eerm. in txoom

Yon Visell, on behalf of FoAM. *Draft*, August 2002.

Abstract

This is a (draft) overview of and specification for an evolving responsivity model component of the txoom environment in its next projected incarnation. The model consists of software which abstracts the behavior of participants in txoom as motion in an internal state space, allowing the response of the space to their movements in it to be colored by this evolving state.

Some goals for the next versions are described along with future directions and tangential nonsense.

This is a working version of the document, meant to facilitate discussion and planning for future txoom installations which may include it, such as that at Great Yarmouth.

1 Evolutionary Environmental Responsivity Modeling

For an overview of the goals, presentation, and construction of txoom, it is best to refer to other documents.

1.1 Overview

The general aim is to associate to the measured movements of individuals in the room a dynamics in the space of states that colors the rooms presentation and response to those movements, lending the longer-timescale evolution of the experiential *cycle* interesting properties such as inertia and memory, and grouping the behavior of those in it, and to do so in a non-rigid fashion. Moreover, it is desired that the geometry of the state-space itself (its shape and response characteristics) may evolve as the state itself evolves, adapting, perhaps, in a direction that better agrees (in an energetic sense, for example) with the activity it sees. We tentatively refer to such a construction as an evolutionary environmental responsivity model, or *eerm.* for short.

"Responsivity evolution" means that the reaction of txoom to actions in the space change as the cycle proceeds, in a way that depends on what has come

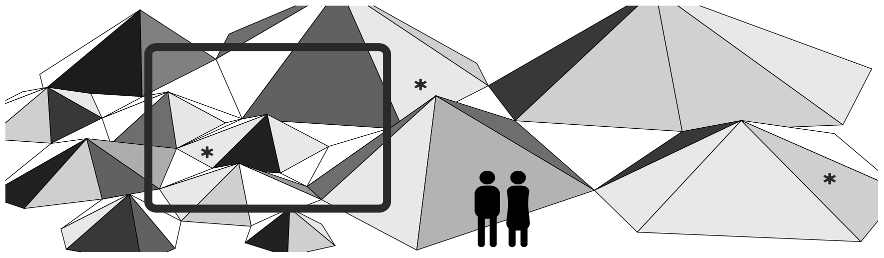

Fig. 1. the shared interaction landscape.

before. The mechanism for this is that the state information is passed on to the media synthesizers, influencing their output, and to the sensor processing/routing component, affecting the global character of the sensor data which is seen.

The result may be viewed as a contrived physics governing the evolution of a collection of point particles on a flexible geometry, thought of as the dramatic space underlying the experience (in contrast to the common alternative of a quasi-fixed narrative).

This dramatic space must be designed (in the artistic, as well as engineering sense) in a way that is compatible with, and in the best case facilitates, the general aesthetic direction of the installation.

The largest challenge here may be that of usability. The aim is to arrive at a system which is sufficiently general to be useful in the variety of incarnations that txoom and its descendants or relatives may assume, while providing a state evolution which is interesting to the designers/synthesists of the rendered experience of the space, and which provides an interface which is as flexible and transparent as possible. A tension to be resolved is that between generality and friendliness.

1.2 Interpretations

Choose any combination of the following eerm interpretations, or invent your own: *Utopian* (the world in an electronic circus-ring! Oz is the ringleader); *Positivist* (it represents the coarse evolution of a construct[ive/ed] space, an accelerated microcosmos having a duration of one cycle); *Perceptual* (the device is a multi-sensory organ [viz. GOB]); *Paranoid* (you are being monitored and probed for study); *Critical* (the directionless addition of extra technology in the vague hope that to do so is “enriching”); *Dystopian* (mirrors a world reduced to meaningless behaviors conditioned by a chaotic electronic jumble).

1.3 *Previous proto-incarnation*

A sensor-to-state data dynamics engine was built for TGarden (2001), but never publicly demonstrated with it. Structurally this engine was similar to that described below, with one important difference being that the model consisted of several instances of a one-entity state space. Moreover, design and tuning of the state space data assignment was done offline (as a hack, more or less), and in tests performed subsequently with txoom, the live-environment state dynamics failed to do anything interesting (basically did not move from the scratch state).

Some areas in which it is hoped that the next versions improve on that earlier study are as follows.

- A multi-entity (multi-particle) state space is employed, with a further possibility to represent "room" particles which may track a sort of average state for the entire space.
- The longer-scale, discrete motion (that between zones of state-mixtures, or *simplices*) will correspond more closely and naturally with the local dynamics.
- The syntax will be improved such that the integration in the system be as transparent as possible. Further, it will be designed in such a way as to make script-driven model initialization straightforward.
- Adaptation of the state-space data to recorded sensor data will be possible, and may facilitate data-driven state-space initialization.
- Visualization tools will facilitate monitoring the movements in state space of txoom entities, and in some future version, will make it simpler to sculpt the interaction topology to the desired form.
- Tools (such as directional constraints for movement across discrete boundaries in the space) will exist for constraining the movement in certain regions of the state space to evolve in a directed fashion.

Because underlying these improvements are important changes in theory and implementation, it is anticipated that they will be completed in an ongoing fashion, across different instances of txoom.

Some of these comprise research tasks, and as a result it is unknown what the precise final form of the solution will be, if indeed a useful solution can be found.

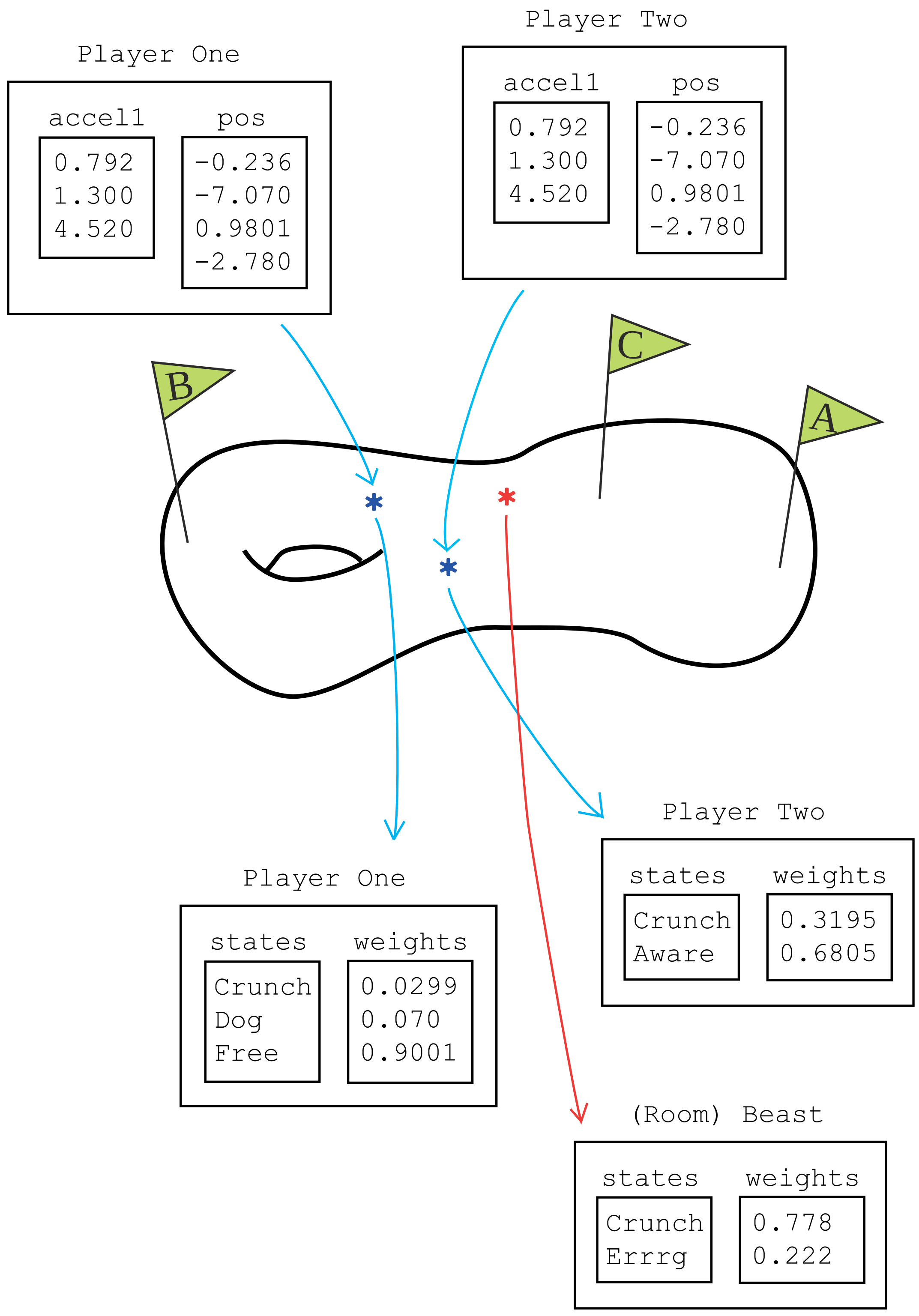

Fig. 2. A view of the *eerm*.

2 Formalism

2.1 Structure

The model is built from a sensor space S which lives over each point of a state space Σ and determines the motion of point-particles on the state space in response to the environmental data gathered in the real room. That is, the geometric domain looks locally like the product $S \times \Sigma$.

Particles representing the different entities in txoom recapitulate through their trajectories on Σ the virtual motion of the corresponding entities in the observed state space. Their motion is driven by the attraction felt by individual states of Σ for the sensor data which is associated with each particle, in a way that falls off with distance on the state space.

The output of the system consists of data describing the proximity of particles to individual states, labelled points on Σ. This output may be obtained by monitoring the evolution of the particle positions individually or by tracking the bulk motion through the insertion of room particles which attempt to follow those of the entities, reducing the bulk evolution of the latter to a sort of average progression of states.

Removed from its responsivity context, this is only a mechanism for coupling a particle dynamics on a topologically general (if regular) space with an incoming data stream.

In an electronically enhanced environmental context, the assignment of meaning and connective flow to the collection of dramatic/scene/environmental/responsive states is the most important design component in eerm implementation.

2.2 Formalism, structure

The eerm is built from a collection of states, together with a set of simplices. Each simplex is a formal normalized weighted mixture of states, and an entity may inhabit a particular mixture if a simplex exists spanning those states.

The alternative viewpoint is that there is a geometric state space, with a collection of marked points indicated states of special interest. The eerm is a computable implementation of this space, for example, through its piecewise linear approximation (although *smoothed* implementations may be achieved through interpolating algorithms such as nurbs).

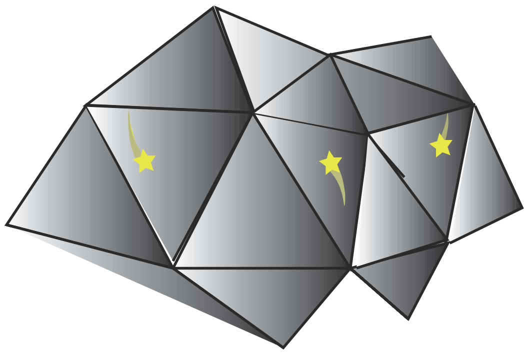

Fig. 3. View showing the discrete structure of the *eerm.*

2.2.1 Labeled and unlabeled models

A model may be constructed by specifying the number of states and their topology, or how they are connected and mixed in the complex.

2.2.2 Regions and grouping

This is where we suggest mechanisms for hierarchical grouping of collections of states.

2.3 Dynamics

2.3.1 Physics, interactions

Each simplex is the convex hull of a set of vertices, which is a linear space. Dynamics of a particle carrying sensor data s on a simplex spanned by states $\alpha = 1, 2, \ldots$ is determined by an energy function for each particle, which may be divided

$$H = T + U_{\mathrm{Local}} + U_{\mathrm{Long}} \tag{1}$$

into a kinetic contribution, T, a potential term depending only on particles and states of the current simplex, and another describing longer range forces. We might cast this in the form,

$$H_i(x_i) \;=\; \frac{p_i^2}{2m_i} \;+\; U(\mu_\alpha, \sigma_\alpha; s_i(t), x) \;+\; \sum_j V_{ij}(x_i - x_j) + L \;, \tag{2}$$

where p_i is the momentum of particle i, m_i its mass, $U(\mu_\alpha, \sigma_\alpha, s;\ x)$ is the coupling energy between the sensor data $s_i(t)$ and the vertices of the simplex, V_{ij} is an optional inter-entity interaction term, the sum extends over all particles on the current simplex, and L encapsulates the long-range potentials (those which extend beyond the immediate simplex).

2.4 Geometric flow

This is where we describe the mechanism for adapting the background geometric data to the sensor data which is being received. This can proceed either in a dynamic or a stochastic fashion, so choose one.

3 Design and topology

This is where we describe the design process involved in determining the states, mixing topology, and parametric data. This may involve some of: deciding on a collection of states, deciding on a set of simplices connecting them (or same in reverse), refining the coarse structure through subdivisions, computing statistics of typical sensor input, assigning sensor-data to states, determining default parameter values (weights for different subsets of sensor data), etc.

3.0.1 Directed dynamics

It is desired that control over the dynamics, or dramatic flow, be available in either a rigid sense (one wants that some combinations of states lead with certainty only to a specific subset of other combinations) or a fluid sense (one wishes things to flow generally from one locus toward another). Moreover, it may be desired that certain key gestures, when spotted, cause an abrupt change in the current state of the corresponding entity.

There are a few approaches to accomplish the analysis/state-correction that we can imagine:

- Topological/structural : a directed transition graph constrains which simplices may be reached from which other simplices in a directed fashion. Certain backtransitions may be forbidden, for example. If a more gradual forward motion is required, the intervening simplex may be subdivided, and directional constraints applied to the intermediate transitions.
- Condition watch : A discrete external condition manager may be designed so that a jump to a particular simplex is executed when a specific gesture or condition is seen. This assumes the existence of an extra-model analysis unit to perform the decision.
- Gesture analysis : One may measure the desired condition (e.g. arm-flappyness) directly, and pass that on to an eerm which has been designed such that the

flow that is desired to be associated (to e.g. arm- flappiness) is energetically favorable.

- Data driven : One may collect data corresponding to the requisite (flapping) behavior and train certain states such that they find it (energetically) attractive.

4 Implementation

This is where we describe the implementation details, and the (low-level) interface.

4.1 Messaging

We will only describe the subset of input and output messages which are central to the functionality here.

4.1.1 Tags:

`<Source>` : Any string identifier for an entity in txoom. With the current technical specification in mind, we can expect that the source tag may have the form "/txoom/player-1/" (insert whatever this is supposed to be).

`<State>` : Any string identifier for a state in the evolution model.

`<Simplex>` : A string identifier for a simplex, or mixture of states, in the evolution model.

`<DataVec>` : A (sub-)vector of pre-processed sensed data associated with one of the entities. Typically this will be the set of floating point values associated with a particular (pre-processed) sensor, or a single data "stream".

`<DataId>` : A numerical indicator of the sub-vector being referenced.

`<Source>/<State>` : A hybrid tag, an example of which might be "/txoom/player-1/scratch/", for use in a heirarchical named address space as in OSC.

4.1.2 Messages

Input: Data and Particle management

`<Source> <DataId> <DataVec>` : Assigns the supplied data vector as sensed data for the entity associated to `<Source>`. If no particle exists associated with the source tag, a new particle is created with that association.

`<Source> <DataIndex> <DataValue>` : updates a single floating-point value.

Input: Topology and state management.

The following messages are useful for creating a simplicial complex from a collection of named states, optionally named simplices, and adjacency relations.

`newstate <Simplex> <SimplexSize>`

`newsimplex <Simplex> [<SimplexSize> | <State1> <State2> ..]` : Creates a simplex of the desired size. The states will be auto-named something like `<Simplex>`$_1$, `<Simplex>`$_2$,...

`deletesimplex <Simplex>`

`fuse <State> <State>` : May be useful for causing two simplices to neighbor one another along a subsimplex by identifying (fusing) the named states.

`split <Simplex> <Simplex>`

`subdivide <Simplex> [<State>] [<Mode>]` : Algorithmic subdivision of the named simplex. The new simplices will have names derived from the original. The default subdivision is barycentric.

`entry <Simplex>` : Causes new particles to enter the complex at `<Simplex>`.

Input: Data initialization:

`means <DataId> <DataVec>` : Initializes means of all states for the range identified with `<DataId>` to the vector `<DataVec>`

`vars <DataId> <DataVec>` : Initializes means of all states for the range identified with `<DataId>` to the vector `<DataVec>`

Output (possibilities):

```
<SourceOut> <StateList> <StateWeights>   : Indicates the active
simplex, or set of discrete states, and the coordinate on that
simplex, or the set of weights for the occupied states.

<StateNum> <StateActiveFlag> [<StateWeight>]

<SourceOut> <StateNum> <StateWeight>

<SourceOut> <SimplexNum> <StateWeights>

<SourceOut> <SimplexTransition>

<SourceOut> <SimplexTag> <StateWeights>
```

4.2 Example: building the structure from a script

The following illustrates a possible script for building the initial state of a model. Arguments to the model itself configure the number of data components and their organization by DataId.

```
file: TxoomSampleConfig.scp

new state scratch
new state skim
new state mess
new state jones
new state insect

new simplex scratch skim
new simplex mess jones insect
new simplex skim mess jones

entry scratch

new sensor accelerometer1 3 // with 3 being the vector size
new sensor accelerometer2 3
new sensor roomposition 2

mean accelerometer1 0. 0. 0. // with e.g. two accelerometers are
mean accelerometer2 0. 0. 0. // worn by each individual
mean roomposition 5. 5.

variance accelerometer1 1.0 1.0 1.0
variance accelerometer2 1.0 1.0 1.0
variance roomposition 3.5 2.0
```

4.3 Preprocessing

It is usually necessary to apply some form of preprocessing to data supplied by hardware sensors of txoom, before this data is supplied to other components of the system, and the degree of processing which is applied may include any of the following: affine rescaling of the data, resampling, linear or nonlinear time-invariant filters, signal adaptive filters, or other ad-hoc methods. ($x \to x^2$, etc?) [1]

4.4 Sensor grouping

4.5 Real-time Controls

A number of other messages will exist to provide real-time control over the evolution.

- Coupling constants
- Masses
- Damping

or to tailor the behavior of the model during system setup to the desired configuration.

4.6 Model Import/Export

In addition to basic saving functionality, it should become possible to import and export geometric model data in one or more standard formats (`.obj` graphic vertex data, for example)

5 Use

This is where we describe usability issues, from the points of view of: system evolution (learning about the system, its future directions, technological enhancements), transparency, utility,...

6 Visualization

A few ideas follow for visualization of the model structure, and the data map it embodies.

The goals of visualization may be any of the following. To elicit an autonomous visual element for the piece; through appropriate model design, to create an electronic *fabric* upon which for things to move and grow responsively (GOB); or to provide console-like feedback on the activities in the space, for its operators (OZ) and/or its actors (photon/laser-tag centers, star trek, sodaconstructor, eeg,...)

The virtual setting for the model is a collection of particles moving on a simplicial complex Σ. One may regard the latter as an N-dimensional piecewise-linear space, akin to a manifold, except that the dimension of the simplices in the complex need not be constant. One model for visualization is to consider a rendering projection

$$\rho : \Sigma \to \Sigma_3 \tag{3}$$

associating to Σ_3 a three-dimensional projection suitable for graphics rendering. If desired ρ may have smoothing properties, etc. It is further possible to implement the ability to toggle to a mode in which the 3D spatial coordinates of each vertex come from the sensor modeling data associated with that point, encoding same in geometry (or perhaps in other rendering variables, such as color, lighting, texture, transparency, and so forth).

A further alternative for visualization might work as follows. For each entity/particle, we have a time-dependent map $p(t) : S \to \Sigma$ from the space of sensor data to the state space. One may consider just the trajectory $\Sigma(t)$ in state space. It is a collection of flagged discrete variables, which may have weights attached. Conventional visualization techniques for these abound. (Tufte...)

References

[1] Bibliographic items from the GOB and elsewhere.