

INT.LIB – A GRAPHICAL PRESET INTERPOLATOR FOR MAX MSP

Oliver Larkin

ICSRiM* – Interdisciplinary Centre for Scientific Research in Music,
School of Music, The University of Leeds

*Work on this project was started during an internship at INA-GRM in 2005, and was continued at Middlesex University Sonic Arts Department in 2006.

ABSTRACT

int.lib (interpolation library) is an extension for Max/MSP that facilitates the control of multiple parameters using an intuitive graphical interface. The power of int.lib lies in its modular, multi-layered approach to preset interpolation, real time visual feedback, and integration within the Max environment.

This article discusses existing examples of the preset interpolator concept and describes design considerations, technical implementation and features of int.lib.

1. INTRODUCTION

When performing with highly parameterised software, musicians are often limited by the amount of control offered by the computer keyboard and mouse. Even using a hardware controller, it is impractical to adjust large numbers of parameters within a short time frame. Likewise, when composing, it can be difficult to create transitions between multiple parameter values, involving automation of each parameter individually. One approach that can help in these situations is “preset interpolation”. Presets (also known as snapshots, programs or patches) are configurations of multiple parameters (and should be familiar to all users of audiovisual processing and synthesis software). Interpolating between presets facilitates non-discreet transitions and the discovery of new “hybrid” settings that blend the characteristics of two or more existing presets.

Preset interpolation systems have been available in computer music applications for some time [2] and several different kinds of system have been realised (both graphical and non-graphical). Some systems offer interpolation between a low number of presets simultaneously (e.g. [5]) and some allow multi- or n-dimensional preset interpolation. Various approaches have been taken to graphical multi-dimensional interpolation, which differ in their visual representation of presets and in their underlying algorithms [1][2][3][4].

Max/MSP features its own preset interpolator in the form of the *patrstorage* object (introduced in Max 4.5), which allows one-dimensional and multi-dimensional interpolation via the *recall* and *recallmulti* messages respectively. In addition, several graphical multi-dimensional systems have been implemented in third party libraries [3][6][7][8].

int.lib uses a gravitational approach to multi-dimensional preset interpolation and is inspired by the

INTERPOL interface of the SYTER (Système Temps Réel) [1] which was developed at INA-GRM (Institut National de l'Audiovisuel - Groupe de Recherches Musicales) in the 1980s. This method allows the user to interpolate between multiple presets by navigating a two dimensional environment in which presets are represented by balls. The size of each ball and its proximity to the cursor affects the weight of the associated preset in the interpolation. This approach was chosen for its visual simplicity and applicability to the design goals of the project.

2. DESIGN GOALS

int.lib has been developed to turn the preset interpolator concept into a complete performance-ready interface for Max MSP. The design goals were as follows:

- to allow the control of multiple parameter sets independently from one encapsulated interface
- to abstract the user interface from the max patch
- to facilitate rapid layout of the interpolation space
- to be fast enough to support interpolation of many presets featuring many parameters
- to be easy to understand and use

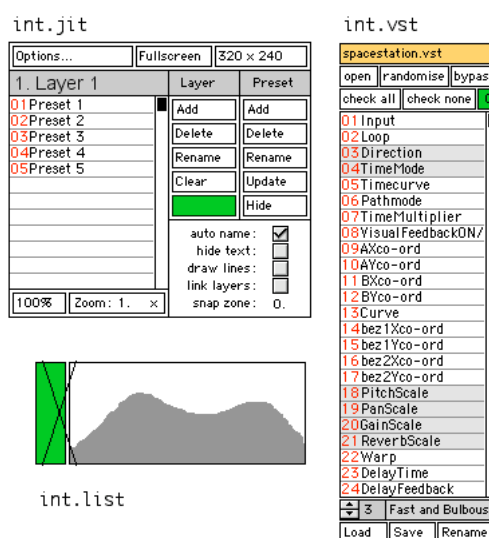


Figure 1. The three main objects. Here *int.list* is shown connected to Max's *multislider* object.

3. THE INT.LIB OBJECTS

int.lib consists of three Max patches which can be loaded either as “abstractions” or by using the *bpatcher* object (see Figure 1)

int.jit is the control centre, which deals with preset management. This object also opens an OpenGL window called the “Interpolation Space” (see Figure 2). This is the main graphical interface, which is used to control interpolation. The interpolation space can be maximised so that it occupies the full screen.

int.list allows lists of normalised int or float values to be used as presets. An argument to *int.list* specifies the length of the list.

int.vst is a wrapper for MSP’s *vst~* object which allows the user to load a VST plugin and select certain parameters for interpolation. Non-continuous parameters can be removed so as to avoid abrupt changes during interpolation. *int.vst* also provides a function to randomise the selected parameters. The object supports VST programs and saving and loading of the standard *fxp* and *fxb* preset formats.

Beneath the surface, int.lib is implemented in JavaScript. OpenGL graphics are programmed using Jitter objects, which are instantiated and controlled within the JavaScript code. Since int.lib has no dependencies on custom externals the library is compatible with Max and Jitter running on Windows and OSX.

4. METHOD

When navigating around the interpolation space, the distance from each ball to the cursor is calculated. A coefficient is derived for each preset based on the distance and the radius of the associated ball. This coefficient determines the preset’s weight in the interpolation. Finally, each parameter in the preset is multiplied by the coefficient and the values for each preset are summed.

Compared to some other multi dimensional preset interpolators, this method is computationally expensive since many calculations are performed on every mouse movement. In practice performance has proved acceptable, even with a large number of presets and parameters.

The number of layers and presets is that can be used in int.lib is limited only by the computer’s processing power.

5. FEATURES

5.1. Layers

In int.lib multiple interpolation layers can be presented simultaneously in the interpolation space. For each layer there is a crosshair, which indicates the current point of interpolation for that layer (see Figure 2.). Optionally the interpolation points may be linked so that all layers are controlled simultaneously.

Instances of *int.list* or *int.vst* are given an argument when they are initialised which associates them with a particular layer. Each layer is colour-coded in order to quickly identify which presets and objects are linked to which layer. Layers and their presets may be named and labeled in the interpolation space.

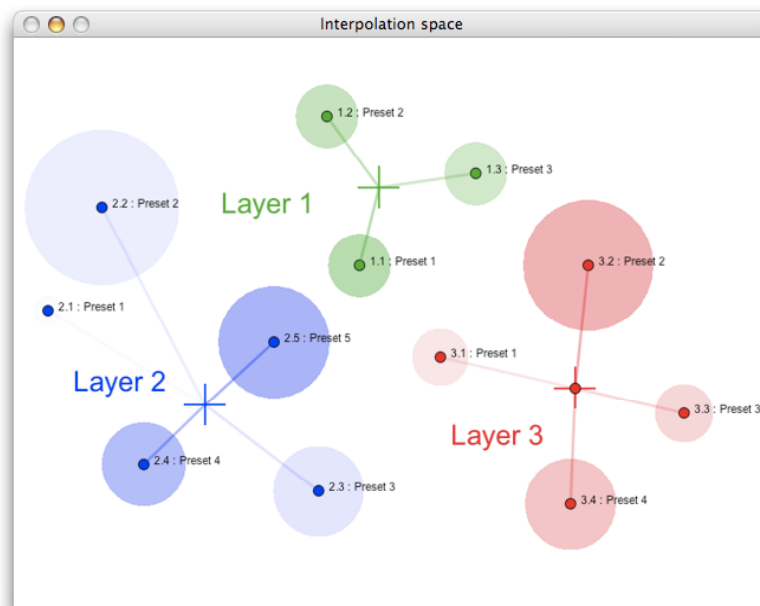


Figure 2. The Interpolation space showing three layers. The active layer’s crosshair is highlighted.

5.2. Magnification

Using multiple layers can result in many presets being onscreen at one time. With the mouse wheel or key commands one can zoom in on a specific area of the interpolation space and focus solely on the active layer or on certain presets. Further keyboard commands move the camera between layers. Lines may be drawn from the interpolation point to each preset in the layer (see Figure 2). This is useful if one or more presets are off screen. When zoomed in, the camera follows the cursor as the user navigates around the interpolation space.

5.3. Visual Feedback

Existing graphical preset interpolation systems present the user with a static environment. As they navigate to a preset, only the results (e.g. the sound output) and the position of the cursor give an indication of the balance of presets in the interpolation. If the presets are similar the auditory indication may be minimal, likewise if there are a lot of presets near the cursor it may be difficult to see which preset has the strongest weighting. *int.lib* is designed to be responsive so that the user is aware of which presets are being included in the interpolation. This is achieved by linking the weight co-efficient of each preset to the transparency of the associated ball - if a preset has zero weight, only a “handle” at the centre of the ball is displayed, if the preset has full weight the ball is a block of solid colour (see Figure 3).

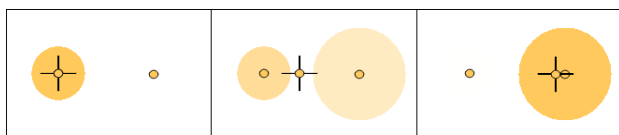


Figure 3. Visual feedback

5.4. Layout Commands

Preset layout functionality is achieved with the mouse buttons and keyboard shortcuts, allowing the user to quickly design and manipulate their interface. Geometric arrangements of presets can be created very easily and preset positions and sizes may be randomised. During interpolation, if a satisfying combination of presets is found, one can press a key to create a new “hybrid preset” based on the current values of parameters.

5.5. Snap Zone

When controlling parameters such as frequency, tiny movements in the interpolation space can result in changes that are very obvious to the ears – if one tries to navigate to a certain preset, the output will only be correct when the interpolation position matches the exact co-ordinates of that preset. To solve this problem *int.lib* features a variable called the “snap zone” which is an invisible circle around each preset. If one navigates inside this circle, the interpolation position will snap to

the co-ordinates of the preset and the interpolation engine will ignore all other presets in its calculation.

5.6. Automation

int.lib provides two methods of sequencing / automating transitions between presets. The first of these allows freehand gestures to be recorded, played back and looped. The second method uses a point-and-click Break Point Function generator that allows the user to add a series of points on the interpolation space, which are joined to form a trajectory. Each line segment is listed in the BPF editor with its duration in seconds and a signed curve factor to adjust the velocity as the interpolation point traces the trajectory of the segment. Positive curve factors result in a pseudo-exponential velocity curve and negative factors result in an inverse pseudo-exponential curve.

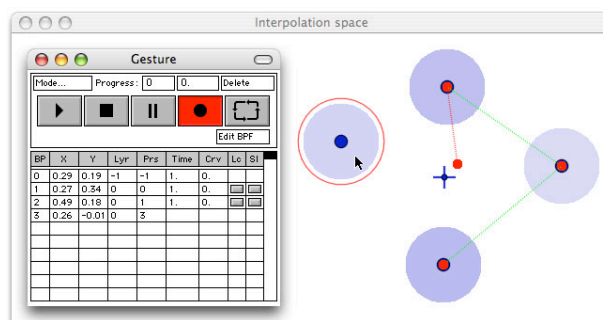


Figure 4. Break Point Function and editor.

When defining the break point function, if one moves their mouse over a preset, that preset is highlighted. When created, the new breakpoint is flagged as “selective” and is snapped to the position of the highlighted preset. Any lines drawn between two breakpoints that are flagged “selective” are coloured green. During playback, only the two associated presets will be used in the interpolation calculation. This enables precise transitions from one preset to another over a specified time period. The user can decide whether transitions are selective or not, which makes it easy to switch between one-dimensional and multi dimensional preset interpolation.

5.7. Pattr Integration

int.lib reports all preset, plugin and layout data to Max’s *pattrstorage* object, which allows multiple setups to be stored and recalled using xml.

5.8. Input

Finally it is worth noting that the interpolation positions of different layers may be controlled by sending messages to the *int.jit* object. This means that *int.lib* can be driven by any 2D input controller (not just the mouse). The system can be used as a rapidly configurable two-to-many mapping layer for gestural interfaces.

6. CONCLUSION AND FUTURE WORK

int.lib has been released [9] under the GNU LGPL and has received a positive response from users. The library is robust, easy to use, and provides Max users with a powerful performance controller, sound design and composition tool. Areas of interest for future versions include improved integration and compatibility with the *patr* family of objects and interface speed improvements.

7. ACKNOWLEDGEMENTS

The author would like to thank Daniel Terruggi, François Donato, Phillipe Dao and Emmanuel Favreau at INA-GRM, Martin Robinson at Middlesex University Sonic Arts Department, Kia Ng and everyone at ICSRiM.

8. REFERENCES

- [1] Allouis, J-F. and Bernier, J-Y. "The SYTER project: Sound processor design and software overview" *Proceedings of the International Computer Music Conference*, Venice, Italy 1982.
- [2] Bencina, R. "The Metasurface – Applying Natural Neighbor Interpolation to Two to Many Mapping". *Proceedings of the International Conference on New Interfaces for Musical Expression*, Vancouver, Canada, 2005.
- [3] Momeni, A. and Wessel, D. "Characterizing and Controlling Musical Material Intuitively with Geometric Models". *Proceedings of the International Conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003.
- [4] Spain, M. Polfreman, R. "Interpolator: a two-dimensional graphical interpolation system for the simultaneous control of digital signal processing parameters". *Organised Sound* Vol. 6, no.2 pages 147-152. 2001
- [5] Favreau, F. GRM Tools.
<http://www.grmtools.org>
- [6] Lefèvre, A. VECT library.
<http://www.adlef.com/>
- [7] DeTar, C. Colorblobs library.
<http://theendmusic.org/>
- [8] Place, T. Tap Tools library.
<http://www.electro-tap.com>
- [9] Larkin, O. int.lib library.
<http://www.olilarkin.co.uk>