched into costumes worn for traditional Bangladeshi folk dancing. She experimented with Inertial Measurement Unit (IMUs) sensors, these devices have built-in accelerometer and gyroscope sensors that measure velocity and orientation movement data. Being small and lightweight these devices are a suitable choice for artists or performers wanting to attach sensors to performers, dancers or audiences to detect the dynamics of their movements but avoiding constricting movement with a bulky device or cables. In this case, the artist used IMUs with InteractML to detect dance phrases from Bangladeshi folk dancing (See Fig. 5).

Audiences would experience this work through an Oculus Quest VR headset that presented a Bangladeshi lake scene. The artist configured different dance phrases, such as stamping to a rhythm, to effect various elements within the virtual environment: including triggering animations of a model of a Kingfisher bird, the light intensity in the scene and the dynamics of a cloth that represented a traditional Sari dress (See Fig. 6). This artist worked individually, using a Regression algorithm to train the system to recognise changes in velocity of foot movements. There were some issues in this process as the data from the IMU sensor was not stable enough to provide consistent results whilst performing the training movements. To remedy this issue, the InteractML tool provides a way to input data ranges to train a model manually, in a way which is quick, straightforward and follows the same workflow as performing the movements. Once the artist observes the range of values the IMU sensors was measuring, they could enter the boundary values with the appropriate output pairing. In their learning process they adapted example graphs and scripts adapting these for their use case. As they with most creative coding they find solutions similar and modify for their work.

InteractML made this possible by allowing the artist to easily hook-in the IMU sensors into the graph, quickly and easily experiment with using different dance phrases to control different elements within the scene, all with minimal programming needed.

Figure 6: Digital rendering of artists virtual Bangledeshi lake scene showing kingfisher and sari animations. Image courtesy of the artist. Bushra Burge, 2021

6.2 Interactive VR Artwork: Recognising hand movements from Oculus Quest controllers

Sara Tirelli and Sergio Bromberg took a unique approach to using InteractML to recognise movement, the work compares the audiences movement to a training set of human movements and artificially produced movement data to determine whether or not the audience is human or a machine. Here, the artists were inspired by reflecting on a machinic understanding of human movement that machine learning captures.

Using the Oculus Quest hand controllers, the audience is prompted with a graphic of female face asking in a robotic voice to prove that you are a human by moving (see Fig. 7), each users movements is then added to the training set of human movements. This approach lead the artists to experiment with what type of movement feature better indicated 'human movement' to a machine learning system. First the artists considered recording the velocity of the user as they considered the speed of movement characteristic of dynamics that might describe expressive movement. The artists also considered comparing the distance between a human movement and an artificially recorded movement, stating that if the inputted movement was too 'perfect' then the user must be a machine.

This project was produced by a pair of artists collaborating to use InteractML, as such the process they used to train their system was collaborative. One artist taking the role of operating the graph- indicating vocally when they start and stop recording data, the other artist performs the movement once the operating artists instructs them they have started recording- this means they were able to get into the correct position ready for the performance. Many collaborating artists chose to work in this way as it reduced the cognitive load of the training process, the artists performing the movement could concentrate on the movement itself while the operating artist could make sure the system was configured correctly, receiving the correct data and start and stop the recording accordingly.

The artists configured InteractML to decide whether the audiences movements are human or machine, then used the outputted decision to drive a state machine from within the Unity game engine to move onto the next appropriate part of the sequence in their virtual experience (see Fig. 8). These ideas would not be possible without the ability InteractML provides to train machine learning models within the infrastructure that Unity provides.

Figure 7: Digital rendering of artists work using depth capture and shaders in Unity game engine. Image courtesy of the artists. Sara Tirelli and Sergio Bromberg, 2021

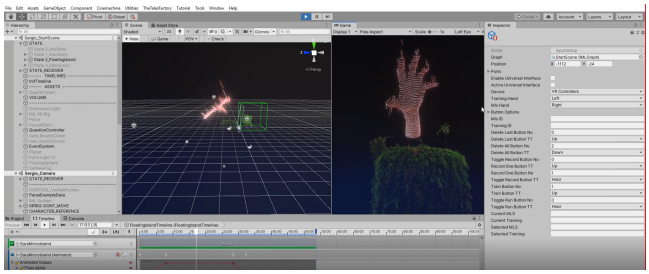

Figure 8: Artists work from within Unity game engine using InteractML to drive a state machine. Image courtesy of the artists. Sara Tirelli and Sergio Bromberg, 2021

6.3 Machine Learning Algorithm Supporting Different Styles of Movement Interaction in the Virtual Environment

As we have seen in Section 4.1 the different machine learning algorithms offered by InteractML tend to be better suited to supporting different types of movements. In addition, although the output values are completely customisable in data type (for example Floats, Vector3, Strings), each algorithms output also tends to be better suited to controlling different elements in a virtual environment. Some examples of what our users produced for each algorithm are detailed below.

6.3.1 Regression. Owing to the continuous output values a Regression algorithm produces, this allows finer controls of elements in a virtual world, for example granular control of the intensity of light in a scene. Artists using a Regression algorithm used movement and output control pairings such as:

- Moving a VR controller from one position or orientation to another to control properties in sound or music such as the volume or pitch
- Changing between hand poses to control the properties of particles systems. Such as the amount of particles, or the direction and strength of attractive or repulsive forces

- Changing the velocity of head movements (when wearing a VR headset) to control material and shader properties such as colour, opacity and specularity.

6.3.2 Classification and Dynamic Time Warping (DTW). These algorithms are well suited to recognising distinct poses (Classification) or gestures (DTW) in order to trigger a specific event in a virtual environment. Artists using a Classification or DTW algorithm used movement and output control pairings such as:

- Using hand tracking to detect different hand poses to interact with virtual characters by triggering animations of expressions
- Using Kinect skeleton tracking to recognise dance phrases to trigger musical phrases

7 RESULTS AND DISCUSSION

7.1 Learning The Tool

We observed that many of the participants did a lot of their learning through adapting examples. They would edit an example graph to try to fit into their use case adapting each stage of the graph until they produced their desired outcome. In interviews and discussion they would cite these as an integral part of their learning process 'I don't think I could have done it without the examples. Now that I've, like, I've seen a copy of it. I can apply it'. If something went wrong in the graph rather than referring to the documentation they would compare their work to these examples sometimes working backwards from the graph to find where they went wrong. This behavior was most seen in users from artistic or dance background with a less experience in programming and Unity. This opportunistic approach [2] is a common way of learning to code and creating work for many programmers and creative practitioners. Developing well documented examples with tutorials of more complex set ups would help in supporting this user group to engage with the tool and be able to experiment in their creative practice with it.

7.2 Process

Workshops and hackathons gave us a great insight into how users first encounter the tool and the initial stumbling blocks of its use. The longitudinal study of the tool through the residencies allowed for discoveries in how artists used the tool as part of their practice. This gave us insight into real world use beyond what was just initial learning problems to what misunderstandings and difficulties persisted beyond the initial learning phase. Whilst in the workshops participants sometimes struggled to grasp the machine learning concepts and the flow of the graph beyond the the initial prototypes, within weeks the residency artists had developed a solid mental model of the system which was evident in our weekly interview with them.

Many of the artists involved in the residency and hackathons have continued using the tool in their work. We have continued the community and support on a Discord channel. Through this we have seen how the tool has influenced how interaction is designed in their virtual environments and their artistic practice.

8 CONCLUSION

Advances in sensor and movement tracking technologies are enabling full body and movement-based interactivity for immersive applications. Implementing movement-based interaction designs, however, is currently complicated, effectively limiting who can create immersive applications that use movement-based interaction techniques. To alleviate this issue, we have presented an IML tool InteractML, which enables users to rapidly prototype movement-based interactions in their immersive applications. We believe that this does not simply show the usefulness of a particular tool, but of a general approach that uses interactive machine learning that enables the design and implementation of movement interaction in an embodied way, i.e. designing by moving[14]. This is particularly relevant for virutal reality because movement interaction is so common and contributes to the illusions of presence, plausibility and embodiment[13, 33].

InteractML can also be implemented for training in built experiences meaning users could design custom personal interactions. InteractML enables users to be included not only in the design of the interaction but also in the implementation. Its quick iterative process means custom interaction could be easy to implement. This has great scope for making immersive technologies more accessible to more people.

For this version of InteractML, we have built it as a plug-in for Unity game engine software. Motivated to make IML for movement interaction in immersive creative work more accessible, particularly to creators without a programming background, we considered this software suitable to build InteractML within. Accessible, affordable, well documented and supported by the developer community and allows for the creation of fully formed immersive applications with full control over every aspect of the virtual environment, mostly without the need to do any scripting (although this option is there for developers if they wish). More importantly for creative practitioners developing immersive work, there is the possibility to easily hook in different devices into their projects, such as different VR systems and movement sensors, it is also possible to bring in virtual assets from other creative applications, and so serves a range of users and applications. However, the downside to such customisability is that often set-up times are long and creators working with InteractML that are not already familiar with Unity are faced with a steep learning curve to get started.

As a technique, machine learning is currently obscure and even domain experts have difficulties as they consider methods "black box", and have difficulties in interpreting results [26]. In this sense, creators that are not familiar with machine learning would need to learn a whole new way of working for designing movement interaction. Yet, we believe in the ethos and design of InteractML, alongside the methods and instructional documentation that comes with it, we have facilitated an easy overcoming of this learning curve by fundamentally rethinking machine learning in terms of usability and customisability. In the development of InteractML we advocate and support a new generation of creators to use machine learning to design movement interaction in their immersive creative work.

REFERENCES

[1] Sarah Fdili Alaoui. 2019. Making an Interactive Dance Piece. *Proceedings of the 2019 on Designing Interactive Systems Conference*. https://doi.org/10.1145/3322276.3322289

[2] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming. (2009), 1589–1598. https://doi.org/10.1145/1518701.1518944

[3] V. Braun and V. Clarke. 2006. Using thematic analysis in psychology. 3 (2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[4] Scott Andrew Brown, Michael Gratton, and Bronwen Williams. 2019. A Visual Programming Tool for Creative Practice Pedagogy in Embodied Interaction and Media Arts. *Proceedings of the 31st Australian Conference on Human-Computer-Interaction*. https://doi.org/10.1145/3369457.3369495

[5] Jamie Bullock and Ali Momeni. 2015. Ml.Lib: Robust, Cross-Platform, Open-Source Machine Learning for Max and Pure Data. In *Proceedings of the International Conference on New Interfaces for Musical Expression* (Baton Rouge, Louisiana, USA) *(NIME 2015)*. The School of Music and the Center for Computation and Technology (CCT), Louisiana State University, Baton Rouge, Louisiana, USA, 265–270. https://doi.org/10.5555/2993778.2993845

[6] Baptiste Caramiaux, Alessandro Altavilla, Scott G. Pobiner, and Atau Tanaka. 2015. Form Follows Sound. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/2702123.2702515

[7] Carlos Gonzalez Diaz, Phoenix Perry, and Rebecca Fiebrink. 2019. Interactive Machine Learning for More Expressive Game Interactions. *2019 IEEE Conference on Games (CoG)*. https://doi.org/10.1109/CIG.2019.8848007

[8] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive machine learning. *Proceedings of the 8th international conference on Intelligent user interfaces - IUI '03*. https://doi.org/10.1145/604045.604056

[9] Rebecca Fiebrink, Perry R. Cook, and Dan Trueman. 2011. Human model evaluation in interactive supervised learning. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/1978942.1978965

[10] Rebecca Fiebrink, Dan Trueman, and Perry. R. Cook. 2009. A Meta-Instrument for Interactive, On-the-fly Machine Learning. *New Interfaces for Musical Expression (NIME)*, 280–285.

[11] John C. Flanagan. 1954. The critical incident technique. *Psychological Bulletin* 51 (1954). Issue 4. https://doi.org/10.1037/h0061470

[12] Nicholas Gillian and Joseph. A. Paradiso. 2014. The gesture recognition toolkit. *The Journal of Machine Learning Research* 15 (2014), 3483–3487. Issue 1.

[13] Marco Gillies. 2016. What is Movement Interaction in Virtual Reality for?. In *Proceedings of the 3rd International Symposium on Movement and Computing - MOCO '16*. ACM Press, New York, New York, USA, 1–4. https://doi.org/10.1145/2948910.2948951

[14] Marco Gillies. 2019. Understanding the Role of Interactive Machine Learning in Movement Interaction Design. *ACM Transactions on Computer-Human Interaction* 26 (2 2019). Issue 1. https://doi.org/10.1145/3287307

[15] Marco Gillies and Andrea Kleinsmith. 2013. Interactive Machine Learning for Virtual Agents. *Intelligent Virtual Agents: 13th International Conference, IVA*, 29–31.

[16] Clarice M Hilton, Carlos Gonzalez, Ruth Gibson, Michael Zbyszyński, Phoenix Perry, Rebecca Fiebrink, Nicola Plant, Bruno Martelli, and Marco Gillies. 2021. InteractML : Node Based Tool to Empower Artists and Dancers in using Interactive Machine Learning for Designing Movement Interaction. (2021), 1–5.

[17] Caroline Hummels, Kees C. J. Overbeeke, and Sietske Klooster. 2007. Move to get moved: a search for methods, tools and knowledge to design for expressive and rich movement-based interaction. *Personal and Ubiquitous Computing* 11 (10 2007). Issue 8. https://doi.org/10.1007/s00779-006-0135-y

[18] Kristina Höök. 2018. *Designing with the Body: Somaesthetic Interaction Design*. MIT Press.

[19] Robert Jacob, Audrey Girouard, Leanne Hirshfield, Michael Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. 2008. Reality-Based Interaction: A Framework for Post-WIMP Interfaces. *SIGCHI conference on Human factors in computing systems*.

[20] David Kirsh. 2013. Embodied cognition and the magical future of interaction design. *ACM Transactions on Computer-Human Interaction* 20 (3 2013). Issue 1. https://doi.org/10.1145/2442106.2442109

[21] William Latham, Stephen Todd, Peter Todd, and Lance Putnam. 2021. Exhibiting Mutator VR: Procedural Art Evolves to Virtual Reality. *Leonardo* 54 (6 2021). Issue 3. https://doi.org/10.1162/leon_a_01857

[22] Lian Loke and Toni Robertson. 2010. Studies of Dancers: Moving from Experience to Interaction Design. *International Journal of Design* 4 (2010). Issue 2.

[23] Raul Masu, Nuno N. Correia, Stephan Jurgens, Ivana Druzetic, and William Primett. 2019. How do Dancers Want to Use Interactive Technology? *Proceedings of the 9th International Conference on Digital and Interactive Arts*. https://doi.org/10.1145/3359852.3359869

[24] D. A. Norman. 2010. Natural user interfaces are not natural. *interactions* 17 (2010), 6–10. Issue 3.

[25] Claudia Núñez-Pacheco and Lian Loke. 2018. Towards a technique for articulating aesthetic experiences in design using Focusing and the Felt Sense. *The Design Journal* 21 (7 2018). Issue 4. https://doi.org/10.1080/14606925.2018.1467680

[26] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. 2008. Investigating statistical machine learning as a tool for software development. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*. https://doi.org/10.1145/1357054.1357160

[27] N. Plant, R. Gibson, C.G. Diaz, B. Martelli, M. Zbyszyński, R. Fiebrink, M. Gillies, C. Hilton, and P. Perry. 2020. Movement interaction design for immersive media using interactive machine learning. In *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3401956.3404252

[28] Nicola Plant, Clarice Hilton Goldsmiths, Marco Gillies Goldsmiths, Michael Zbyszyński, Rebecca Fiebrink, Phoenix Perry, Carlos Gonzalez Diaz, Ruth Gibson, Studio Gibson, and / Martelli. 2020. Programming by Moving: Interactive Machine Learning for Embodied Interaction Design. *Workshop "The UX of Interactive Machine Learning" at NordiCHI '20* (2020), 1–4.

[29] Nicola Plant, Clarice Hilton, Marco Gillies, Rebecca Fiebrink, Phoenix Perry, Carlos González DÍaz, Ruth Gibson, Bruno Martelli, and Michael Zbyszynski. 2021. Interactive Machine Learning for Embodied Interaction Design: A tool and methodology. *TEI 2021 - Proceedings of the 15th International Conference on Tangible, Embedded, and Embodied Interaction* (2021). https://doi.org/10.1145/3430524.3442703

[30] Anna Rizzo, Katerina Eh Raheb, Sarah Whatley, Rosa Maria Cisneros, Antonio Camurri, Vladimir Viro, Jean-Marc Mato, Stefano Piana, Michele Buccoli, Amalia Markatzi, Pablo Palacio, Oshri Even, Augusto Sarti, Yannis Ioannidis, and Edwin-Morley Fletcher. 2018. WhoLoDancE: Whole-body Interaction Learning for Dance Education. *Workshop on Cultural Informatics*, 41–50.

[31] Elena Márquez Segura, Laia Turmo Vidal, Asreen Rostami, and Annika Waern. 2016. Embodied Sketching. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/2858036.2858486

[32] Richard Shusterman. 2012. *Thinking Through the Body: Essays in Somaesthetics*. Cambridge University Press.

[33] Mel Slater. 2009. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society B: Biological Sciences* 364 (12 2009). Issue 1535. https://doi.org/10.1098/rstb.2009.0138

[34] Robert Wechsler, Peter Dowling, and Frieder Weiß. 2004. EyeCon–A motion sensing tool for creating interactive dance, music, and video projections. *SSAISB Convention*.

[35] Danielle Wilde, Anna Vallgårda, and Oscar Tomico. 2017. Embodied Design Ideation Methods. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3025453.3025873

[36] Michael Zbyszynski, Mick Grierson, and Matthew Yee-king. 2017. Rapid Prototyping of New Instruments with CodeCircle. *Proceedings of the International Conference on New Interfaces for Musical Expression* (2017), 227–230.