Pd/GEM & Metaphorical Networks
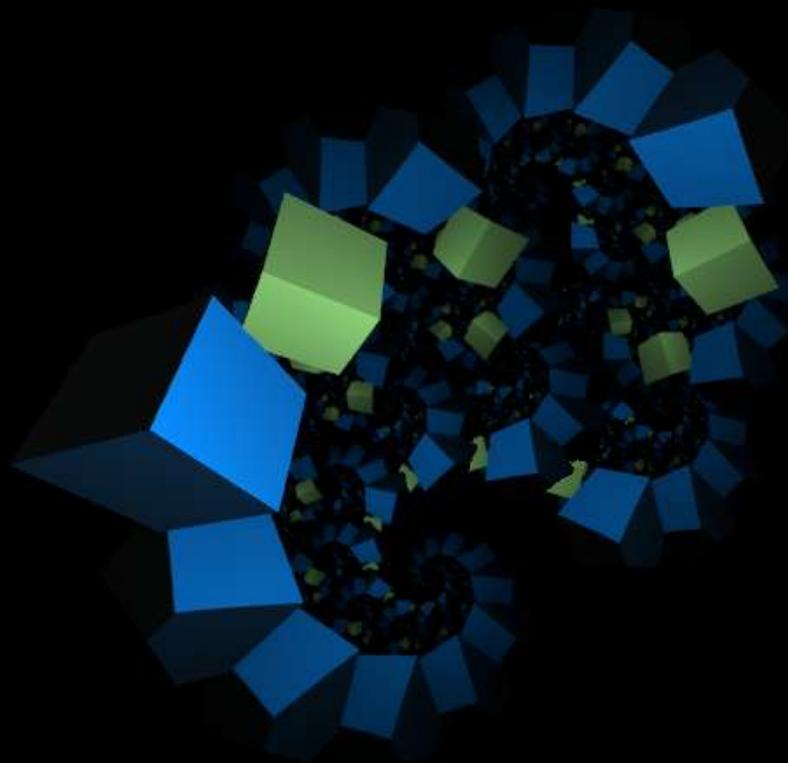
Taught by Ben Bogart January 15th 2005
@ Interaccess Electronic Media Arts Centre
http://www.ekran.org/pd

Workshop Content

- Examples of projects created with pd/Gem

- What is the technology?

  – What is Pure-Data?

  – What is Open Source software?

  – What is a Data-Flow language?

- Introduction to Pd with interactive lecture

- Introduction to GEM with interactive lecture

- Metaphor and Data-Flow (metaphorical networks)

- Advanced Pd topics

- Free exploration and project for Open Lab

Example Patches

There are many example patches floating around. Below are a few patches that can be downloaded. I've also included a few examples from my personal collection.

- For download:

    - Video tracking (à la softVNS)

    - Gem-image-grid (image distortion & UI)

- Personal explorations:

    - OpenGL feedback

    - Physical modelling and animated text

    - Sneak preview of pixelTANGO

What is the Technology?

- Pure-Data (Pd) is a Data-Flow programming language.

- Pure-Data is Open Source – it is developed by a community of users and is meant to be a gift to the community. It is not a profit-seeking venture.

- A Data-Flow language is programmed by controlling the behaviour of a flow of data. This flow is not unlike the flow of water in a plumbing system.

- Data-Flow languages tend towards having three types of operations:
  - Inputs (sources of data)
  - Filters (changes the way the data flows)
  - Outputs (destinations of data)

Pd/Gem Lecture

At this point we are going to do a guided lecture of Pd/Gem. You can download the Pd lecture at:

http://www.ekran.org/pd/PD-Lecture.tgz (Linux/OSX)

http://www.ekran.org/pd/PD-Lecture.zip (Windows)

http://www.ekran.org/pd/PD-Lecture.pdf (Non-Interactive)

Metaphor and Data-Flow

- A definition of metaphor:

  - Mapping of the conceptual domain of one word, phrase or sentence (the source domain) to another (the target domain). (Lakoff & Turner)

- Working example of a metaphor:

  - As an eye opens the sun rises

  - An opening eye is a sun rising

- The conceptual domain of a "sun rising" (awakening, birth, beginning) is applied to an "opening eye"

Metaphor Modelling

- In a computer system, the relationship between these terms can be modelled:

  - The "sun rising" describes a process from the sun being set to the sun having risen. We can symbolize the start and end of this process in a computer. For example, the start of the process can be represented as s0 (for set) and the end of the process as s1 (for risen).

  - The "eye opening" is also a process where e0 (for closed) is the start and e1 (for open) is the end.

  - We can attach these two scales so that when the eye is closed (e0), the sun is set (s0). When the eye is open (e1), the sun has risen (s1).

Modelling with Data-Flow

- In a Data-Flow language, these two (numeric) scales can be visually represented in such a way that the metaphor becomes readable for a human:

    - The "eye opening" is the target domain or output. Its scale from e0 to e1 could be represented by a series of video frames showing an eye opening.

    - The "sun rising" is the source domain or input. Its scale from s0 to s1 could be literally attached to a physical sense. This scale could be represented by the values returned by a LDR (light dependant resistor).

The Metaphor Becomes a Network

- This model is a programmatic representation of the metaphor.

- As the brightness in the room increases, the resistance of the LDR changes. This range of resistance becomes the source domain. The source domain is mapped to the target domain, which is then represented as a video image of an eye opening.

- This is the creation of a metaphorical network – a group of symbols that are mathematically connected to create metaphorical relationships.

Why Data-Flow?

- Data-Flow languages are ideal to create metaphorical networks because by definition they deal with networks of symbolic items.

- Data-Flow languages are highly focused on working with scales of numbers (such as audio signals).

- The most valuable skills for creating such networks (after creativity) are linear algebra and calculus.

Advanced Pd/GEM (1)

- Cord-less communication in Pd:

  - 'send' and 'receive'

- Gem grouping and layering:

  - 'separator', render-order and 'trigger'

- Advanced message usage:

  - $ in message boxes: variables

- Abstractions vs Subpatches:

  - $ in object boxes: arguments

  - Scoping 'send' and 'receive'

Advanced Pd/GEM (2)

- Graph on parent (GOP) abstractions:

  - Modular UI

- Network communication: (TCP/IP & UDP)

  - Netsend and netreceive

  - Open Sound Control (OSC)

- Internal Pd messages:

  - Dynamic patch creation, patch to build a patch

- Pd command line options:

  - How to use Pd as a component in a system

Assignment for Open Lab

Your assignment (which you should develop over the remainder of this class and during next week's Open Lab) is to create your own metaphorical network.

At this stage, we won't worry about using actual sensors but will use an automated (or interactive) number scale to represent your source domain. Be as creative with your symbols and metaphors as you wish. There may be more than two terms and the system need not deal with the sensed environment at all.