Rules-Based Pattern Simulator

Kentyn Grey Reynolds
July 2007

Visual Synthesizer is an animation generator that grows visual patterns based upon intuitive interaction with an artist. Visual Synthesizer represents the world as an environment filled with objects whose behavior is constantly evolving according to a dynamic set of rules. An object is constructed from a graphic shape with a set of attributes that capture the object's response to current conditions. An object's behavior is displayed through the mapping of the object's attributes to visual elements. The artist controls the object's behavioral patterns and attribute mappings through the use of MIDI machine controls, and captures the evolution of the system as video animation.

In Visual Synthesizer, the environment contains the attributes of space, time, light, and other manifestations of energy. Inside of the environment is a lattice that defines a section of space and subdivides this section into discrete locations. The lattice does not need to be a fixed structure, but can evolve with time. Shapes populate and align with the spatial structure defined by the lattice. Objects are composed of shapes plus a set of attributes that hold information about the state of the object's response to its surroundings. Observable patterns emerge from the lattice as objects organize into communities of related attributes.

The artist observes the evolution of the environment by defining a start and end point for a period of time. The selected time period may be subdivided into a set of evenly spaced pulses. Each object is aware of the environment's pulse and responds by evolving its state to the next discrete moment in time. Each discrete moment in time is captured as a single frame of animation. The sequence of animation frames may be viewed as a video recording that captures the fluid unfolding of the system's behavior.

The environment appears to be at rest when all objects contain identical attributes. In order for a growth pattern to begin, an event must stimulate one or more objects into a new state. Multiple events or patterns of events may move through the lattice and sequentially force waves of state transitions in objects. Forces such as wind may be represented as energy events that propagate into the lattice with a specified density, path, and speed. Energy events propagated into the lattice are absorbed by objects and in turn contribute to their evolution. In Visual Synthesizer, the artist interacts with the real-time propagation of events into the environment.

Behind the interface, that the artist interacts with, are the architectural details of a rules-based pattern simulator. At the core of a pattern simulator is a rules engine that executes a set of rules for each object in the system at each pulse of the clock. The rules engine takes a snapshot of the surrounding conditions and helps evolve the object's state to the next moment in time based on the attributes contained in the snapshot. Conditions in the environment that surrounds a set of objects are always changing and the rules engine must take this into account and evolve the rules according to these changes.

Visual Synthesizer communicates with other systems through the use of video and audio. Photos, movies, graphics meshes, or MIDI recordings are drawn into the system and converted into waves of events that force changes in the environment. The objects in the system respond to these waves of events, and in turn the system captures the objects' response through the use of video animation.

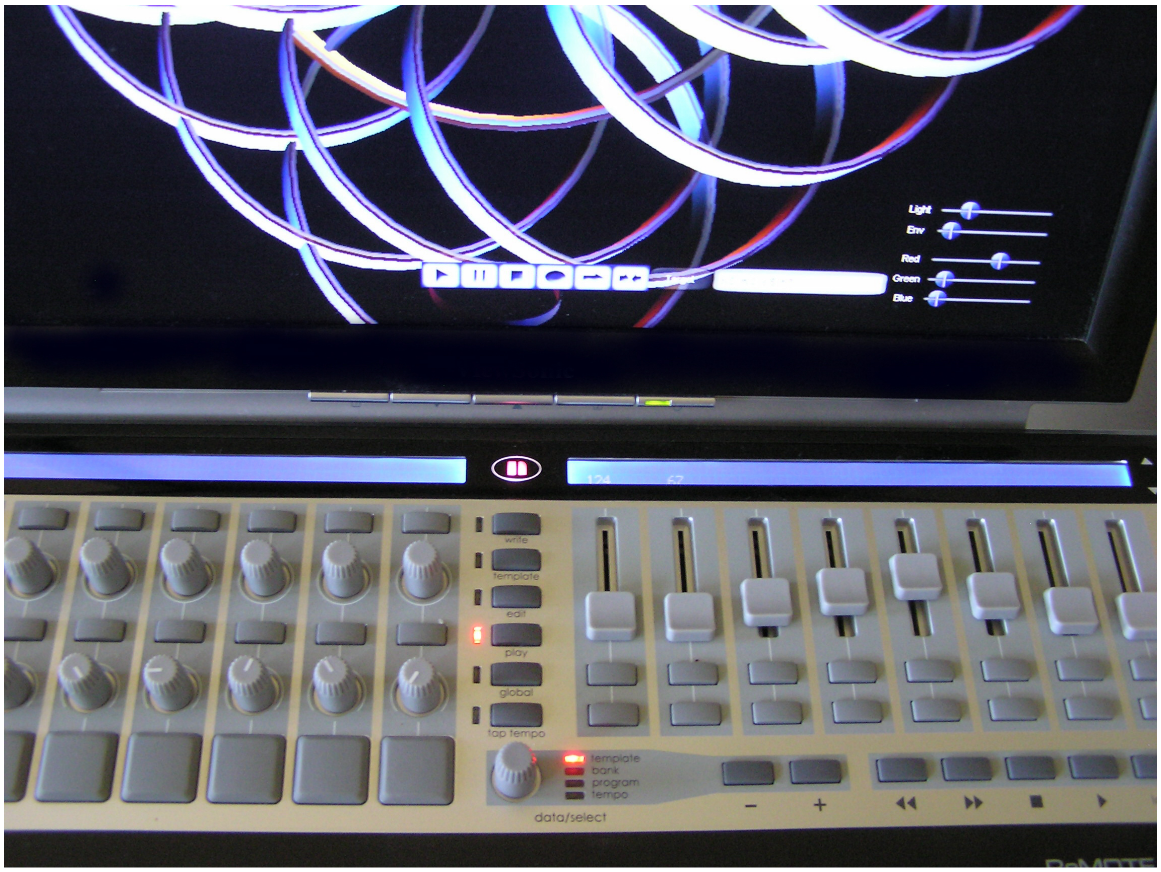


Fig 1.Visual Synthesizer's MIDI Controllers

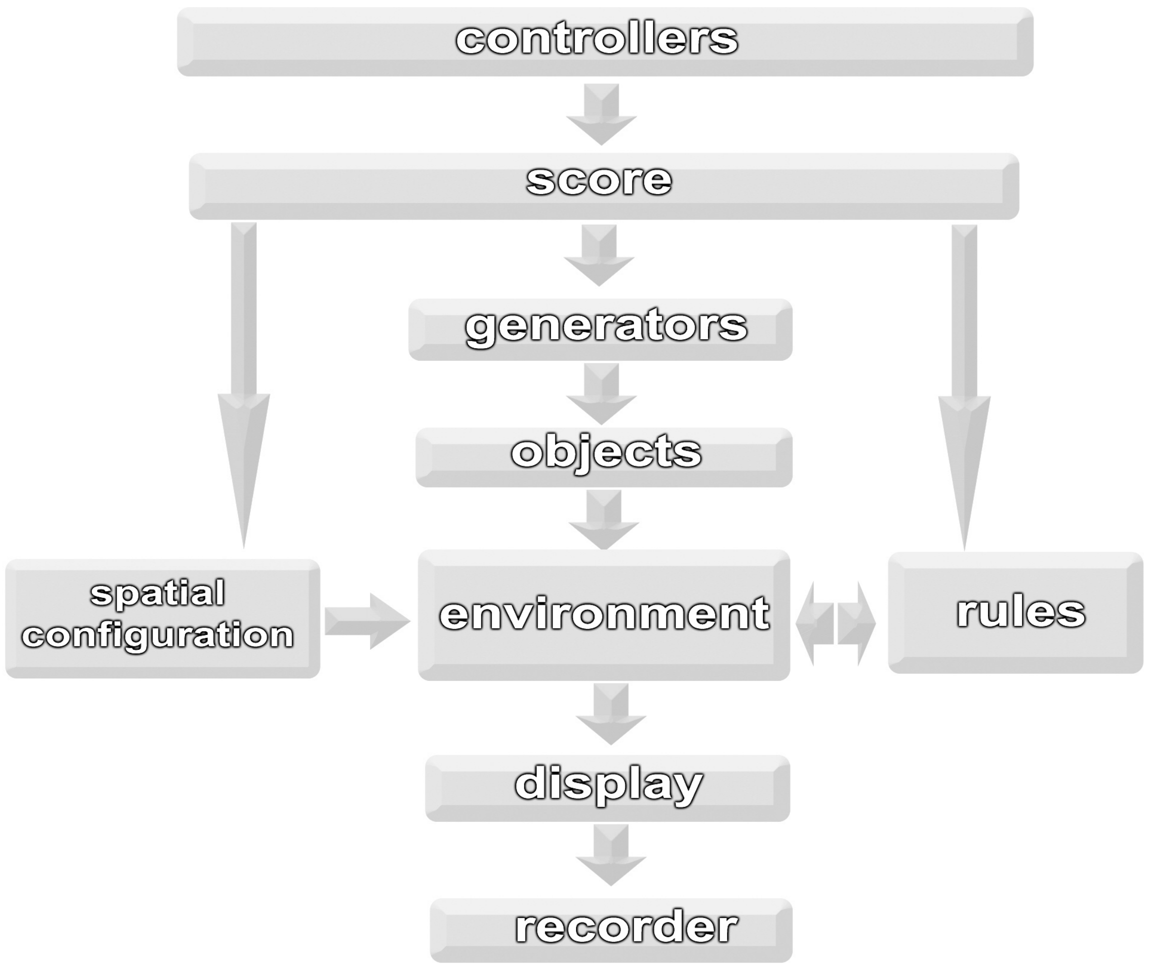


Fig 2. Visual Synthesizer Architecture

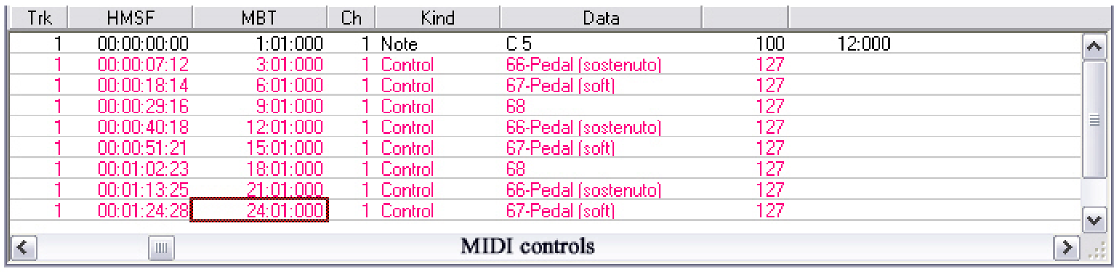
The diagram (above) shows the architecture of Visual Synthesizer. The ‘environment,’ holds arrays of objects according to the specification contained in the ‘spatial configuration’. ‘Controllers’ are responsible for the master clock and interaction with the observer. The ‘score’ records the events produced by the controllers in real-time and then allows the score to be replayed. The ‘generators’ map events from the score into patterns of events that are propagated into the system. The ‘rules’ allow an object to communicate with other objects in the environment and to unfold to the next moment in time. Each object in the system defines itself through its attributes and is aware of both its past and present state. The growth and evolution of objects inside of the environment is mapped for ‘display’ and then recorded to HD video.

Visual Synthesizer has evolved over many years of development through a process of constant simplification of the architectural model. The guiding principle behind the development of this architecture is that complexity indicates a flaw in some aspect of the fundamental design and refactoring is a better solution then debugging. As a result of this design process, the scope and functionality of Visual Synthesizer is quickly revealed through a simple examination of the nine system components listed above.

Inside of the environment is a lattice that occupies a section of space. The spatial configuration component instructs the lattice how to subdivide this section of space into discrete locations and tells the lattice how objects may occupy this space. The spatial configuration component determines if the lattice is modeled as tessellations, compound structures, or complex shape-based structures. The structure of the lattice determines what constitutes an object's immediate neighbors.

Objects are shape-based entities that occupy the lattice and align with its spatial definition. A simple example of the relationship between and object and the lattice is how a compound lattice structure (based upon the stacking of a cube, tetrahedron, and hexagonal prism) will require that only a cube shape occupy the location in space declared as a cubic. Objects are the system's 'actors' who follow instructions regarding how to behave.

The dialog between a set of objects, the environment, and external systems is managed by the system's generators. The generators map streams of input into events that are propagated to a set of objects. Examples of external streams of input are: video, MIDI, photographs, and graphic meshes. In addition to mapping external streams of input into events, the generators can generate patterns of events that range from single pixels to complex meshes. The propagation of events to objects may be structured to move at a given rate and follow a specified path through the lattice.



Trk	HMSF	MBT	Ch	Kind	Data		
1	00:00:00:00	1:01:000	1	Note	C 5	100	12:000
1	00:00:07:12	3:01:000	1	Control	66-Pedal (sostenuto)	127	
1	00:00:18:14	6:01:000	1	Control	67-Pedal (soft)	127	
1	00:00:29:16	9:01:000	1	Control	68	127	
1	00:00:40:18	12:01:000	1	Control	66-Pedal (sostenuto)	127	
1	00:00:51:21	15:01:000	1	Control	67-Pedal (soft)	127	
1	00:01:02:23	18:01:000	1	Control	68	127	
1	00:01:13:25	21:01:000	1	Control	66-Pedal (sostenuto)	127	
1	00:01:24:28	24:01:000	1	Control	67-Pedal (soft)	127	

MIDI controls

xml MIDI map

```
<grObject id ="0">GRAPH_RULES</grObject>
<grCat id ="0">COMPARE_VALUE</grCat>
<grSub id ="0">1</grSub>
<grProp id ="0">
  <x>GRAPHICS_X</x>
  <y>GRAPHICS_Y</y>
  <z>GRAPHICS_Z</z>
</grProp>
<grStart id ="0">
  <x>2.0</x>
  <y>5.0</y>
  <z>4.0</z>
</grStart>
<grEnd id ="0">
  <x>2.0</x>
  <y>5.0</y>
  <z>4.0</z>
</grEnd>
```

xml Rule

```
<neighborFace id ="0">
    <direction id ="0">N_ABOVE</direction>
  </neighborFace>
  <neighborFace id ="1">
    <direction id ="0">N_LEFT</direction>
  </neighborFace>
  <neighborFace id ="2">
    <direction id ="0">N_RIGHT</direction>
  </neighborFace>
  <neighborFace id ="3">
    <direction id ="0">N_BELOW</direction>
  </neighborFace>
  <calcOperation id ="0">RULE_COUNT</calcOperatio
  <compOperation id ="0">RULE_EQUAL</compOperatio
  <compareVal id ="0">
    <x>4.00</x>
    <y>4.00</y>
    <z>4.00</z>
  </compareVal>
```

Fig 3. MIDI Events and XML Mapping Instructions

At each pulse of the system clock, the generators propagate events to the objects, then the rules read the attributes of an object and a defined set of neighbors, and finally the rules instruct the object how to evolve from its current state to the next state. The structure of a rule may be as simple as averaging or counting attributes from multiple neighbors and then performing a stated action based on a logical decision. These simple rules can produce interesting results, but somehow they do not capture the nature of a dynamically evolving environment. In order to capture the nuances of a dynamically evolving environment, rules need to modify their processing, comparisons, and actions based upon a real-time conversation with the surrounding environment. Visual Synthesizer controls the configuration of a rule set in real-time by translating environmental changes into new instructions for the rule set.

Many times a rule may dramatically change the attributes of an object in a single pulse. These dramatic changes appear in the animation sequence as an abrupt change of state and can be visually disturbing. From a system perspective, these abrupt changes are a byproduct of the quantization of time and space into discrete steps and locations. Visual Synthesizer smoothes out these steps by viewing an object's new state as a target that will be reached in a given number of sub-pulses of the system clock. A good example of this behavior is the multiplication of an object into a matrix of new objects where the target is the new matrix but each singular object needs to slowly grow into its next state. Visual Synthesizer provides five distinct methods of inter-frame growth: size, transparency, color, speed, and location. The inter-rule interpolator subdivides the range between two states into smaller steps so that growth appears to be continuous.

Once the environment, objects, rules, and time period have been defined, it is time to turn the master clock on and set the generators in motion. Visual Synthesizer responds to the passing of time in two distinct ways. The first method is best described as real-time, in which the rules execute the best they can and are allowed to drop processing as needed in order to keep up with the clock. In the second method, the rules receive the clock information, stop the clock, compute the frame, and then re-enable the clock to go to the next frame. The advantage of the first method is it allows the system to be used as a real-time performance device. The advantage of the second method is that each frame is allocated all the time it needs to finish processing, and once the computation of the frame is complete it may be recorded to any HD format.

Inside of the environment are all the elements that support the filming of the scene: movable lights, cameras, and background images. It is not enough to color an object and turn a couple of lights on… instead the object's light absorption, reflection, subsurface scattering, and interactions with reflections and shadows from neighboring objects must be taken into account. One of the best models for simulating this depth of light interaction is Precomputed Radiance Transfer (PRT), which computes the illumination of a point as a combination of incident irradiance. A rule may simultaneously build an image while controlling the lighting effects applied to the image.

The primary goal of a Visual Synthesizer session is the production of an animated sequence that explores the nuances of pattern-based growth. Every detail of the final animation must reflect the image quality that a professional artist requires. The final output must be compatible with the highest resolutions that HD standards provide, and frames must flow smoothly. The animation sequence must be synchronized with a host of other processes through the use of SMPTE time code. The animation sequence must start and stop at specific points in time, synchronize frame rates with various recording devices, and propagate events at specific time queues. All of these factors must be persistent and provide instantaneous restoration of an environment and its viewing/recording tools.

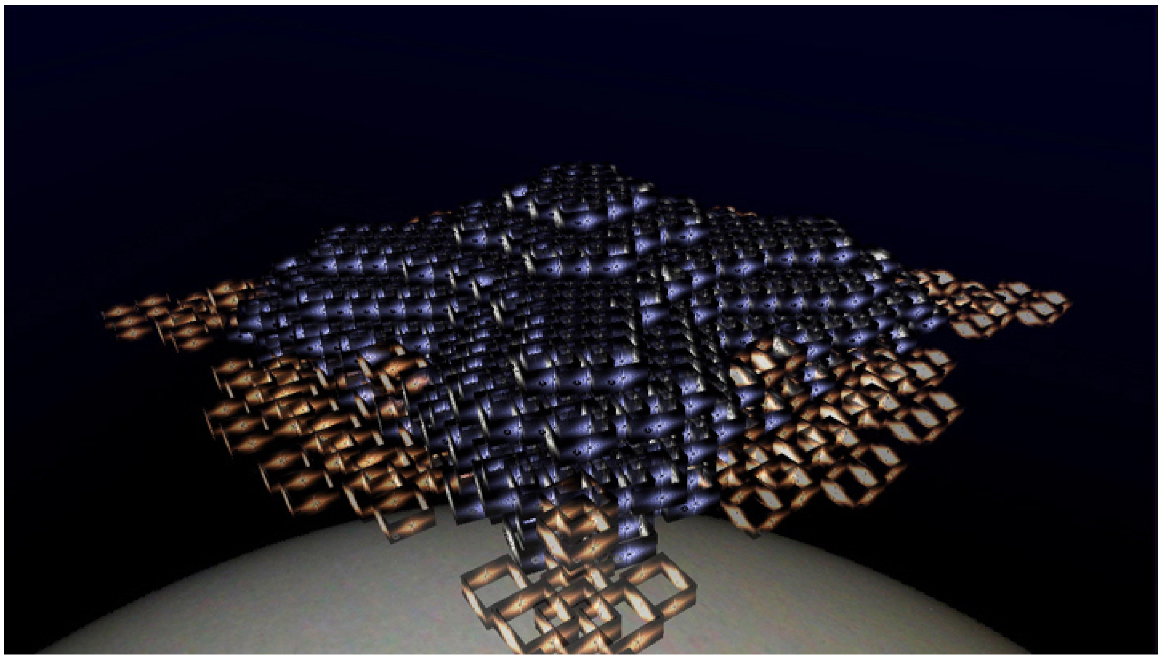


Fig 4. Three Dimensional Cubic Growth Pattern

Visual Synthesizer may be used to explore many of the well-known rules from the field of cellular automata but the real strength of this instrument is its capacity to quickly design and explore environments that dynamically evolve over time and to interface with a wide range of standard video production processes. Visual Synthesizer is an instrument that does not follow an established methodology but is simply dedicated to the exploration of any spatial behavioral pattern the artist decides to pursue.