# Block Jam: A Tangible Interface for Interactive Music

Henry Newton-Dunn
Interaction Laboratory,Sony Computer Science Laboratories, Inc.
3-14-13 Higashigotanda
Shinagawa-ku, Tokyo 141-0022, JAPAN
henry@csl.sony.co.jp

Hiroaki Nakano
Creative Development Group,Sony Design Center
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141-0001, JAPAN
nakano@dc.sony.co.jp

James Gibson
Human Interface Design Group,Sony Design Center Europe
The Heights, Brooklands
Weybridge , Surrey KT13 0XW, UK
gibson@dc.sony.co.jp

## ABSTRACT

In this paper, we introduce Block Jam, a Tangible User Interface that controls a dynamic polyrhythmic sequencer using 26 physical artifacts. These physical artifacts, that we call *blocks*, are a new type of input device for manipulating an interactive music system. The blocks' functional and topological statuses are tightly coupled to an ad hoc sequencer, interpreting the user's arrangement of the blocks as meaningful musical phrases and structures.

We demonstrate that we have created both a tangible and visual language that enables both the novice and musically trained users by taking advantage of both their explorative and intuitive abilities. The tangible nature of the blocks and the intuitive interface promotes face-to-face collaboration and social interaction within a single system. The principle of collaboration is further extended by linking two Block Jam systems together to create a network.

We discuss our project vision, design rational, related works, and the implementation of Block Jam prototypes.

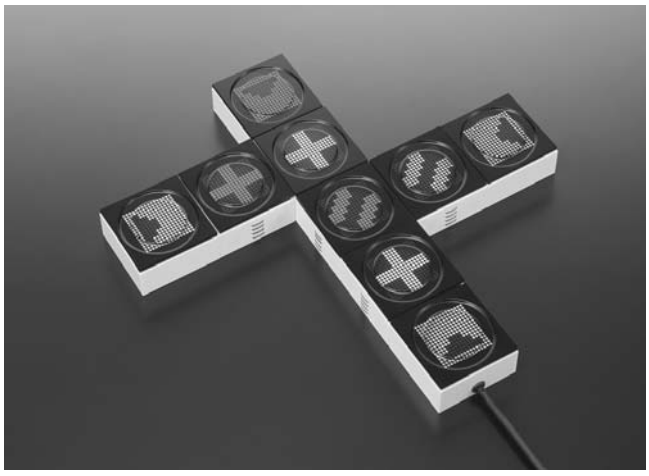

**Figure 1. A cluster of blocks, note the mother block on the bottom right**

## Keywords

Tangible interface, modular system, polyrhythmic sequencer.

## 1. VISION

We believe in a future where music will no longer be considered a linear composition, but a dynamic structure, and musical composition will extend to interaction. We also believe that through the introduction of such media the divisions of composer, performer, and audience will be blurred.

Our second aim is to put the group experience back into music. We understand that the musical experience changes with technology. Musical technology provides greater control, more possibilities and greater access to the beginner or novice.

The way we receive and listen to music is also changing, not just in terms of Low-Fi to Hi-Fi or Phono to Tape to CD to MD to MP3, but also in terms of experience, music is moving from a social experience to a personal experience, from campfire to orchestra to living room to Walkman. Degrees of separation have occurred between the composer and the performer, the performer and the audience. This trend continues. Inversely, technology is moving towards community, towards the group, towards the network.

Block Jam is the first in a number of projects aimed at addressing this disparity within the technologically mediated musical experience.

## 2. DESIGNING BLOCK JAM

This project was initially created in December 2001 as an exploration into both new types of musical experience and novel tangible interaction techniques. The project was a collaboration between designers from Sony CSL Interaction Laboratory and designers from the Sony Design Center. Decisions were weighted towards functional relevance, with a view towards an aesthetic outcome. Designing the project fell into two distinct areas, designing the musical experience and designing the tangible interface. Though separately described below – these processes were actually designed concurrently, decisions made in one area often affecting the other.

### 2.1 Designing the Musical Experience

Through a musician's physical gesture, his vocal or instrumental performance, we are able to unlock sensations, emotions and thoughts. Musicians develop these skills through formal study and practical application. Music is not difficult to appreciate or enjoy, but difficult and often frustrating to create or play, especially for the untrained.

Interactive music has the potential (and the promise) to help release us from this difficulty and frustration. In 1970, Mathews and Moore created the first interactive music program called *Groove* that proved to be experientially "almost irresistible"[[3]]. Since Groove, there have been many attempts to design and implement new computationally mediated musical systems and interfaces. It should be noted that this paper is not concerned with computer music systems, but rather systems for interacting with music.

These systems and interfaces fall into many categories; from improvisational or continuator systems [[4]], Hyperinstruments [[5]], to simpler systems that are usable by the novice. There are a host of commercial available

experiences like those created by new media companies such as *Hi-Res*[17], or software titles such as *Rez* [15] or *Parapa the Rappa* [16] for the PlayStation 2 platform.

Our goal for this project was to create an interactive music system that would both engage the novice user and the musically trained, and promote collaboration and communication among the participants.

We can think of music as a communicative conduit, as a performance, and as a mono-directional expression. We prefer to consider music as bi-directional (or even omni directional) since music is about teamwork and group interaction. It is a promoter of communication, excitation, and mediation, and most importantly to us, creative interplay. This notion can be identified in the compositions of artists such as Arnold Schoenberg, Karlheinz Stockhausen and notably John Cage. Umberto Eco speaks of the Poetics of the Open Work [2], in which each layer of interpretation adds to the completeness. Therefore, he suggests that a piece of music remains incomplete (or open) until interpreted from the score by a musician, then in turn by the audience.

Bearing this in mind, we decided on two key points to be used as anchors for our design of a musical experience:

1. The collaborators have a common musical frame of reference (much like Schoenberg's twelve tone system) within which they can each assume a different role.
2. The musical frame of reference should be compos-able, and must remain open (interpretable).

These translated into an asymmetric network of interactive music applications, emulating the structure of a band (e.g. a drummer app, guitarist app, bassist app, etc).

This suggested a need for a collaborative interactive music framework. Such a framework would allow composers a new creative space for themselves and their audience – the participants - to explore.

Block Jam is the physical realization of an application that could be used for such an interactive music framework. As a starting point, we divided the potential applications into two categories – sequencer type applications and gestural type applications.

Sequencer type applications are comprised of modular musical elements that are arrange-able and interchangeable within a sequential context to create a novel musical outcome such as Toshio Iwai's Composition on the Table [[9]].

Gestural applications by contrast are based on the modulation of tone, pitch or timbre to create an expressive output. An example is Ivan Poupyrev's *Augmented Groove* [[8]], where a user modulates and mixes techno compositions by manipulating real LP records tracked by a PC using visual markers. Effects, filters, and samples are triggered according to a record's movements (up-down, rotation, tilting, and shaking).

We considered that a necessary limitation to our design was that it must be stylistically neutral. Unlike Augmented Groove, we did not want to tie the applications or interface to be organized around a particular genre, particularly if our longer-term goal is to build a compose-able interactive music framework.

For our first application, Block Jam, we chose the sequencer type. We felt that it would be less problematic to create functional mappings and far easier for it to remain neutral and open.

## 2.2 Designing the Tangible Interface

When it came to designing the interface for Block Jam we started by looking at interactive toys and sound devices currently available for children, good examples being *SoundBlocks*, *Musini* and *Phonics Tiles* available from Neurosmith[[6]] (who produce children's products based on research into linguistics and cognitive science).

Children's toys tend to be physically organized and actuated, have meaningful use of shape and color, are iconic and often include sound. Children's toys also have immediacy. These were all characteristics we hoped to include and emulate in our design, the major difference being that our target users were adults, who require a much greater sophistication than children in order to be experientially engaged.

Other toys of particular note were Friedrich Froebel's Gifts (#2 to #6), which are sets of wooden blocks that vary in complexity, shape and color. The sets were designed to help children learn, explore, and create. The shapes, being primitives, hold no direct meaning, but when combined they can create endless iconic forms such as houses, castles, towers and bridges. The forms act as a mechanism for eliciting experience. This suggested the possibility of a programmatic equivalent - a modular tangible interface whose rules are simple enough to be easily understood, but whose outcome is potentially complex enough to be continually engaging. Perhaps we could use blocks to program sound.

The idea of using modular tangible blocks for building programmatic structures is not new. We formed two approximate categories for our assessment of tangible interfaces, functionally heterogeneous and functionally homogeneous. Functionally heterogeneous tangible interfaces are where different physical artifacts are used to represent different functions. Functionally homogeneous tangible interfaces consist of a single type of physical artifact with a single function – typically, many of these artifacts could be interlocked to produce programmatic outcomes.

### *2.2.1 Examples of Functionally Heterogeneous Modular Tangible Interfaces*

FitzMaurice identified the possibility of using tangible objects (Graspable User Interface) in *Bricks* [1] to extend the Graphical User Interface (GUI). Different tangible artifacts (as metaphoric transducers) could be used to represent functional elements of a GUI. This promoted the notion of a *space-multiplexed* interface, as opposed to a *time-multiplexed* interface (such as the mouse).

Brygg Ullmer's *MediaBlocks* [10] further extended this work by using tangible objects for the containment, transport and manipulation of a digital media system. Wooden blocks known as *phicons* (physical icons) were used "as a seamless gateway" to control the GUI.

Jun Rekimoto's *DataTiles* [[12]] integrated both the graphical and the physical user interface. It promoted the use of tagged transparent objects as interaction modules, which mixed visual feedback with physical interactions and created a physical language for combining multiple tiles to create a "sentence". This system relied on screen used as a base on which the transparent tiles were placed.

Mitchel Resnick's *Behavior Construction Kit* [7], which allowed children to build behavioral machines using sensor bricks (e.g. IR) and output bricks (a motor) connected to a programmable brick. Interestingly the project was envisioned as a means of making ubiquitous computing

accessible to children, but was later realized as Lego Mindstorms, which is sold as a "Robotics Invention System"[[11]].

### *2.2.2 Examples of Functionally Homogeneous Modular Tangible Interfaces*

Frazer, J.H. et al identified the first modular tangible systems in 1982 [[13]] with two systems, *Tree Searcher* and notably *Intelligent Beer Mats*. The Intelligent Beer Mats were used to describe 2D topological configurations of flat square physical units to a host PC. The system was intended as a prospective application for the architectural building process.

Mathew Gorbet's Triangles [[14]], similar to the Intelligent Beer Mats consisted of a set of identical flat, plastic equilateral triangles, each with a microprocessor and unique ID, again, allowing a host PC to rebuild a given topology. The triangular shape was chosen because it held no meaning, thus the Triangles were applied to varying media oriented applications. Unfortunately, the Triangles were limited by a lack of an integrated output and input interface

## 2.3 Interface Requirements for Block Jam

Many of the tangible interfaces that we looked at were:

1. Reliant on a separate GUI [1, 9,12]
2. Had stateless interface artifacts, (they just acted as passive handles) [1, 9, 13, 14]
3. Had no, or limited means of direct digital interaction with an artifact itself [1, 9, 13, 14]

A clear advantage of a digital artifact is that it can be functionally dynamic. Our primary requirement for the Block Jam interface was that each tangible artifact would have a dynamically change-able state. This in turn suggested a need for a means of displaying the state, and a means of changing it.

Early sketches included a large variety of possible functions/states [Figure 2]. These were grouped and then mapped to a variety of shapes, to create a visual/shape interaction language. The language included different types of input mechanisms and function groups.

The input mechanisms included a variety of state and stateless dials, wheels, buttons, and sliders. The functions were grouped into effect (DSP) modulators, route functions – including route splitters and route mergers, sound triggering functions, and more compositionally algorithmic functions, such as automatic key changing.

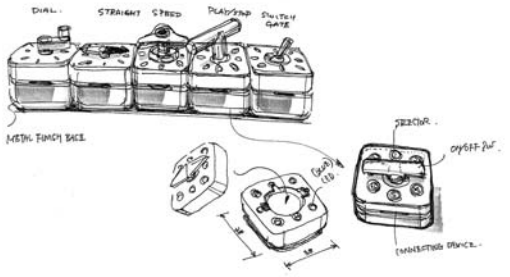


**Figure 2. Early sketch (describing a more complex interaction language)**

However, it soon became apparent that the potential complexity of assigning separate functions to separate objects could become overwhelming for the user within the context of our design. We found that designing an interface that makes music easily controllable at a higher level did not necessarily mean adding new functionality (as is normally the case with interface design), but rather it meant refining and rationalizing existing functionality.

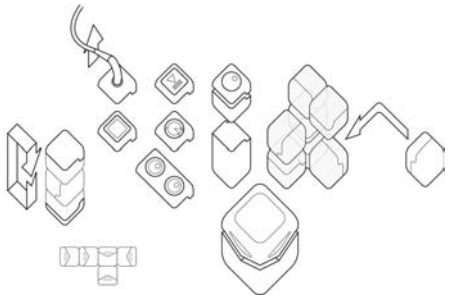

**Figure 3. Later sketch (exploring the ramifications of stacking blocks in 3D)**

We started a continuous process of removing functionality, attempting to boil it down to the absolute bare minimum. During this process, we realized that the tangible artifacts' different functional states, did not have to be mapped to different shapes. We had so few functions left that the original shape groupings became meaningless. Each artifact's state or function could be represented by its display mechanism alone. One primitive shape with a common set of input and output mechanisms could be used.

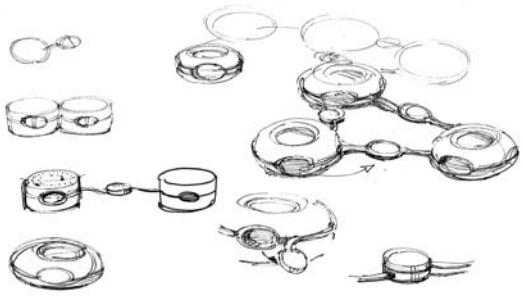

**Figure 4. Sketch of a non-tessellating structure**

We explored tessellating and non-tessellating structures and different stacking, and layering solutions. Tessellating shapes were attractive because they're easy to assemble, input mechanisms are accessible, and connected shapes could be easily read. However, they're best suited to 2 dimensional arrangements – 3 dimensional arrangements (though exciting) had problems of visual and input mechanism occlusion [Figure 3]. Conversely, non-tessellating structures [Figure 4] were equally suited to 2D and 3D, especially when self organizing [Figure 5], but suffered from bad input mechanism accessibility and poor shape readability.

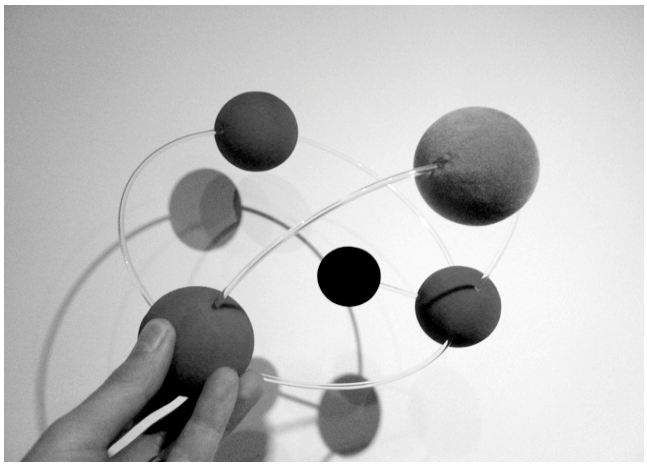

**Figure 5. Mock-up of a self organizing structure using plastic rods to hold the nodes in tension**

Finally, we chose a form factor based on a square shape rather than a triangular, or a circular shape because it implied directionality. A message arriving at one side could easily be imagined passing through to the opposite side. A 3-color LED matrix (16*16) was chosen as the display mechanism for its simplicity, and two input mechanism were added, a button for toggling the state/function, and a dialing gesture for choosing sound [Figure 6].

**Figure 6. The final block design, note the milled groove in the surface and the connectors on the side**

### *2.3.1 The Anatomy of a Block*

Externally, a block consists of a white ABS resin box, with a black acrylic top [Figure 6]. It has connectors on the side, which connect power and pass serial data. The black acrylic top has a circular milled groove in it's surface, and an LED matrix below its surface that is visible through the acrylic when illuminated. The LED can light up red, green, or orange. Incidentally, the black acrylic was chosen because it helps with the LED visibility.

The blocks can sense two types of input, a click, and a dialing gesture. The click is measured via a simple sprung button placed on the under side of the circuit/LED matrix, so, when the acrylic top is pressed so the button is pushed. The dialing gesture is sensed via an array of eight infrared optical reflectors arranged two to a side along the path of the milled groove. As the user dials, his/her finger is guided by the groove.

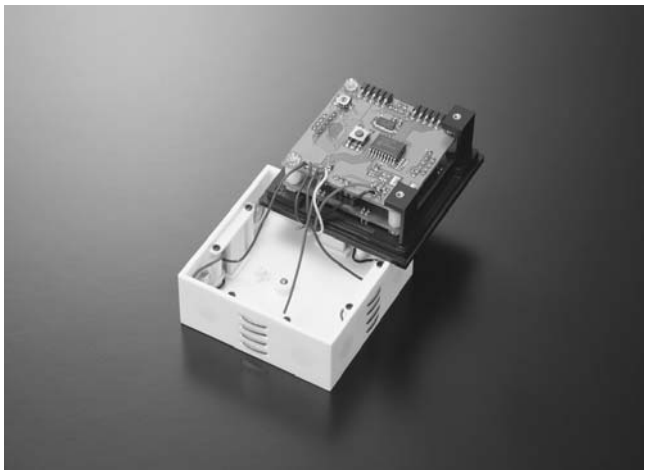

**Figure 7. Inside a block, note the button for detecting the click**

The blocks physically connect to each other by means of two hidden magnets on each side of the box. Each block has a unique ID, allowing the blocks to communicate over a common bus. Data is sent from a block when it is connected, when a neighbor has been disconnected and when a block has sensed a user's interactions. Information received updates a block's status and tells it which icon to display on the LED matrix.

The data passing between the blocks and the PC is handled by a PIC microcontroller [Figure 9] in each block communicating over a common bus. The LED matrix, the click input and the array of optical reflectors used to detect the dialing gesture were all handled by second PIC microcontroller. The only calculation performed by the blocks was the dialing to keep the data flow on the common bus to a minimum (otherwise risking too much noise). All other functions were handled by the controlling PC, which told the blocks what to display and when to display it.

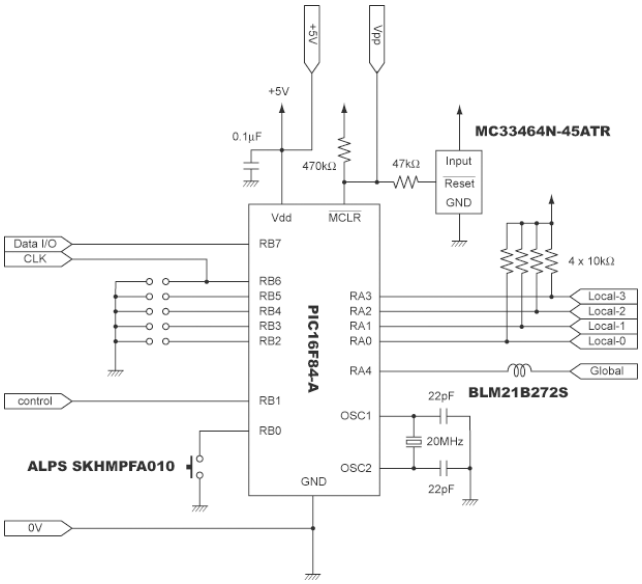


**Figure 8. Schematic showing a blocks I/O system**

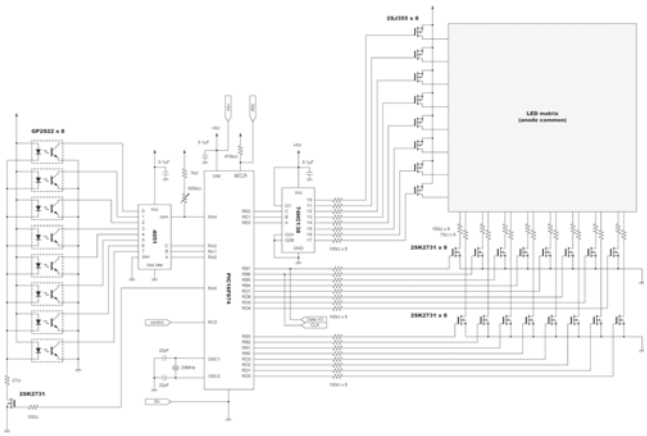

**Figure 9. Schematic showing the input system (click and dial) and the LED matrix control**

## 3. INTERACTING WITH THE BLOCKS

We created two types of block, *play blocks* and *path blocks*. Play blocks start, stop and control the speed of a sequential instance or marker known as a *cue ball*. Path blocks control the route that a cue ball travels. A play block can only be connected on one side (towards which the play icon is pointing), where as a path block can be connected on four sides.

As the blocks are added together, they form a *cluster* [figure 1]. A cluster is connected to a PC (for computation only) by a tethered play block (known as the *mother block*). The tether (wire) provides common power and a common serial connection to the cluster.

A cue ball *bounces* from one block to another within a cluster according to rules determined by the state of each path block. Every block metaphorically contains a sound, so as a cue ball (or cue balls) bounce from block to block it determines a sequential composition, creating music.

### 3.1 Starting and Stopping a Cue Ball

Clicking a play block toggles a cue ball instance on or off. Therefore one click starts a cue ball and another stops it (in addition to toggling, a timer is measuring the length of the clicking action). This measurement allows for different speeds of the cue ball to be started. A quick click starts a quick cue ball, a medium length click starts a medium speed cue ball, and a slow click starts a slow cue ball (the amount of time a click is held down while stopping a sequence has no functional relevance – the cue ball just stops).

Since events in our system are quantized to a fixed beat/grid, this translated as a click of less than one second starting a cue ball that bounced every beat. A click that lasted from 1 to 2 seconds launched a cue ball that bounced every two beats. A click longer than 2 seconds launched a cue ball that bounced every four beats.

Multiple play blocks can be added to a cluster at the same time, and so multiple cue balls can run concurrently. Experientially, concurrent cue balls, especially when they are of different speeds (e.g. two slow and one fast cue ball), create musical layering and complexity. A play block only controls the cue ball that it has launched.

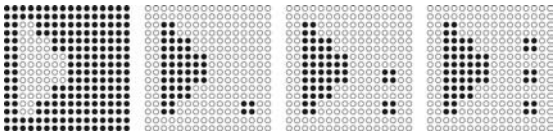

**Figure 10. Play Icons displayed on the 16*16 LED matrix**

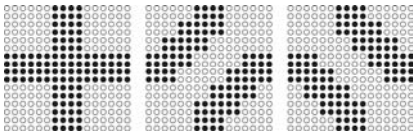

**Figure 11. The straight and corner function icons**

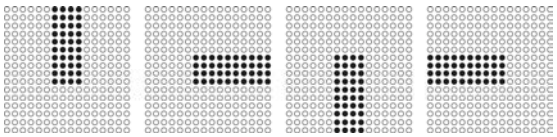

**Figure 12. The four rotated states of the gate icon**

### 3.2 Controlling the Path of a Sequence

Clicking a path block toggles between various path changing functions. The path changing functions determine which side of the block a cue ball will exit from relative to the side through which it entered. The path changing functions are:

1. The *straight* function, which bounces the cue ball straight along the given axis
2. The *corner* function, which bounces the cue ball at 90° relative to the given axis
3. The *gate* function, which jumps the cue ball in the direction indicated by a changing icon displayed on the LED surface, the graphic rotates 90° every time it is triggered

When a cue ball comes to the end of a route, i.e. it has nowhere to jump next, the cue ball loops back to the play block from which it originated. If a cue ball bounces onto a play block, its route is reflected 180°, returning in the opposite direction.

The corner function was often used to build large loops, but they could also be used as a means of isolating one area from another. For example in Figure 13, a cue ball launched from the top red play block would never reach the bottom two green blocks.

The gate function was designed to act as a type of counter, sending a cue ball in a given direction every fourth count. More often, it was used as a randomizer adding variation to a cue ball's route. When a large number of play blocks were gathered together and all set to the gate function, with several play blocks initiating several cue balls, the system constantly varied, never sequentially looping, much like the behavior of cellular automata. Although fascinating to watch and consider, this did not produce interesting sound. The output, though quantized, felt structure-less, and without flow or variation. It generally produced an un-engaging din when used with harmonic sounds (surprisingly), but was more palatable when we used rhythmic sounds.

### 3.3 Choosing a Sound

Milled into the top surface of a block's display is a circular groove. If a user places their forefinger into the groove and follows it rotationally, the block senses a dialing activity [Figure 13].

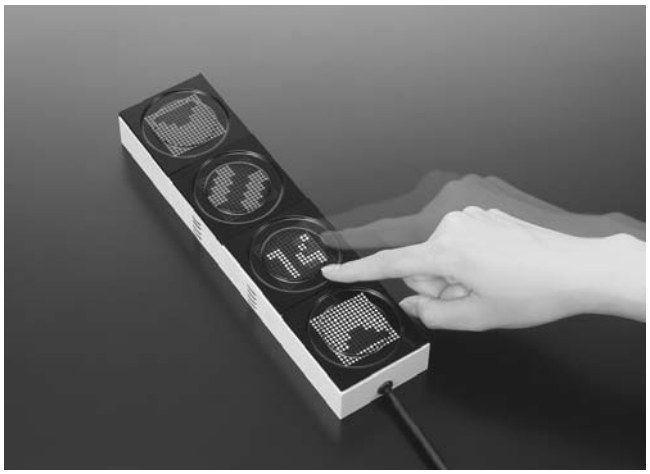

**Figure 13. Dialing a new sound**

Dialing changes the block's sound ID. As the user dials, a number representing the sound ID on the display counts up or down from 0 to 15. Dialing clockwise counts up, and dialing anticlockwise counts down. The numbers were grouped into three colors, so sound IDs 0-5 are displayed red, 6-10 are displayed orange, and 11-15 are displayed green. (More colors were possible, but we limited the number to three for easy readability)

After the user has finished dialing the chosen sound ID the block's display reverts back from the shown number to the functional icon. The icon retains the color of the sound ID.

For example, if we take a play block that has a sound ID of 1, the play icon shown on the block's display will be red. If a user then dials in a sound ID of 12, after the user has finished dialing the display reverts to the play icon, except that this time the icon is green.

The different color groups allow the user to approximately know visually which block has which sound ID. In our prototype, we mapped all the guitar and organ type sounds to the red group, vocal sounds to the orange group and percussive sounds to the green group (sound ID 0 is never mapped to a sound, it can be used as a musical pause by a user). The iconic use of color greatly assisted a users ability to keep track of the structures they had created, and which sound they had put where. This was not something that we had originally considered - our initial prototype only had red LEDs, which led to user confusion and frustration. The addition of the color groups allowed the users to play more freely, they no longer had to remember which sound was related to which ID, but instead could say "I want an orange sound here, and a green sound there".

The resulting sound that a block played was dependant on two variables:

1. The block's sound ID
2. The speed of the cue ball that triggered it.

This meant that a block doesn't just contain one sound, it actually contains three; one sound for a slow cue ball, one sound for a medium speed cue ball, and one sound for a fast cue ball. Consequentially, a single block can play one of 45 possible sounds from 15 sound IDs.

Having the combination of different cue ball speeds and sound IDs, in conjunction with the modular tangible interface, though apparently simple, allowed the user an enormous number of musical possibilities, in an intuitive, easy to use interface without any prior musical knowledge.

# 4. OVERVIEW OF THE BLOCK JAM SYSTEM

The blocks are connected to a *mother box* via the mother block. The mother box mediates serial data from the blocks to a PC. The PC is connected via MIDI to a sound module to render the output.

The software architecture on the PC comprised:

1. The topological model
2. The functional model
3. The mapping
4. The sequencer

As a block is connected to the cluster, it is activated. It sends a message to the PC telling the topological layer its unique ID and a number identifying which side (0-3) and the ID and connecting side of the adjoining block(s). From this information, a topological model of the blocks is deduced.

Attached to the topological model is a functional model. The functional model tracks all the users interactions and changes each block's state accordingly. A metronomic event triggered by the sequencer tells the functional model to send a list of output events to the mapping layer if there are one or more active cue balls. The sequencer finally sends the appropriate MIDI data to the sound module.

## 4.1 Mapping the Sound to the Sequencer

Every time a cue ball bounces from block to block, it triggers an event. The trigger event is then mapped to an array of MIDI events called a *part*.

The part is then passed to an ad hoc sequencer, which quantizes the MIDI events (according to a time stamp) and passes them to a MIDI sound module, which renders the resulting sound out put. A user experiences the sound coming from a block metaphorically. We use a part instead of playing a sample directly because it allows us a to do much more compositionally – after all, MIDI can control much more than a sampler.

If two or more cue balls trigger the same block at the same time then all of the resulting parts will be passed to the MIDI sequencer. If the cue balls are of different speeds, for example one fast and one slow cue ball, then the cue ball will render the two different parts (i.e. the block's sound fast part + slow part). If, on the other hand the cue balls are the same speed, there will be no additive effect.

## 4.2 Composing for Block Jam

Because we are using MIDI and a sequencer, music can be easily composed for the Block Jam system. During the Block Jam's development, we worked closely with musicians, in an effort to maintain parity between our system and how a modern musician might want to compose their music. It became apparent early in our prototype development that we needed to create a simple authoring structure that the musicians could use. The structure needed to cover two areas, authoring the possible 45 parts (time stamped arrays of MIDI messages), and determining the setup for the MIDI sound module.

After assessing many different sound modules, we chose the "Reason 2" virtual sound studio and software synthesizer because of its high quality output and its modular patching system. The modular patching system allowed the musicians to rapidly plug together a number of

virtual instruments and create the rendering context they required. We then created a grid that the musicians could follow in the Reason sequencer, with a space allocated for each of the 45 parts. A part being a time stamped MIDI array, is essentially a small sequence – so to export the parts to the Block Jam system a musician simply exported the entire MIDI song. Because of the grid, we knew where each part was located in the song and could easily parse the data. The same rendering context (Reason file) was used to output the music for Block Jam. This structure allowed the musicians to rapidly compose music for us, which meant that we could try a variety of mappings and types of sound.

Semantically the length of the sounds rendered from a part should have some parity to the speed of the sequence. In the case of our prototype, fast parts tended to last one beat, medium parts two beats long and slow parts one bar (4 beats in 4/4 time). This assists the user in connecting the resulting sound to the activity of the cue ball.

Of course, what is mapped and how long its duration is, is entirely up to the composer who is authoring the music for the system. It is also worth noting that a part does not even have to contain sound triggering (note On) MIDI data, it could just contain sound altering data, such as control change or pitch bend messages.

An obvious limitation to the Block Jam system (in terms of composition only) is that any sound can be played with any other sound at any give moment – which means that all the sounds should work together. The easiest way to achieve this is to keep the instruments distinct (timbre), and play all the notes in the same key.

A way around this limitation would be to create an algorithmic layer (as previously mentioned) that could automatically handle global events such as key changing. Realistically, this would have to be very carefully mapped to Block Jam, giving greater compositional variety without the user losing their sense of control.

## 5. USER RESPONSE

We were hoping to elicit an engaging or thought provoking response from users using Block Jam; therefore, we relied on anecdotal information and our own observations of participatory demonstrations, to assess usability and understanding of the system. Participatory demonstrations were made to members of the Sony community and outside visitors to our laboratory. Block Jam was also demonstrated at SIGGRAPH2002 in the Emerging Technologies Section [[15]]. Response from users was overwhelmingly positive, satisfying our aim to design an engaging system.

Users required a minimum of instruction, and would usually spend 3 or 4 minutes "working it out". Users tended to have the greatest initial difficulty with the most popular feature – dialing. A conflict can occur because the acrylic surface is used for dialing and clicking, so when a user pushes down while dialing (thus clicking) at the same time, they're also inadvertently toggling a blocks function. This could be easily fixed by making the round area in the center of the surface move independently of the surrounding milled groove. Once a user understood the conflict, it no longer interfered with their interaction.

Another observed confusion arose when users tried to select different speeds of cue ball – the play icon (indicating the speed) is displayed after the click interaction has been displayed – so user feedback is after the fact. One user suggested that we should animate the play block icon while it is being clicked – so the change in the speed indicator can be watched as it's being held down.

Users often built similar structures, the abstract complexity of the structure reflecting their understanding of the how the Block Jam system functioned. The first structure built was usually a single row of blocks. Then a user would discover the corner function; this was usually followed by the question "can I build a big loop?" From the "big loop" structure users tended to build a series of 3 rows of different lengths with a play block at each end creating a system for easily observing the timing in relation to the different cue ball speeds. After this point, it was very much up to the individual – some wanting to build visual patterns, massive structures, organized structures, and playful random structures. Once a user started building, it was quite hard to make them stop. We had initially been a little concerned about how quickly a user might get bored of the system – how wrong we were! Users loved to play. Some wanted to use it as a tool for performance, others wanted to collaborate – "It would be great for parties" was a typical response. Everyone wanted to take them home.

Block Jam is not a musical instrument; it is an alternative means of controlling a sequencer. It has no means of continuous control or gesture, and all interactions have a musical latency inherent to quantized sequencers. Conveying expression on such a system might be construed difficult if not impossible. However, users were able to control the musical output expressively through structure and timing, from gentle harmonies to complex beats and rising crescendos.

## 6. WE JAM

In addition to promoting face-to-face collaboration by using a tangible interface, we hoped to promote and explore remote collaboration through a network. We extended the Block Jam prototype by adding a second mother block and PC, to create two nodes and named the new prototype *We Jam*. The nodes were interconnected via MIDI as a means of simulating a real-time network. Though pertinent, we decided to avoid issues of latency normally associated with networks and decided to focus on the issues pertaining to the interaction alone.

A key element in our vision was the notion that different users within a framework interact using different applications – creating a functionally *asymmetric* dynamic. We realized that we had an opportunity to emulate this within We Jam, by allocating each node a different sound set. The first node was allocated a rhythmic sound set running at a speed of 120 beats per minute (bpm). The second was allocated a more harmonic sound set running at a speed of 60 bpm. So when interacted with individually, the nodes provided the users with very contrasting musical experiences, one fast and edgy, the other slow and melodic.

The sound sets were composed so that when the users played together they would produce a pleasing result. We found that inexperienced users, at first tended to play erratically trying to discern which sound belonged to whom, but after a few minutes would start to play cooperatively. Experienced users would play cooperatively, improvisationally reacting to the other user's actions.

## 7. CONCLUSION

In this paper, we have presented the design and implementation Block Jam. We have demonstrated that the Block Jam system succeeds as both a tangible interface and musical application – eliciting a positive experience and provoking a collaborative response in users.

By spatially-multiplexing a sequencer through the use of a tangible interface have created novel interactive

possibilities. The notion of collaboration was further extended by the addition of a network to create We Jam – our latest prototype to date.

We hope to continue this work; exploring alternative types of tangible interface, different types of application (looking at gesture in particular), and further explore the use of a network. In our next project we hope to create a collaborative interactive music system for remote users using hand held devices.

We are also interested in applications for the Block Jam tangible interface other than music. After all, nearly all media is sequential, so the system could easily be adapted for visual applications, or creating dynamic narrative structures.

## 8. ACKNOWLEDGEMENTS

We would like to thank the Sony CSL Interaction Lab members, particularly Dr. Jun Rekimoto and Ivan Poupyrev for their help and support. We would also like to thank members of the Sony Design Center, particularly Kei Tostuka and Yutaka Hasegawa. We are also indebted to Ryota Kuwabuko - who designed the blocks' integrated circuit and display mechanisms, and to Kenjiro Mastuo who composed the music used for testing the system and the music for the SIGGRAH demo.

## 9. REFERENCES

[1] Fitzmaurice, GW. (1995) Bricks: Laying the Foundations for Graspable User Interfaces. CHI95 Proceedings ACM Press.

[2] Umberto, E. Translated by Cancogni, A. (1989) The Open Work. Harvard University Press.

[3] Mathews, M.V & Moore, F.R. (1970) Groove – A Program to Compose, Store, and Edit Functions of Time. Communications of the ACM, vol. 13, no.12, December 1970.

[4] Pachet, F. (2002) Playing with Virtual Musicians: the Continuator in Practice. IEEE Multimedia, 2002.

[5] http://www.media.mit.edu/hyperins/

[6] http://www.neurosmith.com/

[7] Resnick, M. (1993). Behavior Construction Kits. Communications of the ACM, vol. 36, no. 7, pp. 64-71, July 1993.

[8] Poupyrev, I. (2000) Augmented Groove: Collaborative Jamming in Augmented Reality. SIGGRAPH 2000 Conference Abstracts and Applications, ACM Press.

[9] Iwai, T. (1999) Composition on the Table. SIGGRAPH'99, Electronic art and animation catalog July 1999.

[10] Ullmer, B et al. (1998) MediaBlocks: Physical Containers, Transports, and Controls for Online Media. SIGGRAPH'98, Conference Proceedings 1998.

[11] Lego Mindstorms, http://www.legomindstorms.com

[12] Rekimoto, J. et al (2001) Datatiles: A Modular Platform for Mixed Physical and Graphical Interactions. CHI2001, Conference Proceedings 2001.

[13] Frazer, J.H. (1982) Three Dimensional Input Devices, Computer/Graphics in the Building Process, March 1982

[14] M.G. Gorbet, M. Orth, and Hiroshii Ishii. Triangles: Tangible Interface for manipulation and exploration of digital information topography. CHI'98 Proceedings, pages 49-56.

[15] Newton-Dunn, Nakano, and Gibson. Block Jam. SIGGRAPH2002, Conference Abstracts and Applications, page 67. ACM Press

[16] http://www.u-ga.com/rez/

[17] http://www.us.playstation.com/games/SCUS-97167/parappa1024_win.html

[18] http://www.hi-res.net/