behavior, animation, music:

the music and movement of synthetic characters

Marc Downie

BA, MSci, Magdalene College,
University of Cambridge (1998)

submitted to the program in media arts and sciences, school of architecture and planning, in partial fulfilment of the requirements for the degree of master of science in media arts and sciences at the massachusetts institute of technology, february 2001

author

program in media arts and sciences

Bruce M. Blumberg

assistant professor of media arts and sciences, Asahi broadcasting corporation career development professor of media arts and sciences, thesis supervisor

Stephen A. Benton

chair, department committee on graduate students,
program in media arts and sciences

© massachusetts institute of technology 2000. all rights reserved.

behavior, animation, music:

the music and movement of synthetic characters

Marc Downie

submitted to the program in media arts and sciences, school of architecture and planning, in partial fulfilment of the requirements for the degree of master of science in media arts and sciences at the massachusetts institute of technology, february 2001

abstract

This thesis begins with the idea that reactive, behavior-based 'synthetic characters' can become an appropriate platform for musical experimentation.

This idea motivates the creation of a new *behavior system* for these characters. This system, in addition to providing the basis of the work described therein, appears to solve some outstanding problems in character creation. Next, work on the creation of characters' *motor systems* is described, culminating in a new framework for characters to learn and understand their motor actions while remaining within an *example-based* animation domain. Finally, several musical applications of these systems and this *character-based* approach are discussed. These applications take the form of visual and audio interactive installations that consist of, in part, musical creatures built from this framework.

thesis supervisor: Bruce M. Blumberg. assistant professor of media arts and sciences, Asahi broadcasting corporation career development professor of media arts and sciences

behavior, animation, music:

the music and movement of synthetic characters

Marc Downie

submitted to the program in media arts and sciences, school of architecture and planning, in partial fulfilment of the requirements for the degree of master of science in media arts and sciences at the massachusetts institute of technology, february 2001

Tod Machover

Professor of Music and Media, Program in Media Arts and Sciences

thesis reader

Emilio Bizzi

Eugene McDermott Professor in the Brain Sciences and Human Behavior

thesis reader

acknowledgements

I would like to thank the following cast: **Bruce Blumberg** – direction; **Bill Tomlinson** – production; **Michael Patrick Johnson** – technical; **Ari Benbasat** – pastoral; **Tod Machover** – for my time in MediaLabEurope; the remainder of the synthetic characters group at The Media Lab with whom I've worked closely (they are: **Rob Burke**, **Scott Eaton**, **Jesse Gray**, **Michal Hlavac**, **Damian Isla**, **Yuri Ivanov**, **Matt Berlin**, **Chris Kline**, **Ben Resner**, **Ken Russell**, **Song-Yee Yoon**) – environmental; **Linda Peterson** – patience; my **parents** – understanding; **Alison James** – love.

contents

list of figures

introduction

Reactive behavior systems are powerful decision making architectures. Here we concentrate on the creation of *synthetic characters* with such architectures. The resulting authorship process can focus on high level concepts like 'intentions', 'expectations', 'emotions' and 'memories'. However, as reactive systems, they traditionally know nothing about time.

Both music and movement, on the other hand, are structures in time. Can a unified approach to the generation and management of temporal patterns within these architectures, informed by musical goals, help solve some of the big problems in synthetic character design? Can the resulting approach be used for both the animation of characters and the music that they produce? Can this architecture be used as a basis for new 'motor learning' style techniques in both motor control and music?

By drawing such parallels between musical problems and the problems of behavior and motor control we create here a system in which characters movements can be shaped by training in real-time. Conversely, by creating music with the resulting systems the goal is to be able to perform similar training and shaping within a musical domain.

This thesis is the introduction and an exploration of an idea: character-based music.

why a synthetic character?

the synthetic characters group

This thesis work built on and integrated with the work of the Synthetic Characters Group at the MIT Media Laboratory. As a group we are fundamentally interested in the problem of how to build intelligent systems. Our methodology is to create complete, but simple, autonomous characters. These characters are situated inside virtual worlds that are in turn situated inside complete interactive installations.

Participants are typically presented with a graphical and aural window onto a virtual world. In such an environment the actions, interactions and emotions of the characters must always 'make sense', be believable and compelling to those participating in the interaction. It is equally important that the creators of these characters understand and can conceptualize their creations.

why characters?

The idea – create interactive installations with compelling synthetic characters – is a good one. Expressive autonomous characters can offer participants modes of interaction and relationship not shared with other artifices. In this reasoning we are surrounded by a galaxy of influences and ideas: the success of Disney and the animated cartoon [89]; our willingness to ascribe intentions, beliefs and desires – to take the "intentional stance" [27] – towards creatures; our fundamental need to construct "theories of mind" [5,70]; our familiarity with creating and maintaining narrative forms involving characters; and the fun that we derive from playing with things that play back with us. I propose that these ideas and their power survive beyond different surface aesthetics and different media.

Looked at from the other side – that of the author of the installation – the reasons for the character based design are equally compelling. Cre-

ating an interactive artifact starts with what might be the ultimate "blank sheet of paper" – lacking in conventions to use and exploit, lacking in constraints to direct or focus. Breaking down the design into terms of characters gives us a place to start. Begin with their stories, flesh out their interactions with each other and their world; take their motivations for and their reactions to these interactions. With this come constraints and direction – complexity through interacting simplicity, the visibility of actions; and references to well known conventions (of character, cinema, computer game and literature). Such 'old media' bearings are essential to new media works [73].

two plateaus

expressive characters

Not surprisingly, creating compelling *autonomous* characters turns out to be rather hard. What they need to survive in their virtual worlds, made unpredictable by our participation and other characters is some kind of intelligence – an autonomy through an intelligence, that is, to replace scripts and direction that they can no longer have in this medium.

Prior to the commencement of this work the Synthetic Characters group had built two large public installations based off essentially the same mass of engineering and the same mass of conceptual technique – this system is well described in [49] and is a clear direct descendant of the ideas in [10].

A longer critique of previous group work is found within, but the three main problems that [the authors of] synthetic characters faced were:

- our synthetic characters needed awareness of their own bodies.
- our synthetic characters' were naïve about time and the temporal extent and location of actions. They were unable to represent or predict future states – fundamental to representing the passage of time.

- modelling, learning or adaptation were not well integrated into our characters – neither conceptually or in the engineering.

As the characters we sought to create become more complex, character authors were spending more and more time treating the symptoms of these problems. The lack of a general solution to any of them impinges on all character design. Both animals and music are telling us that we are deficient – that our characters need to know more about their bodies and the flow of time and that neither of these concepts are modelled well statically.

This thesis is an attack on these problems.

intelligent interactive music

We are thus interested in creating adaptive music creatures. But with few exceptions, music composition tools which seek to incorporate machine learning techniques are not interactive. Were we to try to make them interactive we would fail. Brooks' [16] searing critique of "classical" artificial intelligence's inability to survive in a real world while grounded in a symbolic world applies in full force to these (hypothetical) interactive composition tools. After reading Brooks, the state-of-the-art intelligent musical instrument parallels classical AI robots of the 1980's. The field is replete with expert rule systems, knowledge bases and applications of a variety of formal logics [58] – all deeply deliberative, formal and brittle techniques unsuitable for live interaction in unpredictable musical worlds.

On an opposite side of the AI spectrum are the connectionist approaches. These too are present in the computer music composition field (often in the guise of "dynamical systems"). Some of these approaches have the benefit of being data-driven, others at least purport to *listen* to music in some way [91]. Either way, their problems (again, should they be made to survive in an interactive installations) too are well known – they might make excellent modellers of musical structure within a limited domain,

but representation alone without "action" is insufficient to take us close to an interesting inter*action*. Here, and in the now common place data-flow language music paradigm, the apparent complexity of the 'action' is far greater than the mechanisms behind choosing which action to take.

This work can find its roots in the reactive, behavior-based AI approach, and the creation of strong music with a reactive approach would in itself be a contribution to the field. But creating reactive music is itself insufficient. Compelling characters, and I argue, compelling interactions, are more than musical robots that 'do the right thing' or react in the right way. They are creatures whose reasons, intentions and feelings for reacting can be read in interesting ways.The proposed approach here is to make music with synthetic characters; to make interactive installations, performances and pieces with musical creatures.

This work then informs a general point about the interactive installation across media. It is clear that as such installations increase in complexity, and hence decrease in transparency, mechanisms must be found to maintain participants interest. Further, in a field increasingly constructed with complex tangles of data-flow language, we as authors must find our bearings somewhere. By aiming to build complex installations with *intentional characters* we hope to achieve two things:

- allow people to take an *engaging* intentional[1] stance towards the elements of the installation, forging relationships like playing, exploration, recognition.
- setup and maintain the expectations of participant through referencing the conventions found in cinema, computer games, and literature.

1. When forced to interact with complex systems over a long term, people will take an intentional stance towards them – "oh, my computer/car/video is mad at me" (indeed Dennett gives uses a central heating system as an example). But this is intentional stance taken not out of engagement but out of desperation – in general people are not *forced* to use our installations.

These two goals are well addressed in theory by the character metaphor.

For this to occur we have to do a lot of work. The problem cannot be approached piecemeal. It cannot be solved by creating a new "music system" for use by current character systems. First, we create a new kind of behavior system; then we develop new approaches to motor control; finally, we explore what music might come out of these elements. This thesis follows that order.

The result (and this work ends after only the beginning of the exploration of such characters) is a radically bottom-up approach to interactive music. It draws inspiration from synthetic characters, obviously, but also from animals; it takes a certain amount of hope from the recent interest in the biological origins of music [99]; and it seeks to mimic and steal parts of both traditional approaches to computer music composition. But it seeks to build interactions rather than knowledge bases, musical competencies rather than musical rules and music creatures rather than music machines.

behavior

In this section we give an account of the design and implementation of a new *behavior system* for synthetic characters.

Here we address two of the inadequacies mentioned above – the characters' awareness and representation of time, and the integration of learning and adaptation into the heart of the behavior system.

Further, this redesign of the creature kernel enabled the rest of the work of this thesis to take a particularly clean conceptual form. Indeed this work began with the realization that many of the motor problems that we'd like our characters to solve have significant and interesting behavioral components.

The structures and techniques described below are, therefore, the context and terminology for the remainder of this work.

a new behavior system

context

The design in the new behavior system took place within the context of the *Synthetic Characters Group*. Inside this group this new system, and the engineering behind its implementation, is to be the foundation of a large number of future interactive installations. It is therefore, first and foremost, a set of general tools for authoring the characters in such installations. No one installation could motivate or inform all this work.

A small galaxy of ideas and goals have in some way influenced its design, not least of all, the previous installations of the group [48,95,7], based upon previous behavior systems (most typically the *Scoot* system [49], extended by [49,103] and customized for [95]). And, as a technology base shared by the group its development is permanently work-in-progress, worked on by all present members of that group.

goals – learning

One principle difference between this new system (called *innards*) and its predecessor *Scoot* is the decision to have adaptation and learning built into the core – and not just the numerical adaptation of, say, connection strengths, filter constants or statistical models, but support for true, 'live' changes of complexity and topology in response to new experiences.

We want characters to appear to posses long term changes, learning and adaptation firstly because these things are strong indicators of intelligent behavior. The lack of them threatens "the illusion of life" in the long term and strongly limits the kinds of relationships that people can have with characters. Long term changes are required for characters to *recognise* people and for people to recognise their past interactions in present ones. We integrate learning into other aspects of character design including consistent internal states (motivational and emotions)

and appropriate behavior towards appropriate goals. These things lend long term *believability* and *readability* to characters.

guiding learning – "extending the percept tree" on page 48; motivation and emotion – "soft state" on page 78; design learning – "learning to replace inverse-kinematics", page 116; long term persistence – "persistent 'mergeable' graphs", page 148.

Secondly, when implemented well, these mechanisms are useful for the creation of characters themselves. Complex characters are complex systems to author; learning can provide a substitute for cumbersome, brittle and over-fitted engineering. We will see several examples of the adaptive abilities of this system being used to *bootstrap* characters during their authorship and the increasing use of this style of design is one of the long term design goals of this system.

The inclusion of learning presents a number of interesting aesthetic challenges to authors – it, indeed, challenges the *authority* of authors. Given a system that can and will grow in complexity at run-time and perhaps over even longer periods, how does one guide what will be learnt ahead of time? From an artistic point of view it is in no way clear what it means for a system to 'come up with something new', or clear *a priori* how to maintain an artistic vision in such an environment. The complexity of this ambition makes reasons why we go down a character based path towards artistic ends even clearer.

Finally, there are engineering problems associated with creating and using systems that can grow 'by themselves'. In particular they demand the ability to put characters into persistent storage for later analysis and the development of tools with which to conduct that analysis.

goals – *the year of the dog* project

But what kinds of learning? and where to look for guidance? Central to the work of the group is the idea that the best way of creating synthetic characters is to build strong synthetic *creatures* and work out how to stage them. This motivates a focus on nature's (i.e. non-synthetic) creatures as good models for our work. "The year of the dog" project – a group wide focus on installations exploring the creation of virtual dogs took place during this thesis work. Two installations came from this. The first, a training scenario where a (participant controlled) shepherd

trained an autonomous terrier (Duncan). This very simple platform installation was created to test and facilitate our work on learning and adaptation inside behavior system, as well as to similarly demonstrate the new engineering behind these ideas. The second installation "sheep|dog: *trial by Eire*" places the dog and shepherd into a sheep-herding game. Dogs and doggish problems will appear throughout this thesis as motivation and example.

But what are dogs doing in a thesis about music? The purpose of the (ongoing) project is to force us to look again at all the issues involved in creating virtual creatures. For the participant, the dog represents a rich, fantastic and dangerously high set of expectations and conventions. For us it presents a challenge that is at the bounds of plausibility (unlike, say, a virtual human) – and a challenge that is ideally suited for working on training (behavioral and motor learning). The dog forces us to focus on hard problems that we know have been solved (by nature), that we know are worth solving – where we know how well we are doing. The music can only benefit from this.

shaping - learning by successive approximation

One of the great things about dogs is that they combine their strong characters with a high level of personalizability. This ability to *train* dogs is particularly interesting to us here. The training scenario itself comes with a set of conventions about what you can train about a dog and how you should go about it.

An increasingly popular training paradigm is *clicker training* [101,71]. Here we begin by associating the sound of a hand held "clicker" with the delivery of a treat to the dog. This clicker, soon becoming a good predictor of a food, has the advantages of speed, clarity and uniqueness over the hand held treat. This clicker (backed up with food) is then used to differentially reward behavior that the dog owner is interested in.

aural models – "cepstral models" on page 65;
simple shaping – "simple shaping" on page 114;
complex shaping – "learning to chase your tail – part 1" on page 140;

This owner can teach dogs remarkable things. Consider the kinds of motor learning that can be *shaped* in a dog: one can teach a dog to not only shake its paw, but shake its paw in a variety of different ways. Such problems will be called here *simple shaping* problems. Here a continuous variety of qualitatively similar animations must be offered and understood by the motor system and selections in this space learnt by the behavior system. Ideally, in our characters they should be learnt by a process that mimics that used with real dogs – a process of successive exploration and approximation by the trainer and by the dog.

A training situation with a well behaved dog might proceed as follows: initially the trainer rewards the dog for sitting down (that is putting itself in a position where it might shake its paw). This results in a dog that spends much of its time sitting. Then any kind of a 'shake' is rewarded by the trainer. Soon, the dog is shaking his paw often enough for the trainer to begin to raise his reward criterion to higher hand higher "shakes". Thus over time, the height of the dog's 'shake' increases, the (motor consequences of) the dogs behaviors has been shaped. Once (and only once) the behavior has been made to appear frequently, then cues (typically vocal or gestural) can be associated with "shaking" by rewarding this behavior in the presence of the cue.

A more complex example concerns the creation of new animation material. The problem is as follows: if you can direct a dog's nose (typically with real dogs you can use a "training stick"), you can teach a dog to chase its tail. You achieve this by a similar procedure as above – the trainer shapes the dog through a new "animation".

One intermediate goal of the work here is to build a character that can be trained in all of these fashions. If we were to achieve a solution to the complex shaping problem then we really would be taking interactive character animation into a different world. The quanta of virtual creatures has remained the "atomic" animation, at least in the "example-based" interactive animation school of which this work is part. The "ground truth" of action section selection has been found in the limited

number of animations that can be used to indicate that an action is occurring.

Our attempts at a solution to both these problems will be fully described by a discussion of the motor system – still, while designing a behavior system we need to keep our eye on both these key problems.

'shaping' and music

What problems present themselves when trying to learn these new animations? Firstly, a reward regime requires a modelling representation of body movement which the behavior system can use for both the analysis and synthesis of movement. Secondly, from this representation we must be able to produce variations, if the dog is to explore the space of possible movements. During all this the dog's body must manage external constraints - remaining on the ground - and conflicting goals - say, tracking the object of interest with its head. Lastly, this training must be done within the context of the motivational and emotional state of the character.

Unexpectedly, the dog's complex shaping problem relates to problems in interactive music. From a musical angle we are concerned with the creation of new musical material. Firstly, computer music too requires a flexible notion of the start and end of an 'action' or phrase. Secondly, the production of variations and explorations of a musical action is central to music itself, and certainly central to any representation we might wish to use. Managing the external constraints during this improvisation becomes managing the influence of other musical elements in the world, and the resolution of conflicting musical goals are important musical problems for any interactive music.

phase one — building blocks

Now we shall introduce and describe the fundamental building blocks of this new behavior system. We'll begin with the structures that represent actions (the *action-tuple*), the values of actions and categories (the *percept)*. After, we'll look at the dynamics of these structures and how categories are created and learnt. Finally we'll look at extensions and more sophisticated implementations of these structures.

the action-tuple

A behavior system, as the name suggests, is built of behaviors – actions that a character can perform. How these actions are represented and how they are selected between are the defining features of a behavior system.

Inside the core of this behavior system the primitive action type we'll call an *action-tuple*. These explicitly represent hypotheses about the world. Each action-tuple is a statement concerning the likely *outcome* of a given *action* in a given *context*, together with the likely *value* of doing this.

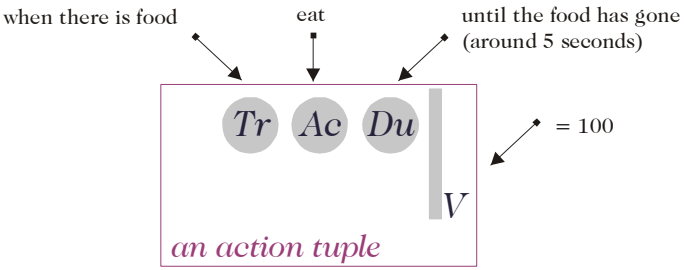


figure 1.introducing the action-tuple

A concrete example: an action-tuple inside our dog might be written in english as:

when there is food | eat food | until it is gone, in around 5s; V=100

This statement says that the value of 'eat food' when there is food is 100; that we should continue to 'eat food' until it gone (usually this would be a command to the motor system); and that this usually takes around 5 seconds (say, for a hungry labrador). Note that it is typically the case that we include some timing information.

Each action-tuple, then, consists of four parts. These parts appear so often in this text we'll refer to them by abbreviations. They are: the **Tr**igger context in which an action ought to be performed (*Tr*); the **Ac**tion itself (*Ac*); that we will '**D**o' this action '**u**ntil' a particular context is achieved (*Du*); and the **V**alue of this statement (*V*).

These statements are designed to be simple but powerful. While their theoretical expressivity is a fraction of full blown logic programming languages (like prolog, or custom logic/imperative hybrids like Funge's "congitive modelling language' [34]) they can, for example, capture all of temporal logics primitive relationships [2]. And we need the simplicity – not just for aesthetics. The simplicity and the *constraints* of this representation are what will enable the automatic creation of new action-tuple statements at run-time. It enables it by not only making the problem of which hypothesis to generate next tractable, but also accessible, guideable and understandable by the authors of the characters themselves.

computing expected values

Perhaps the biggest difference between an action-tuple hypothesis and logical statement is that action-tuples have *value*. At any time an action-tuple can be asked to compute modulation of this, an *expected-value*. This is an estimate of the *value* of carrying out this action at this time – this is not necessarily constant or equal to *V*.

For the food example, we have a situation where initially, the expected value of eating is related to the presence and type of food – in the absence of food, we cannot eat. If we are eating, however, the expected value is related to the amount of food that remains, or at least to whether or not we've finished eating. This situation, where the expected value of an action-tuple is a function of Tr while inactive and *Du* while active, is common throughout the system:

$$\text{action-tuple.expectedValue}(t) = \begin{cases} Tr.\text{evaluate}(\text{t}) \times V & \text{if inactive} \\ (1\text{-}Du.\text{evaluate}(\text{t})) \times V & \text{if active} \end{cases}$$

This above is a codification of the usual definition of expected value. At first, before you attempt an action, its expected value is related to the intrinsic value and how *closely* the world matches its trigger condition *Tr*. While you are performing an action, the expected value is related to how *distant* the world is from the ending condition (including the expected duration of the action) *Du*.

Action-tuples are typically stored in *action-groups* where they typically undergo competitive action selection ("the action-group", page 41) based on these expected values. Thus values must serve as a common currency between disparate action-tuples.

percepts

One of the features of this representation is an equivalence between the constituent parts of the action-tuple – Both of *Tr*, and *Du* have identical functional interfaces and acquire their semantics through their rôle and use in action-tuples. And these interfaces include the evaluate function described above. But what does that usually mean? And how do we implement something like *Tr*: 'when there is food'?

The answer is to encapsulate the ability to determine 'there is food' inside an object, usually creature wide, that I will term a *percept*.

This object servers to encapsulate the flow of information from whatever it is that can supply the necessary data for the food decision, to a model of that data that represents 'yes, food is there' for this data – typically, this is a statistical model that is capable of generating some probability estimate. But this arrangement might fall anywhere between the unimaginative 'world variable `food_is_there` is `true`' or a full blown computer vision inspired model looking at a virtual camera onto the world, initially trained to find dog-biscuit colored objects. Either way, `Percept` can represent a barrier separating this specific implementation engineering from the core of the behavior system.

percept trees

A final important feature of `Percept`'s semantics is that it can be a creature wide container of a concept – say, many *Tr*'s can look to the same percept. If its model changes, all action-tuples that are associated with it are effected. We will return to the procedure for updating percepts below

Percepts can be arranged in trees. Such trees (by being set up before hand) can guide the generation of new action-tuple hypotheses about the world. For example a typical dog tree might consist of the default 'whatever' percept, one that says 'yes' regardless of the state of the world – acting like some blank prior (see figure below). Children of this percept might act as 'something was said' and 'something moved', children of these would contain speech or motion models capable of matching specific utterances. Hypothesis generation can be guided by producing new

updating percepts – "percept updating – model building", page 46

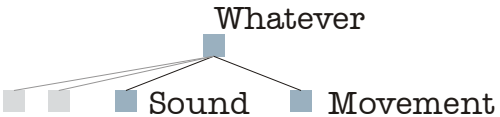


figure 2.initial percept configuration might include "sound" and "movement"

child hypotheses with *Tr* pointing to child percepts. Rhythmic structures are well described by such trees of action durations in *Du* percepts.

It is worth noting that we can use percepts as part of our description of an action *Ac* itself – i.e. statistically model some parameters about the action. Such *Ac* percepts can themselves be located in trees, enabling behavior systems to try out whole classes of actions and slowly refine them. This will form the basis of our solution to the simple-shaping problem, is fundamental to the expressive power of the system, and can provide an abstraction barrier between the behavior system (deciding upon the action) and the lower level details of the motor system (dependent on the description of the action).

Finally, it is important to realise that these trees can grow (and shrink) during the life of a creature, in response to both behavior and perception. It is to these dynamical properties that we now turn.

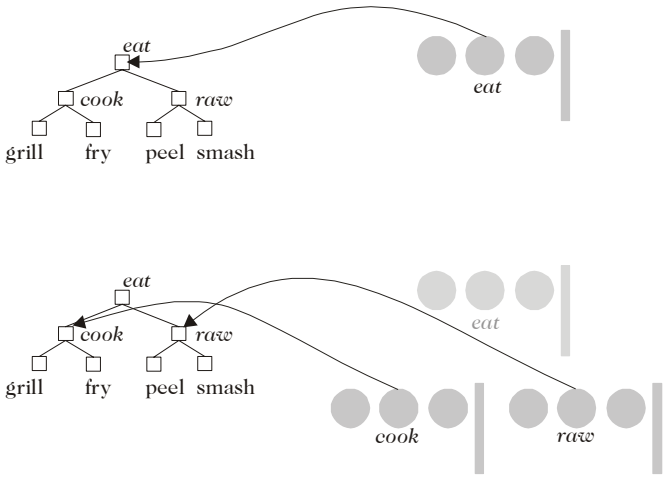


figure 3.action trees can group similar classes of actions together to guide experimentation

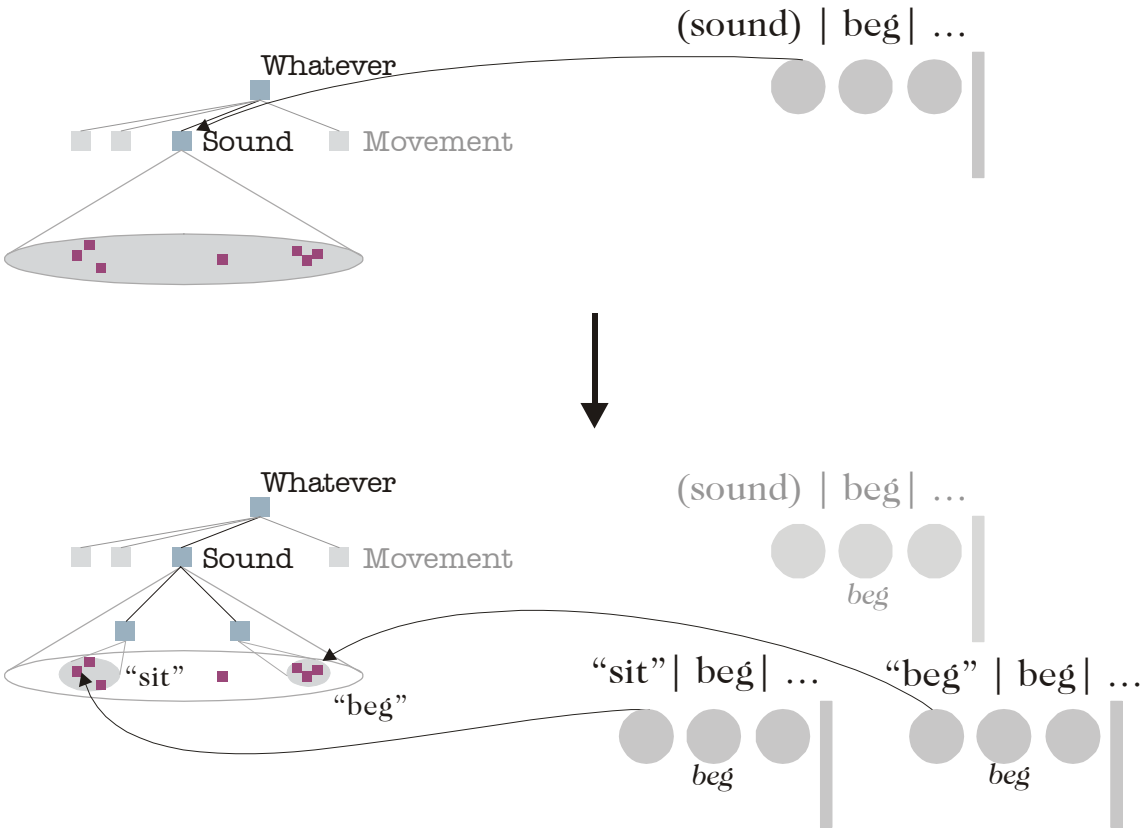


figure 4.percept trees guide new action-tuple hypothesis generation

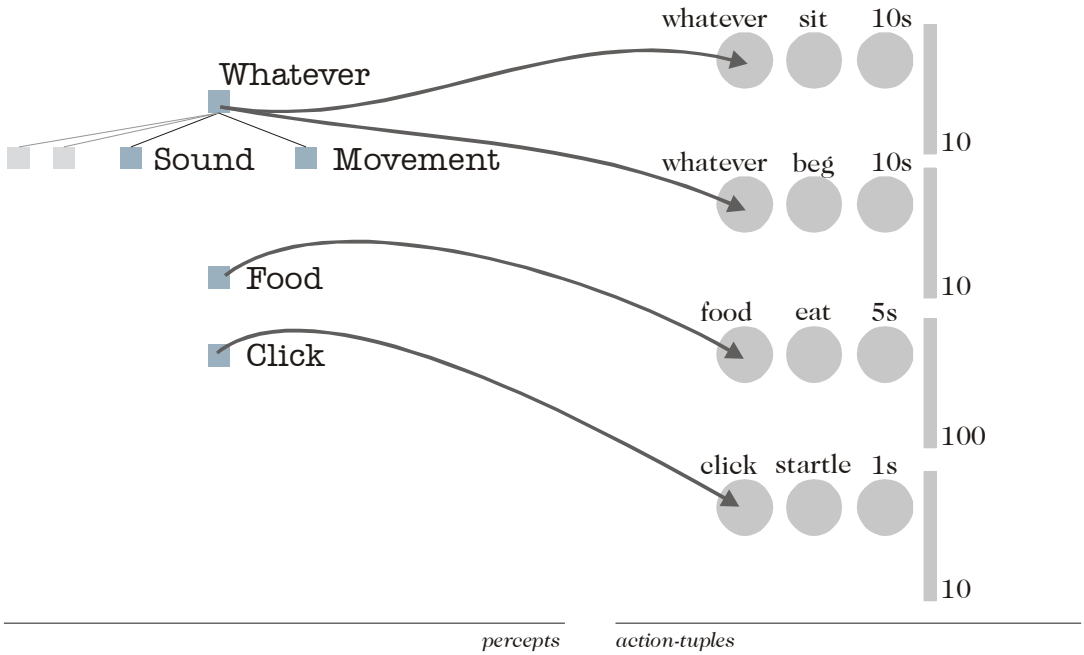


figure 5.building
up a dog – part 1

phase two — dynamics

state objects and decaying memory traces

Now we've introduces the action-tuple and the percept we need something that connects the categorization abilities of percept to actual data to categorize. This last basic ingredient is the *state-object*. Such objects wrap up data flows that percepts require. Here we also augment these transient flows of information with both a history and a computed *saliency*. This history is just the ability to recall the data flow from an earlier time. This ability will be particularly useful during the adaptation of action-tuple values.

The saliency is more interesting – it is taken to be a low-level measure of how stimulating this data flow is. Simplest examples might include volume for a speech flow, or brightness for a flow of visual information – the difference between a loud bang and a soft whisper. Here we expect to find a layer of signal processing – it is here also, where we put simple sensory habituation and sensitization models. The memory can also facilitate the regeneration of other sensory phenomena such as masking and blocking. In fact the architecture explicitly encourages the construction of simple data-flow networks to process and produce this saliency information – the history contained at this level allows non-causal filtering to take place.

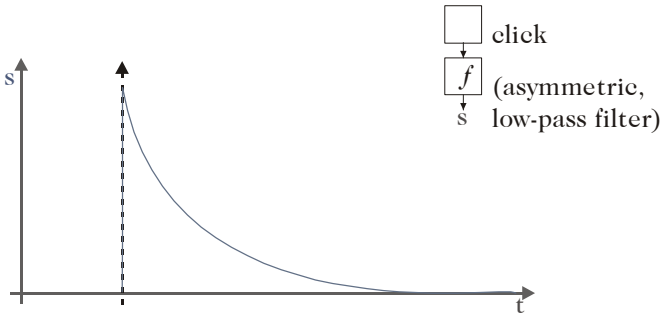
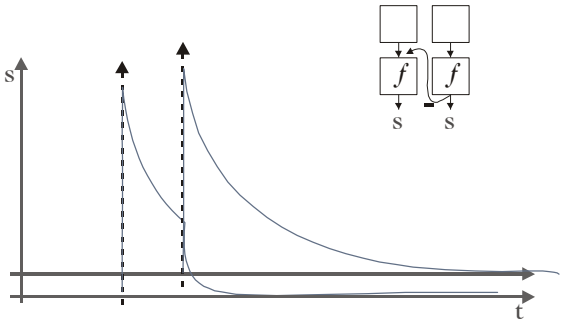
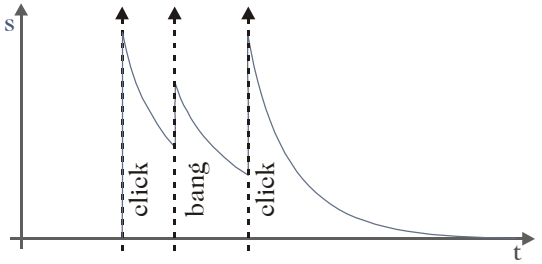


figure 6.graph of saliency over time for a simple



two StateObjects interacting, one inhibiting the others



StateObject keeps track of a data history

StateObject's saliency is also where a characters' motivational and emotions have the chance to modulate perceptions at the lowest level – where, for example, hunger may modulate the saliency of a pre-wired food smell flow. While many of these examples might be modelled with more sophistication closer to the core of the behavior system, it is an important to have a place in the behavior system architecture for the simplest of approaches – and this data flow paradigm is especially useful for the 'quick sketch' of an idea.

StateObject, therefore, can be thought of representing the 'decaying memory traces' of transient, unclassified events; or a lower level background tracking of continuous processes. Either way, it keeps a reference to some object that represents this source of this event or content.

the action-group

Now all the principle components of the system have been introduced. Given a group of action-tuple hypotheses how do we chose between them, i.e. how does the character decide what to do *now*?

A good action selection strategy uses this state balances a several goals. The problem of action selection in the abstract and concrete is discussed much more fully elsewhere (e.g. [15] and for creatures [10]). Our main concerns here include:

- **don't dither** vs **appearing aware**. The dog should not change rapidly between actions, there should be a *hysteresis* introduced, reflecting a cost associated with starting a task. It is more important to get *something* done (i.e. some action completed) that to get exactly the thing with the highest expected value.

  However, the dog cannot carry every action that it performs out through to the bitter-end without seeming absent. It must still, for example, react to sudden loud noises. Tied up in this is what Brooks terms as *coherence* – showing just the appropriate level of persist-

ence towards a task or goal selected from a large variety of (possibly conflicting) goals – and *relevance*.

- **explore vs exploit**. Always choosing the action with the highest expected value is a pathological behavior – without exploration we will learn nothing. But so is neglecting to do the "obvious thing" or exploit the situation is equally uncompelling – there is no point in learning something only to fail to take advantage. This especially becomes important in systems where the innate values of things are being *learnt*.
- **overlapping action vs temporal constraints**. This concerns what it means to "have done something" (as opposed to getting half way through something). A good action selection mechanism must be able to support doing more than one thing at a time. Part of the coherence is offered by the structure of the action-tuple. But in the abstract, this opens up *all* of the issues concerning time discussed previously.

a baseline action selection mechanism

Our base solution to the action selection problem goes as follows: initially probabilistically select an action with probabilities proportional to it expected values. Having selected one, start executing it, and record all action-tuples base expected values. Interrupt it if some action-tuple grows to some multiple of its base value *and* it is higher than the current expected value. In this case select again.

In pseudocode:

$\boldsymbol{A}$ = {action-tuples }, $\boldsymbol{S} = \boldsymbol{A}$. expectedValue(t)
start
{
    choose *active* from $\boldsymbol{A}$ with probability distribution $\boldsymbol{S} / \Sigma S_i$
    last_active = *active*
    active_values = $\boldsymbol{S}$
}
otherwise
{
    ∀i, if ($S_i \geq active\_values_i$) and ($S_i \geq S_{last\_active}$)
    {
        choose *active* from A with probability distribution $\boldsymbol{S} / \Sigma S_i$
        last_active = *active*
        active_values = $\boldsymbol{S}$
    }
}

There are some extensions to this basic recipe. For example, we can parameterize the exploration / exploitation tendency for fine control of the stochastic selection process by pushing the expected-values through a Boltzmann function $p' \leftarrow \exp(V/T)$. High 'temperatures' T, and the action selection mechanism explores more, for low temperatures it sticks to choosing action-tuples with higher values.

Other extensions improve computational efficiency of action-groups. We do not have to ask for an expected value from every action-tuple at every action selection. Several heuristics can guide a partial evaluation of the action-tuples – we might evaluate low value action-tuples less often, or action-tuples which have not won recently less often. Later we will see action-tuples capable of capturing activity patterns between action-tuples, and we can use these models too, to evaluate action-tuples based on how likely they are to become active in the near future.

In fact throughout the remainder of this chapter, we'll see more complex additions. This action selection mechanism has, as written above, moreover shown to be robust in practice.

credit assignment

value updating

Credit assignment is another of the duties of the action-group mechanism. For action-tuple values V are typically not static, but instead change over time. Here we have our first taste of learning.

The simplest credit assignment strategy is the back-propagation of value, with discount. In this strategy some value of a newly active action-tuple gets propagated back to the last active action-tuple. Some notation:

$$V_l \leftarrow V_l.\texttt{blendTo}(V_n, \alpha)$$

here V_l represents the value of the last active and V_n the newly active action-tuple. α is the blend parameter. The intuition behind this is that values of appetitive behaviors should be related to consummatory actions that they lead up to. Large positive consummatory actions (e.g. eating food) provide sources of value for the system grounded in tangible quantities (e.g. food or drive reduction). The inner structure of V can be quite complex, and we will see some examples of this later. Credit assignment could extend backwards along whole chains of appetitive actions at a time, and hidden information could be kept inside the V implementations. But for now, we can still consider V to represent a real value. So, the above equation turns into:

$$V_l \leftarrow (1\text{-}\alpha)V_l + \alpha V_n$$

which can be viewed as the simplest, infinite-impulse response low-pass filter, filtering an 'input' of V_n. This aspect of the behavior system is closely allied to traditional *temporal difference* learning[88].

The default, base α is left as a free parameter, a global control over the plasticities of values (which, again, can be locally modulated). But there are some important complications which effect both V_n and α in this credit assignment process:

- **the discount parameter**. Things which lead up to eating food, while beneficial cannot be as valuable as eating food. So we propagate back only a fraction of the value backwards from actions. Thus we replace V_n with $V_n' = \beta V_n$ in the above expression, with β as the *discount parameter.*
- **all equivalent action-tuples take place in credit assignment**. We back propagate value to *all* action-tuples in the group (including last active one) with blend parameters modulated by their *proximity* to the last active action-tuple and define:

  $$\text{action-tuple}_1.\texttt{proximityTo}(\text{action-tuple}_2) \in \mathsf{R}$$

  Many times these proximities are zero – they have nothing in common with the last active action-tuple – which results in blend parameters of zero (i.e. no change).

  However, consider the situation where there are three hypotheses in our dog:

  whatever | sit | for around 10s; V = 40

  any sound | sit | for around 5s; V = 20

  "sit" heard| sit | for around 5s; V = 10

In this example situation, say "sit" is heard, but, the first action-tuple happens to win – this is possible (indeed likely) because of the stochastic action selection mechanism and the first tuple's higher value. When passing the reward back (caused by the dog biscuit) we must reward all action-tuples that result in a "sit", the external world does not know, and should not need to know, which tuple became active. In this example all these *Ac* are identical giving them proximities of 1 which respect to each other. And it is the usual case that the computation of the proximity is delegated to *Ac* – we will see cases where proximities are somewhere between 0 and 1.

complex proximities – "learning to chase your tail – part 2" on page 145

- ***Tr* modulation**. To make the above strategy work, one final ingredient. We further modulate the blend parameter by the value of *Tr at the time that this action-tuple became active*. The intuition at work here is that,

  any sound | sit | for around 5s; V = 10

  should only be effected by a credit assignment procedure at all if there was in fact a sound.

percept updating – model building

As said before, the percepts referenced by *Tr*, are responsible for containing a model of (a part of) the world context. A character could start off with pre-fabricated classifications that never change. But in the same way that characters ought to be able to learn the value of action-tuples, part of the adaptation that takes place in our characters should be the discovery of new categories from perceptual data. An important goal of this state discovery project is that it be *value-driven* – that the effort (and computation expense) gets spent and accuracy obtained in the regions of state space that are relevant to the performance of high-value actions.

The easiest way of achieving this is simply to not update the models of percepts that no 'high value' action-tuples are associated with. This 'high value' threshold that action-tuples must go beyond is left as a free param-

eter, and is perhaps best specified in terms of the average value of all action-tuples.

Once we have decided which percepts will be updated and which will not, we need to decide what goes into their models. Here the history that StateObject keeps around helps us to the right thing – for *Tr* percepts we need to keep track of what *leads up to* an action being activated. This is in the past, since the model updating and credit assignment take place after the action is completed. Here the acausal filtering that StateObject can perform can be used – particularly salient events should mask less salient events forwards and backwards in time. Other options include updating the model with most salient event inside a window around the start of an action.

The full model that a percept can contain is one which models how a partition of state space relates to incoming value. This forms a linkage between the value propagated back onto action-tuples that relate to this percept and the data that is being modelled.

Modelling all this is a complex task however; not only does one have to deal with one extra dimension of data (value) but one must deal (process or store) with *all* incoming data coming into the model.

A quicker, less complete, approach is to maintain a model of the portion of the state space that leads to the execution of actions *more* 'valuable' than the current action-tuple. This hard categorisation of data into 'valuable' and 'not valuable' lets us present the data to more traditional modelling techniques. Now we can press into service textbook models of data – one dimensional second order distributions, 'standard' speech processing modelling techniques. We are also now in a position to train neural networks or other classifiers on this data.

We have to realize, though, that we use most of these techniques without hope of any proof that they are correct or powerful. We are tracking not just non-stationary distributions, but distributions with non-stationary qualities affected by the actual modelling process itself. This is a problem

widespread throughout embodied intelligence (and certainly multiagent systems), and not confined to this behavior system. My personal bias is towards accepting this as a necessary feature of this field and keeping the models as simple and *human readable* as possible.

extending the percept tree

At its lower levels the percept tree structure naturally corresponds to a cluster structure in statistical modelling techniques – and this is no accident.

The percept tree can be extended as models that represent 'clusters' fission, it can also shrink as child clusters fuse again. Statistical models differ on whether the can they maintain cluster structure locally, or they require a global view – in which case the highest level percept in this portion of the tree is an ideal place to locate this functionality.

Either way, we can again focus attention on the parts of the tree that are linked to 'high value' action-tuples, and only review the clustering topology in these areas. This 'high value', this *innovation threshold*, higher than the threshold over which statistics are maintain, is another free parameter of the system.

When percepts split, new action-tuple hypotheses are generated. When, for example, a percept that a *Tr* is connected to splits then new action-tuples will be created with *Tr*'s that reference the children of that percept. These children percepts might not necessarily be new; the trigger for this behavior might be the action-tuple itself becoming active with a value above the innovation threshold.

Some of the 'zoo' of models, including their fission and fusion properties, currently built will be detailed in the next section.

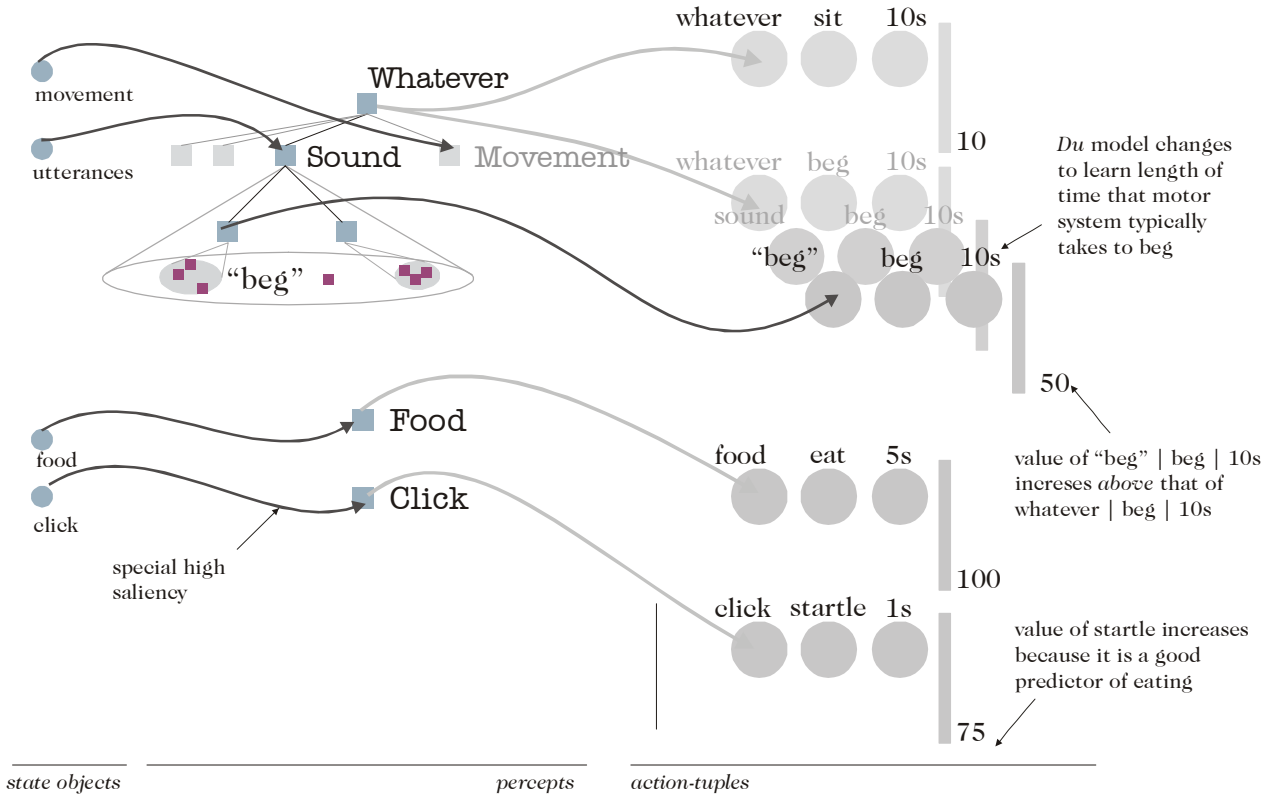


figure 7.building up
a dog – part 2

phase three – models

introduction

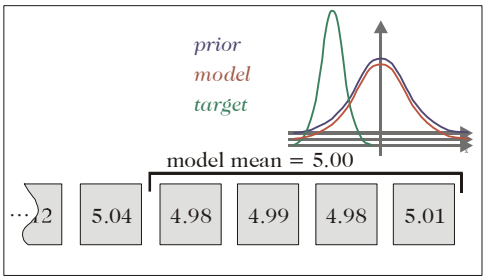
Having set up this framework above, we now go on to explore some of the results that fall out of it. Here we investigate what models we need and what we can do with them; here we also work through what it means for a *Du* percept to split, or to have trees of *Ac* percepts.

Some of the models have concrete applications – we have aural models, we have models of spatial relationships. Others are more abstract – we have models of durations, sequences and simultaneities. In this section we get glimpses of musical structures, for these models are the concrete and abstract building blocks of both organised behavior and organised sound.

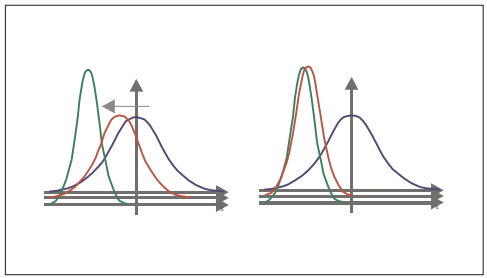
Tr percepts

simple generic one-dimensional models

A simple model to consider is a model of a single real variable. As discussed earlier (see "percept updating – model building", page 46) the easiest path to take is to choose to model based on 'higher-value' transitions rather than model a two dimensional distribution that includes the value of the transition.



1. To track a non-stationary one dimensional 'distribution' we use a rolling buffer. Initially we fill it by sampling our prior distribution



2. Over time, this distribution approaches a target distribution - the distribution of the 'rewarded' class. We generate samples for use by the creature from the current distribution of the model modulated by our prior. The prior ensures stability.

figure 8.a Gaussian model forming a model of the world

combinational models

It is worth mentioning in passing that we can create percepts that look to other percepts – models which say "this *and* that" or "this *or* that". Programmatically we are in the territory of fuzzy logic, and we can take such metaphors over completely into this system. Such logics consistently define the Boolean operations "and", "or", "not" etc. in such a way to extend them to handle "fuzzy" membership. Here `Percept`'s are seen as objects that produce fuzzy membership measures from `StateObject` data flows. The problem of how to learn (fuzzy) Boolean relationships from scratch is still an open one, although guiding the generation of these models by a pre-made percept tree will suffice for simple cases.

activation models

Later, we will see situations where more than one action-tuple may be active at any one time. This will open up the need to coordinate the simultaneous or overlapping activation of action-tuples.

We can turn percepts towards the action selection mechanism itself and build models of the patterns of activation that arise inside action-groups. Some things need to be done together in order to work – we can use these models to capture these contingencies. I suspect that these models will be important in handling overlapping and blended motor actions for complex bodies as well as handling (generating, analysing, and recognising) basic temporal structures like 'the chord' in music.

The simplest start on such a model is simply a Boolean model in a percept set to look at whether a single action-tuple is active or not, itself a Boolean value, denoted by A_i . Such a model maintains statistics about the value propagated back in credit in the two scenarios, A_i = `true` and A_i = `false`.

This approach is suitable for only the simplest of cases. Not only is it hard to construct or induct useful percept trees based on these models, these models do not capture particularly profound activation relationships.

Instead we can keep statistics about all of the overlapping activations. We define a 'distance' D_{ij} between two action-tuples i and j with D_{ij} not necessarily equal to D_{ji}, initialized to some large distance M. For each discrete time-step we update:

$$D_{ij} = \alpha D_{ij}$$

if both i and j are active or both i and j are inactive, and:

$$D_{ij} = \alpha D_{ij} + (1-\alpha)M$$

if one is active and the other isn't. This distance then decays during correlated activity, and increases during uncorrelated activity (this is, in fact, nothing more than a Hebbian update rule). An extension turns this into a useable model – we must replace α in the above equations with either a β, given by:

$$\beta = \alpha^{\text{time inactive}/\text{time active}}$$

in an update where i is active, or by a γ, given by:

$$\gamma = \alpha^{\text{time active}/\text{time inactive}}$$

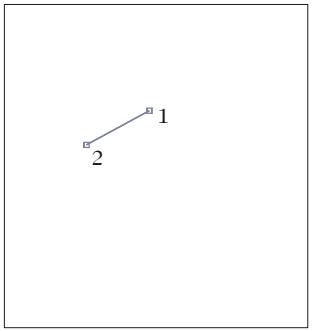
in an update when i is inactive. The intuition: compensate for the fact that action-tuples might spend much time inactive anyway, without there being anything profound to draw from this fact.

Finally, we can note that we don't need a full $O(N^2)$ update each cycle. Firstly, we only need to update every time an activation or inactivation occurs (compensating, again, for different time update intervals, by raising the filter constants to some power of elapsed time). Secondly, we might only keep track of action-tuples after they first share an activation together[1].

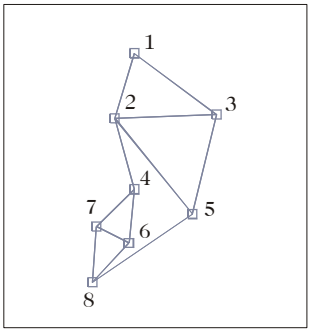
The result of the model is a dense tangled graph of connected nodes representing action-tuples. These nodes cluster into cliques of correlated activity patters. After a while, we will want to create new percepts and assign them to these cliques – percepts which represent this inducted correlation. We can do this if we can automatically isolate a number of cliques from the graph. A procedure for achieving this segmentation will be given in the chapter on animation, where we will see many, many more of these weighted graph structures.

1. Note the bias here towards common *activation* as important, rather than common inactivation. If this is what we want, they we might update D_{ij} differently (with a different α) in the case of common inactivation.

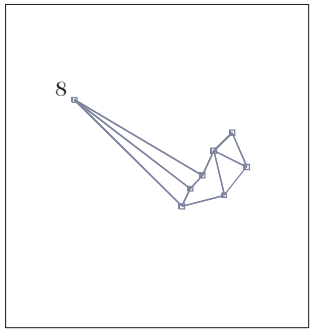
1. As discussed in the text, these simultaneous [in]activation models can be drawn as a network. Here the distances represent connection weights - governed by the duration of overlapping [in]activity



2. Here two action-tuples "*1*" and "*2*" have been simultaneously active; so we add them to the model, with a default distance between them.

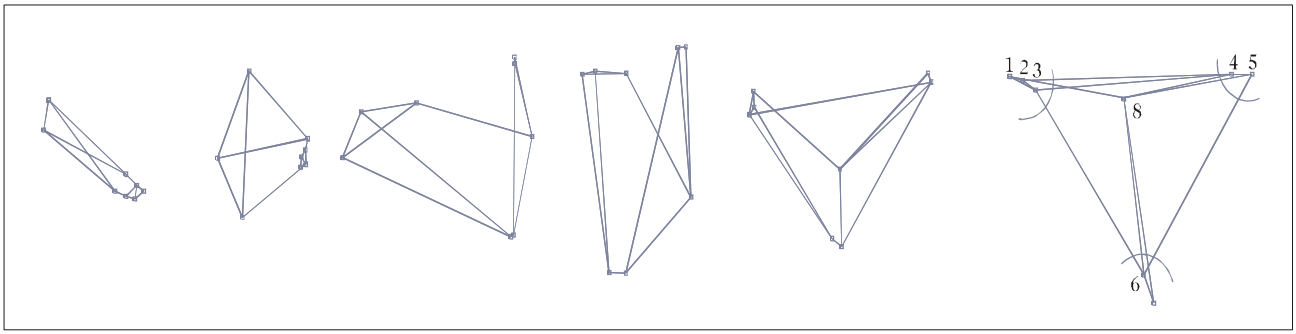


3. after a short sequence of random activations the model looks like the above. Note that some action-tuples have not been simultaneosly active (e.g. 5 and 6) while some have shared more a more correllated activity pattern (e.g. 6 and 7)

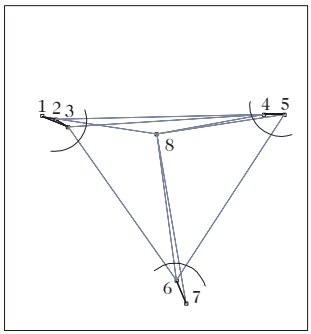


4. We repeat a similar exercise here, but very soon into the sequence we make sure that action-tuple '8' is *never* active at the same time as anything else, and when nothing else is active it is.

gure 9.clique formation



5. Here we see the progressive formulation of three “cliques” of activation, over quite a short time. (approximately 10 activations, over 100 updates)



6. These cliques can be segmented out to form higher resolution, child percepts. The graph segmentation algorithm is disscused later.

5. Here we see the progressive formulation of three “cliques” of activation, over quite a short time. (approximately 10 activations, over 100 updates)

other models

Clearly there are a host of other models that we can borrow from the textbooks. Some textbook classifiers would make a welcome addition to our collection of models. These algorithms, which take in data and target 'labels' are ideal – for discrete labels we can use 'valuable' (i.e. the assigned credit was more valuable) or 'not valuable' (i.e. we transitioned to a lower value action-tuple); for continuous labels we can use the actual assigned value itself. Later, we will see how to keep confidence statistics inside action-tuples. Such statistics can ground the learning rate parameters of these models. However, to dynamically build percept trees with fission and fusion with these models will need another ingredient. This can often be provided by a percept higher in the tree that manages its children. Further on we'll see an example of the maintenance of a population of models that themselves have no idea how to split – see "listen (installation)", page 190.

temporal models

simple durations

Throughout the development of this behavior system, we have see action-tuples described like:

whatever | sit | for around 5s; V = 40

The 'around 5 seconds' is a temporal model, stored in a percept, referenced by an instance of *Du*. The action-tuple, upon starting, can decide how long it ought to be active for by sampling this model. When an action-tuple stops being active it can update this model with the length of time that it actually was active for. The simplest of such models (other than the static, fixed period) is the Gaussian model described above combined, again, with a suitable prior.

But we can take this much further. The explicit duration modelling found in the most primitive atom gives us the ability to create sophisticated temporal constraints and interactions at the heart of the system.

forward transition models

Du is another suitable place for activation modelling, so we can just reuse the activation models from above to, say, give coherence to when groups of simultaneously active actions end. Another model, unique to *Du*, models action-tuple transitions – in short, *what usually happens next*. Initially, we might do this by simply keeping track of the number of transitions that occur between action-tuples. Some notation:

$$p(i \to j) \approx \frac{N_{ij}}{\sum_{j} N_{ij}}$$

where N_{ij} represents the number of recorded transitions between action-tuple i and action-tuple j.

However, there are two things we need to add to this model. First, there needs to be a way of evaluating the probability for an action-tuple transition that we've never seen – that is we need a *prior* probability of any given transition. Secondly, we need a model which tracks potentially non-stationary transition probabilities – what usually happens next might change over time.

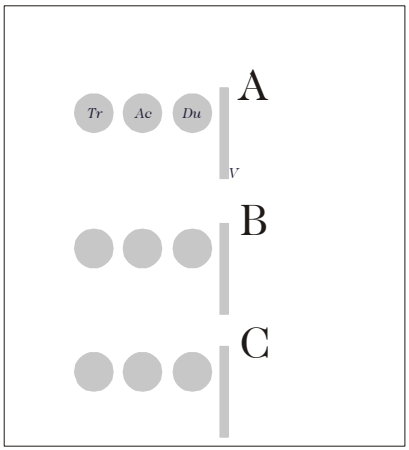
What we have, for N action-tuples, instead:

$$p(i \to j) \approx \frac{N_{ij}' / \sum N_{ij}' + \alpha}{1 + \alpha N}$$

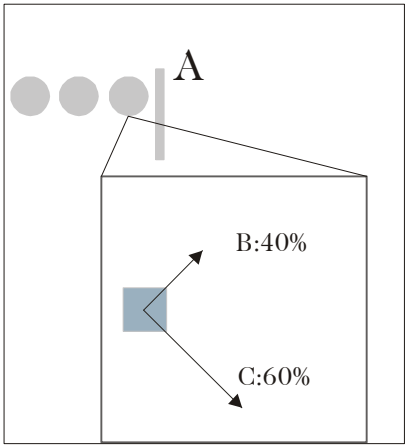
with N_{ij}' representing the number of transitions recorded *recently* (say, within the last 10 transitions) from action-tuple i and action-tuple j.

Once we have these models there are a number of things that we can do with them – in particular they represent a simple set of expectations over what will happen next. But one thing that they will do, by virtue of their position inside the action-tuple – they will perturb the action selection process with the contents of this model. Here we might reinforce the chances of high probability transitions from recurring – lowering the bar for likely transitions and raising it for less probable transitions.

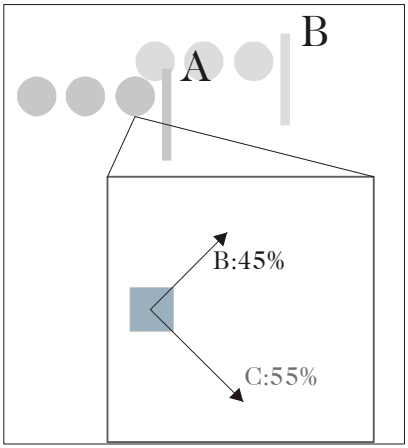
Finally, if we keep, in addition to the statistics above, information about the values of particular transitions, we will have grounds for these percepts to fission apart – creating sub-hypotheses which can be refined (in value, in content and in trigger percept) independently.



1. Consider 3 action-tuples, here, labelled *A, B* and *C*.

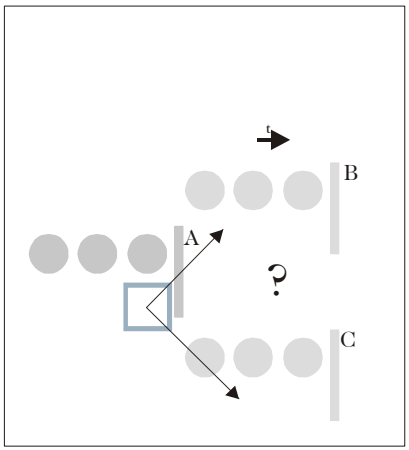


2. Inside a Percept referenced by each *Du* we keep statistics concerning the number of times that we transition to every other action-tuple. These Percepts may be kept privately (local only to a particular action-tuple) or they may be shared between a group of action-tuples.

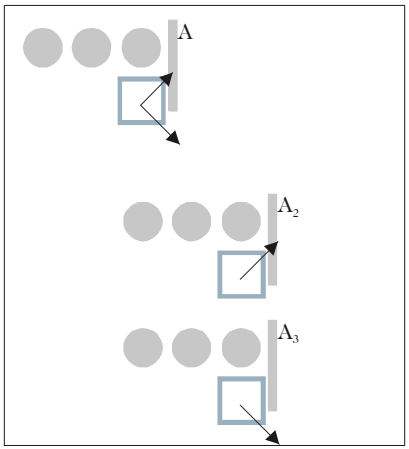


3. These models look at information generated by the action-selection process - in particular the transitions between active action tuples. This is typically done in addition to keeping track of the likely time duration of events.

figure 10.forward transition model-ling



4. While we are building these models up, we can use them to bias the action selection mechanism. Here we can use these statistics to bias which action is selected next, or bias other actions' ability to interrupt this action.



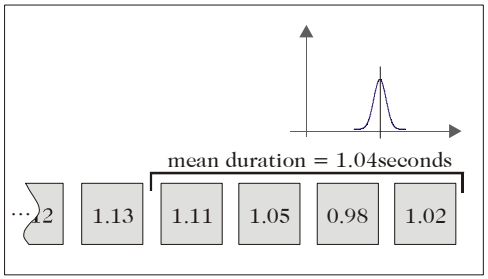
5. Because these models are embedded in percepts we can reuse the Percept fission and fusion mechanism to allow new action-tuples to be created by these forward transition models. This allows the behavior system to explore and refine versions of action-tuple *A* that lead to *B* separately from those which tend to lead to *C*.

exotic duration models

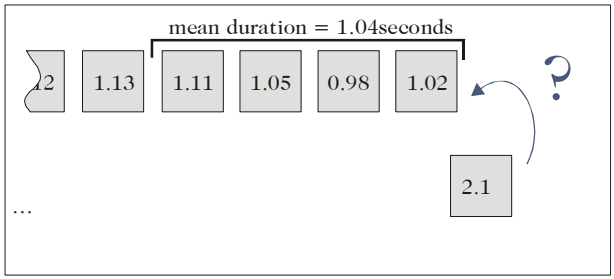
It might come as no surprise to learn that duration models have musical applications – such models will act as the building blocks for larger temporal structures.

The simplest duration model – the simple length model – is going to be useful here. But we can go push the central idea – duration modelling – much further into musical territories.

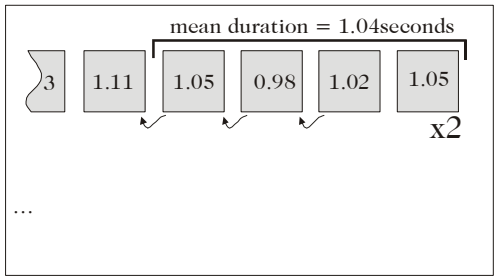
Firstly we might build models that are aware of temporal structures. For example, consider building a duration model, responsible for working out how long an action ought to last, that embodies the preferences for low-integer timing ratios. Before new action durations get added to the duration model an attempt is made to reinterpret them in terms of the existing model, specifically in terms of low-integer ratios of the existing model's mean duration. For a diagrammatic illustration of how this works see figure 11 below. This reinterpretation process is governed by prior or learnt preferences for certain time intervals.



1. here we are modelling a time duration by computing second order statistics on a rolling buffer, or a *first-in-first-out* queue. This is just the "simple duration model".



2. Now a new duration is added to the model. It is clear to see that this duration "2.1" really ought to be interpreted as twice the duration of the model. Adding this to the buffer directly is a mistake if we want a model that knows about the simplest element of rhythmic structure. So, recognizing, that it can be better explained by the model as a low-integer ratio (here 2:1) of the mean duration, we add 1.05 to the model.



3. For *generative* purposes we keep track of the reinterpreting ratios – for when we come to sample this model, we first sample from a distribution of these ratios, then sample the model and then finally multiply them together.

figure 11.low integer ratio blind duration models

Secondly, we note the ability to share percept models between action-tuples, by placing them in percept tree structures – we can use this to provide first order temporal coherence between actions – a coherent pulse across action-tuples.

A duration percept tree structure has two uses. Firstly it can directly represent (through it's hierarchy) the hierarchical structures that are undeniably useful for music [46]. Secondly, it can be used to guide the creation of and maintain competing hypotheses about the timing structures that should be present inside a collection of action-tuples. The maintenance of a population of (possibly competing) views and interpretations is a common theme throughout the computer music understanding literature (take two extremes: the grammar based work of [46]and the connectionist work of [91]). Later we'll see examples of percept trees used this way – *soundcreatures*, page 175 and *listen*, page 190.

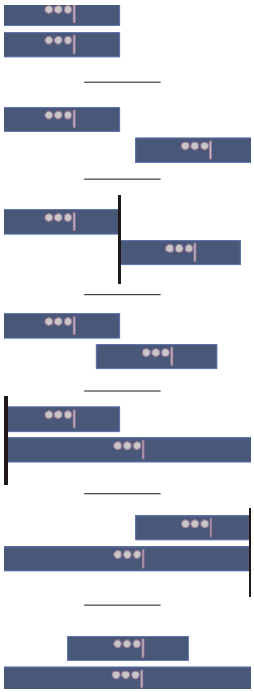


figure 12.temporal relationships

temporal logics

The field of formal temporal logic has a lot to contribute to the description and recognition of complex contingent arrangements of sequential and overlapping actions. But its hard to know where to include the temporal versions of predicate modal logics. Most useful for our purposes is the "interval algebra" [2,69]. While an introduction to this area would take us too far away from the purpose of this thesis, it is sufficient to realise that the action-tuple structure can capture the primitive temporal relations of the interval algebra (all thirteen of them, roughly: *equal, before, touching, overlapping, during, simultaneous-starting, simultaneous-ending* and 6 inverses). We can, by engineering constraints on the trigger conditions (*Tr*) and ending conditions (*Du*), enforce such relationships between action-tuple activations. These structures too, have been shown to be important in modelling music [58], and although no piece here has as yet used these structures, we note that we have the opportunity to play with then in a generative model within our framework.

aural models

cepstral models

We can produce behavior in response to spoken utterances and other discrete sounds by reusing some models common in the speech recognition community. The model chosen in our group are based on a, so called, *cepstral* analysis of sound segments (bounded according to amplitude threshold). This analysis produces the *short time, Fourier transform* of the logarithmic (short time) power spectrum of the sound. Incidentally, this technique is common to both speech recognition and pitch detection – making this engineering the basis for more musical applications (see, *soundcreature*, page 175).

We can define a Euclidean distance metric between vectors in this space. Better, we can define the distance between these utterances as the smallest of all possible distances between one vector and time warped versions of the other. (for details, [105], for theory, [72]).

Categorizations of these utterances naturally fall into clusters. By fixing cluster sizes before hand we can create dynamically growing trees of utterances and hence dynamically growing trees of `UtterancePercept`'s. Other options include fixing the membership of high level percepts during authorship (or a special training phase) and preventing the alteration of this data.

With either arrangement this branch of the percept tree begins with a percept, 'any sound' that simply responds to the presence of sound. Its model collects utterances associated with rewarded action-tuples. And, upon an action-tuple attached to it becoming important, it can perform a segmention of all the utterances it has collected in its model. While musical applications of these models have been the focus here, making robust distance metrics has been the work of others in the group [105].

phase four – extensions

This section shows how we can create more complex characters using the ideas described above. This behavior system is implemented as a collection of interfaces, then base implementations. (An interface is strictly a contract to implement certain methods). While the last section looked at some of the implementations that can be used for percept models, this section looks at some extensions. This programming style is based upon an idea that heterogeneity can lead to *simplicity* — that given a multitude of tools for creating parts of characters that the simple things remain simple, and the complex stay possible,

So the options really do 'fan out here' and this is by no means the last plea for heterogeneity in this thesis. A similar call for hybrid, *bricolage* style systems will be heard in our discussion of motor systems, where no one paradigm seems to solve all problems. For example, one of the themes of this section is the use and the creation of motivational and emotional state inside our creatures. Currently we have no principled way of approaching these topics – we are still very much learning by building. Out of this, and only out of this, might come some formal ideas.

other values

what is value?

Value gets its meaning only through the action selection mechanism. Regardless of the intricacies and extensions, action-tuples with high expected values are more likely to become active. Such action-tuples are the actions that are most valuable for the character to do at that instant.

This definition hides a significant problem, well illustrated by two examples:

- **the *lion problem.*** Here there seems to be a conflation between valuable and "good". Consider a zebra character. It is clearly *very* valua-

ble for such a character to run away from lions. Indeed if the zebra didn't run away in the presence of a lion, then something would look wrong.

The issue now is that actions which typically *precede* seeing a lion and running away from it accumulate, via the credit assignment procedure, value themselves. In short, the zebra learns to run up to the lion just so that it can run away again.

- **the *contrast problem***. Similar problem arises with *startle* behavior. Recall that this behavior is important in the clicker training paradigm – "startle", as a result of hearing the highly salient click emitted by the clicker, precedes the reward of food. Therefore one would expect "click | startle" to accumulate value from "eat food" and then the actions that the trainer is training, say, "beg" to accumulate value by preceding "click | startle".

  Here the issue is that throughout this training regime the dog *always* startles. So the initial Value of "startle" should be high (to ensure that, in the presence of its Tr, namely a click, startle always occurs). But if the initial value of "startle" is so high then we're effectively back to the lion problem, unless we increase the value of everything else to compensate. This is an unacceptable "catch-22" scenario.

Recall that the expected value of an inactive action-tuple is given by the formula:

$$\text{action-tuple.}\texttt{expectedValue}(t) \;=\; Tr.\texttt{evaluate}(t) \times V$$

Allowing both values and saliencies to become negative offers the simplest solution to the lion problem. The chain discussed for the lion problem looks like:

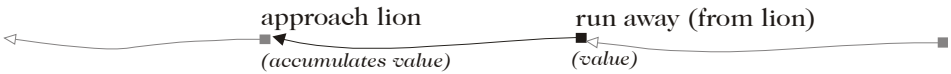
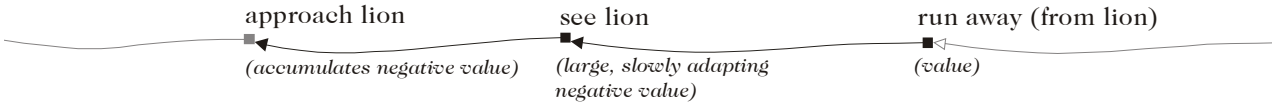


figure 13.the lion problem

If we adapt this chain to become:



The high negative value of "seeing a lion", coupled with the negative saliency returned from the percept "lion", conspire to ensure that not only will this "action" become active in the presence of a lion, but that preceding actions will be strongly suppressed. This "perception action" has a value of far lower *plasticity* than is typically found in an action-group – reflecting the fact that the negative value associated with lions has been acquired through mechanisms that are no longer plastic (e.g. evolution or early nurture).

This is not the hard-coded "hack" solution to the lion problem that it might appear (at least, no more so than evolution or nurture is) – the negative value of seeing a lion, and the negative potential encoded in the saliency of the lion percept both make sense – this is where the implicit ideas of "good" and "valuable" get disentangled. But we need to make sure that we do have a mechanism by which these negative values and saliencies can be created through adaptive processes.

The startle mechanism is created in a similar way. Extending the range of saliencies outward by allowing them to be greater than 1 enables the (highly) salient "click" to ensure that the initially low value startle action does not start off as a primary reinforcer for behavior in itself. As the value of being startled increases perhaps lower `StateObject` mechanisms

habituate slightly to the click. For the sake of simplicity we might choose to implement action selection for actions that trigger off of highly salient stimuli differently for the mass of other actions – separating them into a separate "startling" sub-group that have priority (hierarchical action-groups formalize this idea, page 74). Either way is acceptable, for these two approaches are identical in the limit.

It is interesting to note that the solutions to both these problems ultimately involve time. There is a conflation of the time-scales on which learning occurs here – the zebra doesn't learn that the act of approaching lions is good or bad, the zebra arrives complete with that knowledge courtesy of evolution, or early nurturing. Neither does the dog learn to whether or not to "startle" – it is reflexive, unavoidable.

There is a conflation too, of perception, action, and emotional response that happened in the resolution of the lion problem – this will be discussed later.

We can solve these problems while retaining a scalar Value. But it is questionable whether these approaches scale to more complex characters, or solve all problems associated with this one dimensional Value. Further, soon we will see action-groups used for different purposes, which in turn require more exotic Value implementations. While it is true that all we need to select actions is a consistent idea of value, it is not true that characters don't need to encode and expect the difference between good and bad explicitly.

complex value implementations

Value V has been discussed so far as if it were a single, floating-point number wrapped in the semantics of a low-pass filter, smoothing out changes in its value brought about by the credit assignment process. But neither its scalar nature or this piece of signal processing is required by its interface, and the above section suggests that more dimensions may be required.

A closer analysis of the role that V serves in the behavior system reveals that the only constraints on it are that there must exist operations:

- V_1.toDouble(t) ∈ R — V can be converted to a real number for the purposes of providing a proportional probability for action selection.
- V_1.compareTo(V_2) ∈ {same, bigger, smaller, unknown} — V_1 can be compared with all other V_2 inside an action-group.
- V_1.distanceTo(V_2) ∈ R and V_1.blendTo(V_2, α) — finally it supports a metric and a blending method consistent with a metric:

  V_1.distanceTo(V_2) = 0 ⇒ V_1.compareTo(V_2) = same, and

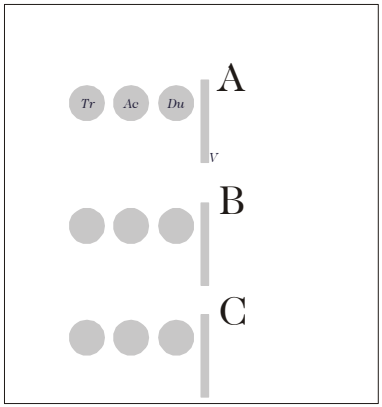
  V_1.distanceTo(V_2) < V_1.distanceTo(V_2.blendTo(V_1, α)) for all α

No transitivity demands are made on the underlying V. Further, different action-tuples can have different implementations of interface V (although engineering abstract algebras is troublesome).

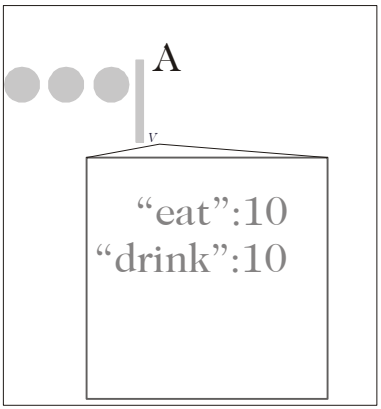
value chaining - the *precedes* relationship

The first class of more complex V implementations includes those that augment a scalar floating-point value with other information. This information, irrelevant for toDouble, distanceTo and compareTo operations, is blended by blendTo in the same way as the scalar value.

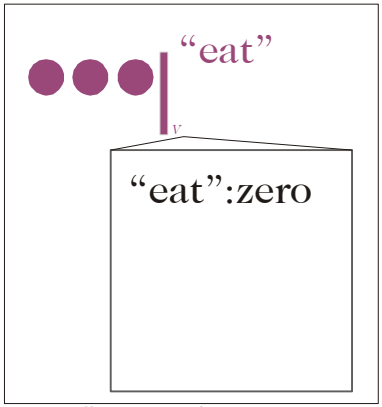
These rather loose requirements open up the possibility of using V implementations to store information about transitions between action-tuples. Consider introducing sources of *tracer dye* into the behavior system. During a V_1.blendTo(V_2,α) operation we propagate some amount of this hypothetical "dye" back from V_2 to V_1. After a while at each V the concentration of these "dyes" gives an idea of the distance to the sources.



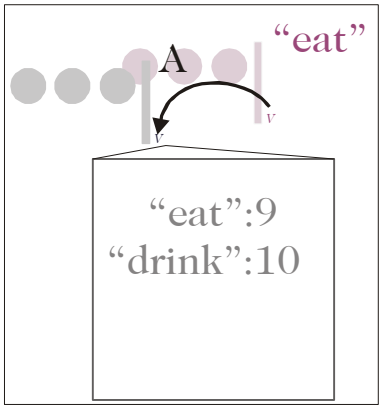
1. Consider again, 3 action-tuples, here, labelled *A, B* and *C*. In this miero world, there are in addition, two consumatory actions “eat” and “drink”.



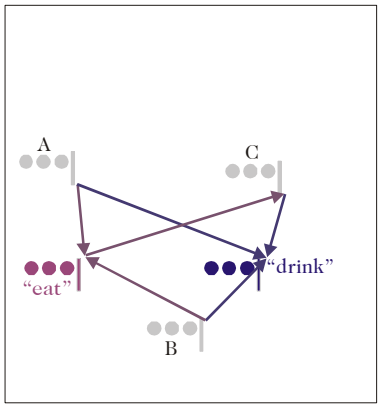
2. Inside a “complex tagged Value” we keep information about the “distance” to consumatory (or “goal”) actions. Here we initialize this distance to the consumatory actions to be 10, some large default value (we could do this



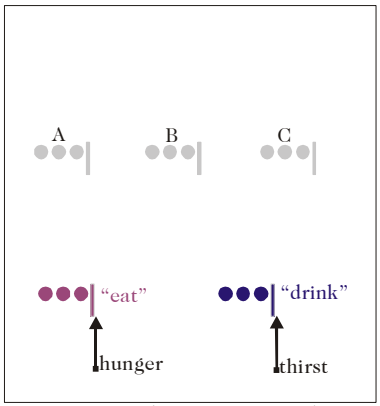
3. Trivially, “eat” is distance zero away from “eat”



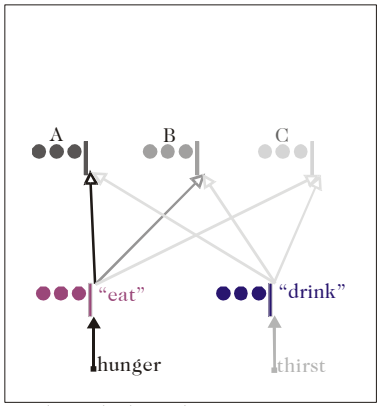
4. So, as well as back propogating some value from the consumator “eat” action, we also back-propogate the distances. In this scheme we blend a small amount of the bundle of tags



5. Over time, we get a better and better idea of the distances between action-tuples and consummatory actions. Here *A* gravitates towards *"eat"*, *B* towards *"drink"* and C remains close to both. *A* might be "chase animals", *B* "follow sound of river", and finally *C* "explore the environment".



7. Now we can have motivational state influence the expected value of action-tuples.



8. Through these distances we can modulate all the expected values in the system. Here, the creature is "hungry" rather than "thirsty"; therefore focus is placed throughout the action selection of the creature on only actions that are likely to result in "eating".

*With V_1.**D** representing the "distance" vector of V_1. This is initialized, on demand, to some default maximum value.*

More concretely, v_1.blendTo(v_2,α) translates to:

$$V_1.\texttt{blendTo}(V_2, \alpha) \Rightarrow V_1.\mathbf{D} \leftarrow (1\text{-}\alpha)V_1.\mathbf{D} + \alpha(V_2.\mathbf{D} + \mathbf{1})$$

Now we can begin to modulate these action-tuples values (specifically, whatever V.toDouble() is to return) based on these distances, and based on the instantaneous expected values of the sources.

Why might we do this? Consider a consummatory action-tuple

see food | eat food | until its gone, around 20s; V=100

A creature might learn all kinds of strategies that are good (and valuable) for getting food (hunting strategies, climbing trees, etc.). These action-tuples will, by this mechanism, become associated with the consummatory action "eat food" because they often *precede* food.

But all these things however are (at the moment) irrelevant, and not particularly valuable, if the creature is not hungry and we shouldn't select actions which accumulated their value by preceding food, if we have no drive for food.

This propagation is enabling us to modulate the whole behavior landscape very quickly, by changing the importance of a small number of goal consummatory actions. This is an important synthesis of what is typically considered one of the uses of motivational state in real creatures – allowing creatures to make relevant decisions quicker. Finally, we note that it gives us an alternative to creating hierarchical action-group trees, and can become an important component in more complex characters.

value signal processing

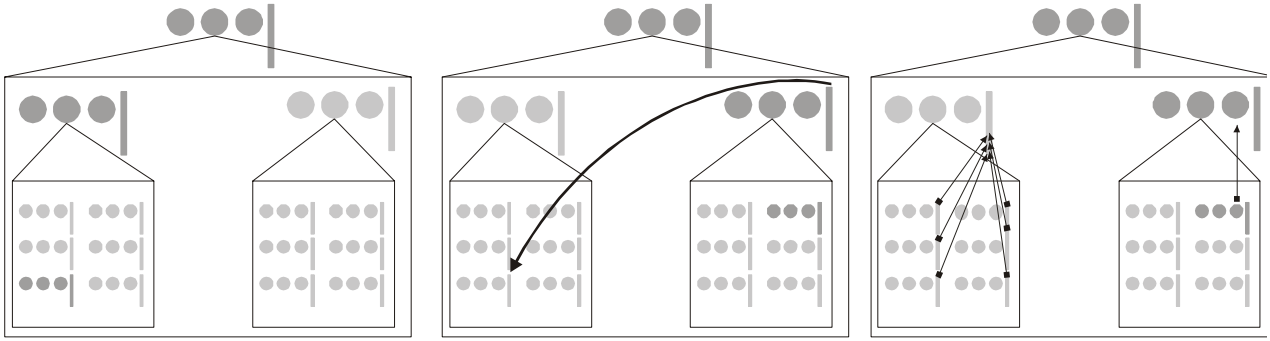
We can look at the V_1.`blendTo`(V_2,α) as an opportunity to perform arbitrary signal processing on V_1 with the contents of V_2. Looked at this way, the vanilla V implementation acts as a low pass filter. But there are other signal processing networks that we could have to capture other effects.

An example: "leaky" networks where value slowly leaks out of a V, perhaps reflecting a reduction in confidence (although there are better ways of representing this, see page 80). Other networks might provide momentum to the credit assignment process. Finally a combination of these ideas may be appropriate, where values are processed in such a way that they develop momentum towards previously *well established* values – which act as basins of attraction for the value of V.`toDouble()`. This style of processing should be linked to the contents of relevant percept models, perhaps taking us towards the learning and identification of different "situations" in the environment. Structures that tie these things together in this way have not yet been developed, but remain an interesting extension to the framework towards the understanding of long term effects of time.

other action-groups

hierarchical action-groups

It is important to note the fact that we can dress an action-group and its action-tuple selecting capabilities up as an action-tuple itself. These groups might have a traditional *Tr* but, while active, perform action selection on the action-tuples that they contain.



1. Hierarchical action selections can be made by unifying action-tuples and action-groups

2. Credit assignment can take place as shown above, to protect modularity

3. Values of parents are modulated by child values, *Du* conditions of parents and modulated by the active child

figure 14.schemes for hierarchical action-selection

Hierarchical action selection mechanisms are certainly not a new idea, (for rationale see [64,26,10]). But there are a few wrinkles, specific to this style of behavior system, that we need to sort out:

- **What should the value of these higher level action-tuples be?** I'm not sure there exists a single answer to this question, but suffice to note that there exists an option of modulating an action-tuple's value by the value of the action-tuples it contains.
- **How should credit assignment occur across groups?** Now that action-tuples can be in other groups, we face the possibility that action-tuple transitions can occur across groups. One note: we should assign only the credit to the leaf nodes, because this shields one group from the insides of another, but enable the proximity mechanism to have special effects.

  Can action-tuples be members of more than one action-group? Multiple membership seems a particularly attractive option – especially for the 'leaf' action-tuples in the tree. These diagrams are suggestive of the power of hierarchical action selection. But it's only a useful idea in practice if action-tuples can tell which group that made them active at any time.

  Indeed the action-group that becomes part of the activation context of the action-tuple – context which also includes information about *Tr*. This is related to the idea of a *pronome* – a place holder for a reference to an object, or a reason of interest [64]. In the diagrams above, the containing action-group references a particular object as the object on which the member action-tuples will operate. When behaviors can be parameterized in such a way, action-tuples can be reused and repurposed elsewhere.
- **How should action spawning take place?** In a static behavior system (where the topology of the action selection mechanism is not changing at run-time) this action reuse is useful. But in a dynamically expanding system, this reusability is all the more powerful.

Whole action-groups can now be spawned in response to the creation of new percepts.

- **What form should *Du* take for such action-groups?** *Du* for action-groups should take into account the temporal extent of the currently selected action. This does not rule out keeping timing information about the length that the whole group is active for, but the *Du* of the individual actions selected within should be allowed to percolated upwards in the tree.

perception as action

The complexity that `Percept` and `State-Object` offer is a fraction of what is required to model all perceptual phenomena. While the category discovery abilities of `Percept` seem solid, and the arbitrary signal processing of `StateObject` useful for simple interactions between stimuli, they cannot support multiple, persistent objects, nor can they mimic higher level perceptual artifacts such as priming, masking or a focus of attention. The simulation of priming requires that model, and preferably learn, complex relationships between the activation of percepts.

But we have already structures capable of learning such relationships – the action-tuple mechanism. The trick here is to build up simple, non mutually exclusive action-groups containing action-tuples whose activation signals a perceptual event, the act of perceiving something. Contained within these groups are action-tuples whose *Tr* looks to percepts as before, and then create a sub-tree of percepts that model those action-tuples' activities. The punch-line here: it's no accident that action-tuples and percept's interfaces and structures are very similar.

One thing that action-tuples have that percepts lack is their value. The *values* that these perception actions posses can be ascribed meaning – they are the values of perceiving these things. This notion can be grounded both by other perception actions and by the values of actions which result *from* perceiving them. These values can ground the "saliency" of

this perceptual event – in this way a zebra could learn that perceptions of lion are large negatively-salient perceptions[2].

In fact, in our discussion of the lion problem above, there were action-groups that became active in the presence of a certain class of object. But a specialised layer of intermediate action-tuples ought to provide focused modelling capabilities for simulating perceptual phenomena. As an example, for temporal perceptual effects *Du* percepts equipped with forward transition models discussed above (see page 58) can simulate and learn priming effects. Another is the correlated activity modelling – which might mimic *gestalt* phenomena.

Regardless, the bigger picture here is that the power that action-tuples have to represent action can also be used to model the world. Action *as* representation has appeared in a number of places in the literature (e.g. [59]) and here it falls out of the framework naturally. This helps ensure that as we build up more and more sophisticated perception abilities we are also building more sophisticated expectations about those perceptions. These expectations are vital to creating the illusion of intelligence in characters. It is towards this and related themes that we now turn.

other state – emotion and motivation

soft state

So far we have been building up more sophisticated ways of orchestrating long term patterns of activation. But we've said very little about emotion and motivation. These ideas are a very important part in modelling characters, not least of all because they form an important part of the inten-

2. This learning is perhaps best done (if we want to do it at all) in a separate micro-world. This required a lot of (worthwhile) engineering. For parts of a character to be taught separately and then *reloaded* into a more complete character, needed not only a persistent storage mechanism, but a way of automatically reconnecting parts into a greater whole.

tions that we ascribe to characters. Motivational state crosses, and hopefully unifies, many of the time-spans that we, as character authors, are interested in. The explanatory power of motivational state over behavior can work on long time-scales– say, characters take significant time to become hungry – only to acted *upon* by behavior over short time-scales – characters quickly get sated by eating food. Characters lacking either side of this interplay are not engaging or *readable* in the long term. Such motivational state is also, I believe, crucial to the production of long passages of music.

Of course, what is necessary for those that interact with our characters is also necessary for the creators of these characters. The brave, foolhardy or dogmatic might try to have motivation state emerge from a synthetic physiology to act upon behavior, or have the illusion of motivation emerge from the behavior itself with nothing underneath.

But motivational states are just too valuable to character authors. Here we choose to model motivational variables explicitly, typically with scalar values, and do that in almost an entirely *ad hoc* fashion. We use them directly to moderate and coordinate the long range behavioral change that we hope participants can read, and we modulate them directly by behavior closing their loop. The glue is a rapidly assembled layer of signal processing primitives – the same primitives used for `StateObject` interactions (page 39).

The two places where the results of these processing can act on the selection of actions – either by modulating the values of actions, or through directly contributing to the triggering of actions.

In the first, and more likely, category we can modulate the value of, say, “eat” consummatory actions based on the scalar value of “hunger”. Through the complex value mechanisms discussed above we can propagate this back to associated appetitive action-tuples. This connection need not be preset and fixed. It might be possible to learn the fact that eating reduces hunger – that when hungry, eating is a good strategy for

bringing the internal *milieu* back closer to an equilibrium point [85]. Whether we do this is largely dictated by where we choose to put the "ground truth" into our characters; whether we wish to encode the fact that food reduces hunger inside our behavior system or encode it somewhere else (eliminating this explicit statement is obviously impossible inside virtual domains).

In the second case *Tr* percepts can model the scalar motivational values directly. This technique can be used to create *scale-free* effects of motivational state – actions, although configured to react to high or low state variables, learn what high or low means in their particular context. This dramatically reduces the number of "magic numbers" present in the design and description of characters, or at the very least converts some of them into a more abstract form.

Both techniques – value modulation and percepts identifying and representing "hungry" – have been deployed in the installations discussed in this work, albeit it extremely simple ways. The response to motivational state of this new behavior system is still fluid. Now we need to look at how, when activated, action-tuples can perturb this state.

note hunger – "exchange (installation)" on page 185.

expectations about value

Although the action selection mechanisms described above require *V* implementations to provide a representation of expected value as a single number we've already seen that it can be augmented with extra information. But we can do more by adding statistical modelling of value into *V*. The goal is to create characters who can get surprised when something unexpected happens[3]. First we need a criterion to judge whether an action transition is out of the ordinary.

Recall that the credit assignment process performs the operation V.`blendTo`(V_n,α) which results in V being moved towards V_n (the value of the succeeding action-tuple) by an amount α. We can view this operation as a non-stationary model of V_n. This view fits with our previous implementations of V – the simplest non-stationary model of a single real

variable is exactly the low-pass, infinite-impulse response filter described earlier. In this view the filter constant α embodies our expectations concerning the time-scale over which the distribution of V_n will change.

The next simplest model that we might have of V_n is similar to the rolling buffer-model described above. Here we get not only the (moving) 'mean' of the distribution V_n but some measure of the variance of this distribution. This variance is what we need to be able to judge whether any specific V_n is exceptionally high or low. Given a Gaussian model we can obtain a *likelihood* of V_n:

$$p(V_n) \propto \exp(-(V_n - V_l)^2 / 2\sigma^2)$$

That is a quantity proportional to the probability of seeing such a V_n. If this probability is below a certain threshold then this value transition is "surprising" to the character.

But *how* surprising? To answer this question we need to keep information concerning our *confidence* that this model is in fact correct. The intuition here is two fold: firstly, if a model has been wrong recently, it should be less surprising if it is wrong again; secondly, if a model has not been tested for a long time, then it should be less surprising if it just happens to be wrong. We look back over a long history to help interpret the present.

Once we have built up a confidence concerning a particular action-tuple we can begin to use it for other purposes. The first candidate includes the modulation of the credit assignment process itself. For example, action-

3. Former colleague Chris Kline talks about expectation generation and violation in synthetic characters in his master's thesis [49]. The following two sections are very much a recasting of some of the mechanisms that he proposes. There are two differences to note. Firstly, his mechanisms can be incorporated right into the very fabric of the behavior system – including his object persistence modelling. Secondly here, there *is* an attempt at modelling some things about the expected results of a characters actions through the action-tuple.

tuples with high confidence *V* models should change slower than models that we have no confidence in. This is especially useful in handling recently created action-tuple's whose initial value we might be unsure of.

A second candidate includes changing the criteria for attempting a percept reorganisation. Recall that percepts associated with high-value action-tuples are given the opportunity to fission off smaller, more refined categories. It might make more sense to promote this percept tree growth in the cases where a high-confidence action-tuple suffers an expectation violation. This could be interpreted as a signal that something in the world has changed, and we need to refine our assumptions and descriptions. While this is not always the correct interpretation, it is often accurate in behavior shaping scenarios – where once a behavior is established through reward, the reward criteria is changed to provoke specialization of this behavior.

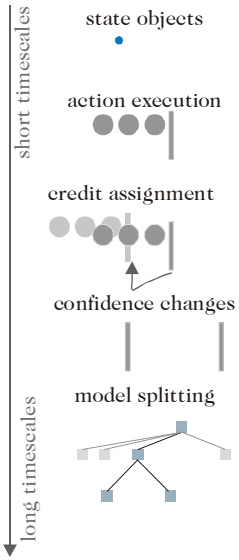


figure 15.timescales in the behavior system

expectations about the consequences of actions

But only part of a character's expectations concern the values of subsequent action-tuples. We have another way of generating expectations and consequently expectation violations – the *Du* part of any action-tuple. This description of the state of the world when the action should finish is consequently an expectation that the world will be that way when the action finishes.

Just as we could define a likelihood of a particular V_n above, we can interpret *Du* as an (unnormalized) likelihood rather simply:

$$p \propto 1\text{-}Du.\texttt{evaluate}(end\ time)$$

that is, the value of *Du* at the time that this action-tuple finishes.

Remember that *Du* invariably includes information about when the action-tuple ought to finish – so at the very least characters can get surprised when they are interrupted. With more complex models referenced by *Du*, more complex situations can be recognised by this mechanism –

for example, if we are modelling forward transition probabilities then surprise can be the result of a character being able to perform an unexpected action.

Taken together these expectation mechanisms are immensely important in the creation of music. It is a matter for future work to show that intelligent interactors must in themselves have representation of constraints and expectations. But who can deny that music arises out of a dynamic context containing constraints, expectations and conventions? When we move from a static to an interactive medium should not some of the these relationships be encoded *into* the fibre of the interaction itself?

emotional tags

We can also use the complex value techniques for emotional modelling. Here we associate *emotion tags* to action-tuples, providing extra dimensions of emotional information. Following on from the work of former colleague Soon-Yee Yoon [103] a full featured emotional space, with an appropriate number of dimensions, is the 'stance, valence and arousal' space – although subspaces of this space also make sense [33,78].

We can put stance tags into values in the dog training scenario. `StanceValue` is an implementation of V that propagates as above with the addition of a stance tag. `StanceValue` perturbs a character wide emotional state towards a particular stance state when such action-tuples become active.

For example, initially we can make the response to 'startle' one of high value and negative stance ("run away"). However, in clicker training, the click is a good predictor of the arrival of food, so the startle response accumulates 'positive stance'. We could couple this and the 'perception as action' (page 77) ideas together, and start associating emotional tags to objects that we perceive.

concluding remarks

What has been described above started with a few, very simple computational structures – the action-tuple, the percept, the state-object and the action-group. Inside the implementations of these structures we might find complexity but importantly these structures come together in the same way regardless. From the outside, we are building characters with a small number of parts which in turn have simple, well defined surfaces.

The behavior architecture described here has been tested in a number of interactive installations. A full training scenario, demonstrating clicker training, speech recognition and simple shaping on a dog was shown inside the Media Lab and at the Game Developers' Conference [35]. This installation served as both proof-of-technology and proof-of-concepts for many of the ideas already presented above.

The two frontiers for our behavior system research are now the 'very big' and the 'very small'. It is important that many of these ideas are put together into a large installation; and no doubt we will learn much by doing this. However our understanding of several of the elements could be augmented by closer study in well controlled experiments.

The remaining ideas (in particular, the 'soft state' work and many of the percept models) have been used to create the (small) installations detailed in the last chapter of this thesis, page 168.

figure 16.duncan,
and the shepherd

animation

This section concerns itself with the connection between the characters and the screen – in particular the design of the characters' *motor systems*.

To begin I present a brief overview of character animation – what it is, and typically how it can be made. The mode of manufacture is important in the discussions which follow, and this thesis focuses on one particular source, that of the 'hand crafted' animation.

Next I will consider the very simple motor system – one which can only play out such animations – and build up complexity over the course of this chapter. By the end I will be in a position to present the last motor system built for this work. This new kind of motor system seeks to solve, or to provide a platform for solving, many of the problems discovered and discussed along the way.

motor systems for synthetic characters

challenges for a character motor system

aesthetic choices

The motor system is responsible for connecting the behavior system to the body of a character and through that to 'outside world' – in our case that world is a virtual world which can then be rendered by a graphics system.

Typically synthetic characters have taken the form of animals or easily anthropomorphized figures. They have limbs and heads, they inhabit 3 dimensional spaces and they appear to be affected by physics similar to ours. This is an aesthetic choice, a choice explored further in this work, but for now we will take it as an assumption.

In short we are drawing on the conventions of cartoon, film, theatre and perhaps even dance – and not without good reasons. Such traditions are rich and well established for both the creators of interactive installations and the participants.

walking, shaking and moving your head

Let us now look at the simplest kinds of motor problems we wish our characters to be able to solve. The basic competencies of a motor system clearly include such 'simplicities' as:

- **simple gestures.** Characters should be able to perform actions given to them though animations from animators. If an animator makes an animation, a motor system ought to be able to play it faithfully.
- **smooth body movement**. Unrealistic and highly noticeable discontinuities are easily achieved in character animation - they should never happen.

- **gross body movement.** How do we get the character to move around the world? This might be as simple as deciding upon which animation to play in order to move in a particular direction; this might be as complex as a whole navigational sub-system. Preferably, higher level behaviors will never need intimate knowledge of navigation.
- **eye/head/body orientation.** Can the character appear to attend to things in the world?

Each of these competencies has a behavioral component that we can make arbitrarily complex. For example, moving around the virtual world might require involve learn spacial maps of the environment, collision detection / avoidance and other approaches to path planning. But without a motor system supporting such behavior it will never make it to the screen.

The first two of these basic competencies are achieved by the careful replaying of 'canned' animations; the remaining two call for something more complex. The task of moving around at the very least requires *coordination* of animations (e.g. `turn-left`, `move forward`, `move forward`, `stop`) to achieve a particular *goal* but ideally any solution requires finer control. But the last competence – orienting the head towards something – demands such fine control. Here we need the creation of a continuous output animation space from a finite amount of animation material.

...in the correct style...

But the problem is harder than this – any solution to the above problems has to keep the character *in character*. By creating a motor system that keeps especially close to the source example animations, one will for short periods of time be insured of success in this particular venture. But on timescales longer than the animations this illusion of life will break down as the character fails to interact correctly with the environment,

fails to attend to things properly and eventually begins to repeat the same animation over and over.

Further, the correct style may change over time, in particular with the characters internal state. A complex character may walk 'slowly', or perhaps 'sadly' or even 'hungrily'. If we cannot achieve these things, then our whole venture is doomed from the beginning. Therefore we add to the list, as a priority:

- **be expressive.** Whatever the character does it must be *in character*.

supporting parameterized behavior

From the behavior level discussion of 'shaping' (page 28) we know we have to produce parameterized motor actions. Clearly a scenario where the dog learns to shake its paw higher and higher will work the best if we can produce a continuous space of `shake-paw` animations covering a variety of heights. Again, if the motor system cannot show it, then there is little point in a behavior system modelling it. Simply stated:

- **parameterized motor actions.** Parameterized behaviors demand parameterized motor actions.

support new motor actions

One of the goals of the behavior system described above is to allow our characters to increase in complexity over time. This can often be achieved by reusing already present motor actions – for example a newly learned hypotheses:

verbal command 'beg'| `beg` |...

reuses the same motor action as:

whenever | `beg` |...

Learning quantitatively different `beg` parameters does not exhaust all the learning possibilities.

The leading example here is our complex shaping problem from the introduction: can we teach a dog to chase its tail by directing its attention with a training stick. After a teaching period the dog will chase its tail in the absence of the stick, presumably for a reward. Although machine-learning techniques have been applied to animation creation this problem demands *live* motor learning of effectively new animations – a task that has not, to my knowledge, been attempted in the literature before. Therefore:

- **create new animations 'live'.** Provide support for the perceptual models and the output mechanisms to enable characters to learn new animations during their lives.

motor level problem solving

This last point suggests opening up the 'contents' of the body to the in[tro]spection of the behavior system so that we can learn and form models of movement. But in many cases we'd like the communication between the two to occupy *less* bandwidth, not more. Experience in the creation of characters brings with it many insights into how the communication between the behavior and motor system should work.

The goal in this work is to make behavior→motor communication take place in terms of *desired pose* descriptions. This could be the behavior system saying "I'd like to walk over there now"; it could be in terms of *end-effectors* – "put my nose near the food. Is it there yet?"; finally it could be in terms of time – "how long, roughly, before I could get my nose there?". At the very least this has the very desirable effect of shielding the behavior of a character from changes in the competence and content of the motor system. So,

- **communicate in terms of end-effectors.** Can the behavior system communicate with the motor system in terms like "get my mouth to the bone"?

maintaining the illusion of (cartoon) physics

Since this is a virtual world, we are free to do anything we want to the position and angles of a characters body. However, this freedom is seldom what we desire. Instead, typically, we want characters to fall back to earth when they jump, characters' joints to resist turning complete rotations and characters feet should stay on the floor rather than under it. These things we don't get for free.

Further there are more subtle dynamical symptoms of physics which we would like to mimic – for example, once the dog gets its nose to the food, it might not be able to keep it there if, say, that pose involves standing up on its hind legs. Another example would appear if we could get a dog's nose to follow a stick – as the stick moves higher and further behind him the dog will have to sit down and eventually beg to get his nose near the end of the stick. All of this is due to [a transparent awareness of] the physical constraints of joint and muscle.

World physics might be made even more interdependent in a cartoon scenario (Wile E. Coyote looks to the camera before falling off a cliff). So, finally:

- **be able to simulate some kind of physics.** Can we do this without a full virtual physics simulation? Could we do this in an example based domain if we did?

finding your food

Another way of looking at these issues, together with many behavioral issues, is to consider a scenario: how does a dog find its bone. Just some of the interactions between motivational state, behavior and motor systems are outlined in figure 17. Its worth noting that while 'finding your food' seems like an easy scenario, none of the problems outlined in this picture are either particularly easy, or satisfactorily solved. This scenario, however, is useful as a measure of our success. Work continues inside the Synthetic Characters Group on various aspects of this problem not ad-

dressed in this thesis (in particular the navigational control). In this section we build characters that can walk around, in the next we'll attack some of the fine-grain motor issues necessary for the synthetic dog's mouth to ever accurately get hold of his bone.

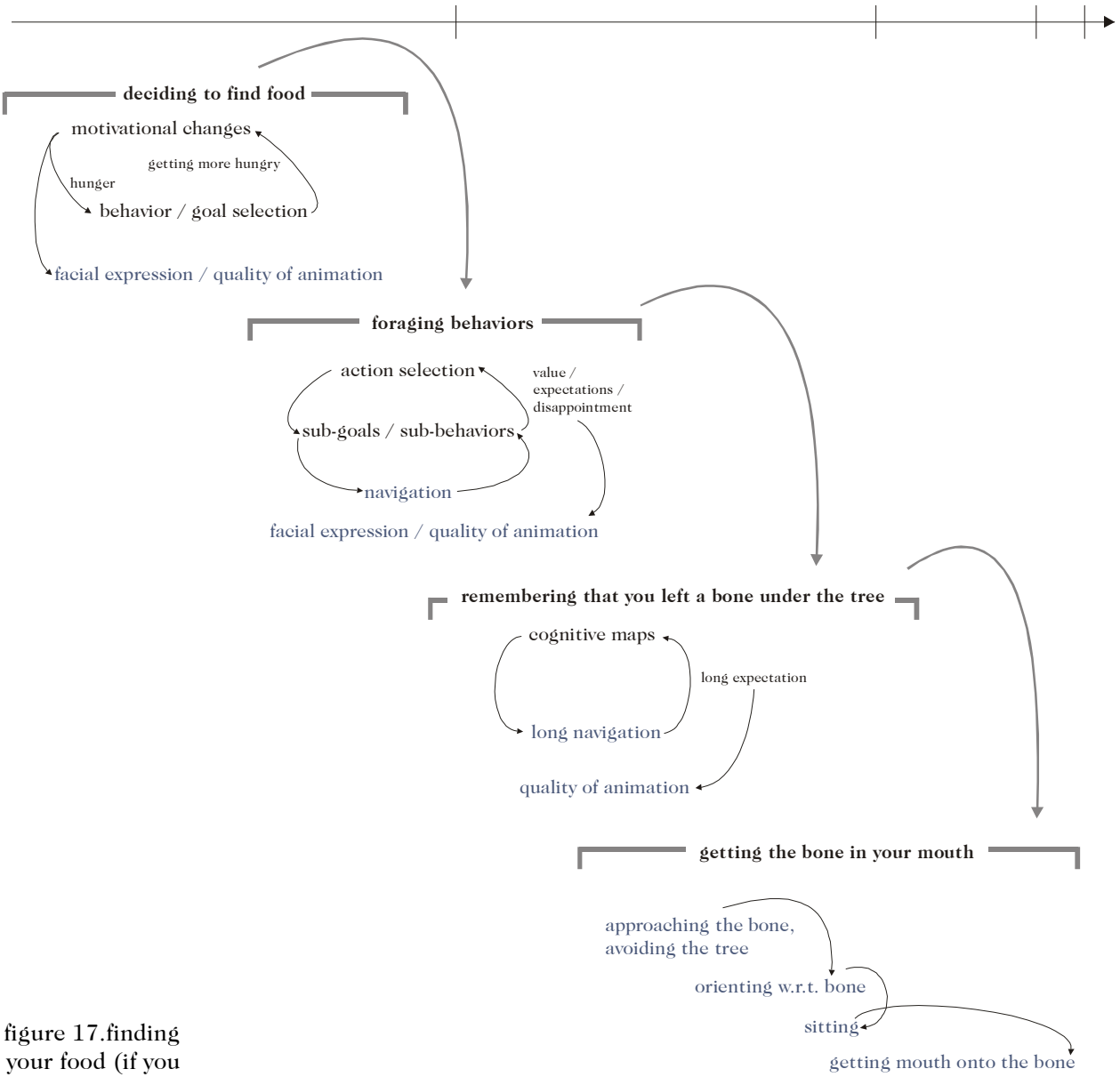


figure 17.finding your food (if you

example based character animation – an overview

Throughout the development above there has been the assumption that we can dynamically create animations, on demand. Before going on to look at the motor systems that we will use, let us stop and discuss how we are going to ground these animations.

The character is represented internally as an articulated figure. By a combination of a number of animated joint rotations and a single translation of the figure as a whole, we can conspire to make this figure come alive.

Many software packages exist to aid in the creation of suitable coherent animations. By far the most common technology at work in this field is the *key-frame animation* – animation is created by smoothly interpolating between a number of poses, placed at particular times.

The skeleton of a character, now animated in this way, goes on to affect positions of vertices connected to its bones. These vertices are connected to form triangles which are eventually rendered on screen. Typically a lot can be achieved by only animating joint *rotations* (rather than joint translations or scales) because typically, character's bones don't change length. In addition there is one translational degree-of-freedom governing the gross position of the creature.

In a fully general system however, other things could be given keys – most notably vertex positions or the details of the connection between bones and vertices. Regardless, this chapter will not concern itself with these things. Similarly, how the graphics system renders the skeleton or the flesh from a hierarchy of joint angles will not be discussed.

The target domain for these software packages is typically the 'finished film' – in precisely the same way as the target domain for a music sequencer is the 'finished score'. (Although recently the expanding market for 3D computer games has made this area of interest for such software developers, and other software has concentrated on computer assisted

choreography [23]). Regardless, the output of these packages is invariably a single, final animation – in short, a description of how joint angles and positions vary over time.

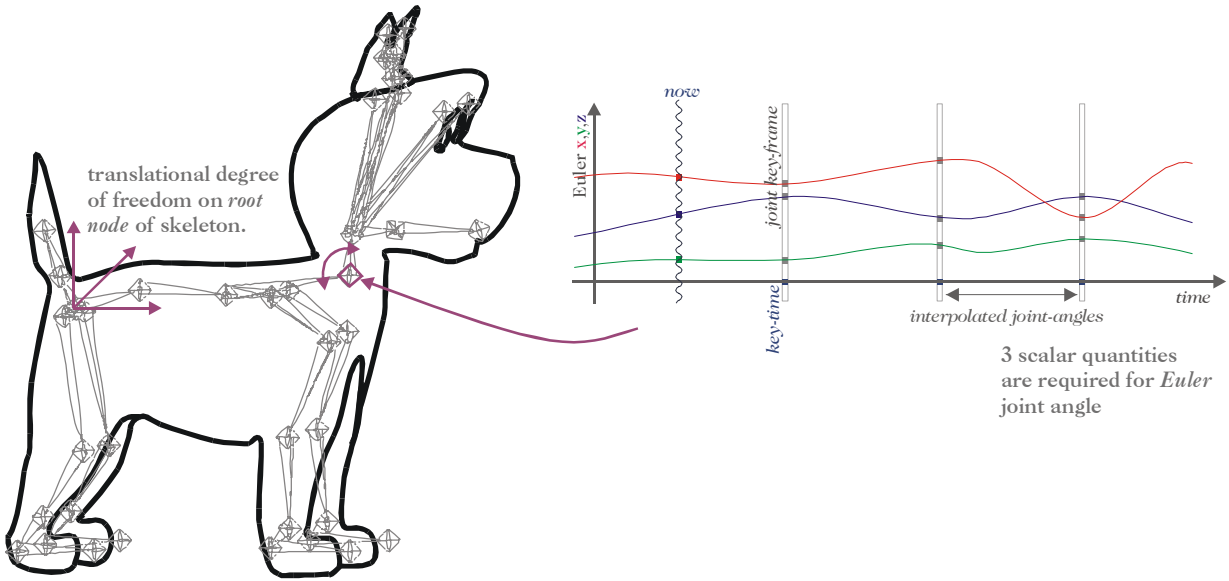


figure 18.key frame animation applied to a dog skeleton

sources of animation data

The hand crafted animation is by no means the only place from where we can obtain animation data. There are a vast range of procedural animation techniques. Strictly procedural approaches differ from the above in their lack of reliance on human animator guidance - although in reality there is a continuum between the procedural and the *example-based*. This continuum, of interest to us here, includes:

- **mathematics – inverse kinematic solutions**. Particularly useful for finding poses for use as key frames, 'inverse kinematics' (IK) seeks to solve the *inverse* problem[1] of finding a set of joint angles that result in a particular *end-effector* (e.g. finger or head) being in a specified position, or a specified position and orientation. Like many other inverse problems this is typically an under-specified problem (many solutions, or no solutions often exist) in a very ill-conditioned space. Still, robust solvers exist for this problem – robust in the sense that they generate a stable solution if such a solution exists locally – and they are now part of the standard animator's software packages. We will return to the solution of IK problems *inside* the character motor system shortly.
- **more mathematics – modulation**. Perlin [68] is another among those who also seek to build complete motor systems for interactive characters. As well as expressing a concern for the *composition* of example animations he also inbuilt the ability to modulate these animations by the application of band-limited random noise functions to joint angles. Sometimes the goal is to leverage the example-based domain, sometimes it is to use these noise functions to create animation material afresh. Either way, Perlin sees these functions as a

1. in contrast to the *forward kinematics* problem: given a complete set of coordinate system transformations along the joint chain, calculate where the end-effector ends up. This is a rather trivial problem to solve in all joint representations useful for computer graphics.

way to move beyond repetitive animation cycles that can become the downfall of example based animations (see, for example, the repetitive animations common in computer games).

- **physics**. There is a whole field of treating the body as a physics-based dynamical system – that of connected articulated masses, with joint properties. Such models seek to find an advantage in "automatically" obtaining realistic physics. Such simulations either require computer learning techniques to control the complex dynamics of such a system (say, through the application of forces and the modulation of impedances) or ideas from control theory to re-incorporate animation examples into their systems. Once we are in the physical controller realm we can *retarget* animation controllers from one body to another [40,36].

  Physics, however, is expensive to simulate and hard to control. Further, once you have a physics simulation, Newtonian physics isn't necessarily what you want. Designing a new physics or even a new creature's body in a physical simulation can be a daunting task. Once one has designed or learnt a controller that acts on a physics-simulation body you have problems with controller brittleness. Having fallen over can the controller get the model to stand up again [54]? Since one of the reasons to simulate physics to achieve high degrees of realism in collisions and unpredictable events its unfortunate to have to avoid such collisions.

- **computer learning**. The task of learning controllers for physics bodies opens up to many traditional computer learning ideas. Sims [84] seminal work uses genetic programming ideas to learn controllers (and morphology) for *interacting* creatures. Genetic algorithms also find purpose in the motor systems of another character creator – Terzopoulos learns controllers for simulated fish, in simulated water [96]. Mataric *et al*. have connectionist representations of controllers capable of learning to perform examples that seem well behaved between and away from stable attractors [6]. Brand learns pose and

transition models from examples in statistical work that will be particularly relevant to us later [12].

All these (successful) techniques share one feature – they require extensive off-line pre-training or genetic selection in well configured domains. While this is a perfectly useful way of solving the very real problems of realistic animation creation and modulation, such off-line techniques cannot be used for on-line learning of animation material.

a diversion - why example-based?

The amount of work related to creating, using, reusing, repurposing and modifying character animation is vast. The variety of literature is created by the variety of goals in the field. Many papers in the field open with a discussion of the exact same issues, weighing them differently they conclude that different approaches are more appropriate. Some take the cost of generating an original source animation to be very high, and seek to avoid it all together (e.g. the physics based approaches); others accept the need for expensive source material to bootstrap the process which is otherwise not data-driven (e.g. motion capture followed by animation manipulation). All these approaches, however, tend to see the work of the human animator (or motion capture system) as ending after they supply some data.

Our concerns often seem closer to that of the world of finished film – the animator, here, is king. The animator is not a necessary evil, nor should their rôle be confined to producing a few animations early on in the creation of a character. We seek to create and use techniques for creating new animations in real time not because creating animations is a drudge, but because we need to, given the interactive medium that we are working with.

Some of the interactive techniques discussed above seem to make problems that we will face later disappear. For example [6] makes it ‘easier’

to specify joint constraints and physics, because it is a model that is built with joint constraints and the physics of bodies.

The problem here becomes how to keep the animator as part of the process. To a Disney animator the task of creating a physics-based, learnt-controller walking character is a under-specified problem – what kind of walk? and why?

There is a parallel here with synthesis algorithms in music – a far older area than computer animation synthesis. Some synthesis algorithms are more popular then others. It is interesting to note that many of the successful algorithms for artificially producing sound started off life as compression techniques[2]. I speculate the reason for this is that such techniques (e.g. LPC encoding [53]) come complete with an inverse transform – with the ability to transform *existing* sounds into their internal representation (e.g. filter coefficients and residue). Other representations and synthesis techniques seek to do the same – for example cluster-weighted-modelling's data driven synthesis [81] – and we cannot ignore the commercial supremacy of wave-table based models. All these models can be described as example-based.

For other popular representations (e.g. FM synthesis [19]) the number of parameters is extremely small and explorable (usually from the need for computational simplicity). Even here there have been attempts to automatically express existing sounds in terms of these parameters [42] – attempts directly analogous to the machine learning of controllers for physics simulations. This is fuelled by a desire to let creators who create within these representations work out "where they are" and enables a method of working involving iteration between external and internal representations.

---

2. the phase vocoder (for musical applications, see [32]) was born of research into compression of voice signals for telephone communication. Linear Predicative Coding (LPC) has a similar history. Wavelet resynthesis was born out of the popularity of wavelet compression techniques [52].

The preference here for example-based motor systems arises from similar concerns. But in the case of animation the arguments are compounded by animators' talent for creating new animations from scratch. Raw procedural approaches to animation can fail to have 'inverse transforms' to allow this iteration. Procedural modification of example animations can suffer from the same problems – the inability to adequately express what it is you expect the output space to be. Subsequently, both fail to adequately leverage or rise to the challenge of a century of tradition of hand-crafted expressive animation. Therefore, we shall concern ourselves with the creation of example-based motor systems from now on.

blend based motor systems

building up a motor system

the animation player

A very simple 'motor system' is one which is given an animation from a menu of alternatives which it then plays it out on the body of the creature. For the sake of simplicity, let us say that it does this under explicit behavior system control – i.e. the behavior system is explicitly choosing what to play next.

Problems with this simple system become quickly apparent. Motor and behavior systems here are tightly coupled, and the behavior system is given plenty of scope to produce discontinuities in body pose – simply by changing its mind over which animation it wants to play out.

A traditional solution to this problem involves the *verb-graph*[3]. We can eliminate any chance of a discontinuity by doing two things: firstly running all animations that we play all the way through to the last frame and secondly, secondly the next animation out of the subset of the menu that happen to have a first frame that lines up with this last frame. All of this computation (working out possible transitions) can be executed before run-time. A directed graph can be formed with animation data as edges and matching first/last frames as vertices.

3. much of the concepts behind these blend-based motor system is articulated in [75].

For example, requests by the behavior system to 'play sit' while the motor system happens to be playing walk might result in a course being set through the remainder of walk and through all of walk_to_stand and all of stand_to_sit until we are in a position to execute sit. If the first and last frame of sit are identical then we can continue to loop sit until the behavior system gives the motor system another goal.

Paths through the verb graph from any vertex to any other vertex that are optimally short can be easily pre-computed (e.g. by the Floyd-Warshall algorithm [22]).

Several issues and peculiarities result from this style of motor system. Firstly, note that, in the example above, upon receiving the request to walk there is no initial change in the appearance of the character – in fact no change can occur until the remainder of this walk 'cycle' is complete. This puts an unnatural pressure on the animators (providing the animation content) to make the animations and cycles as short as possible. At this point one quickly discovers that many repetitions of short animation segments look like just that.

Secondly animators are also forced to spend time hand-crafting walk_to_stand and many, many others like it. As we extend the verb-

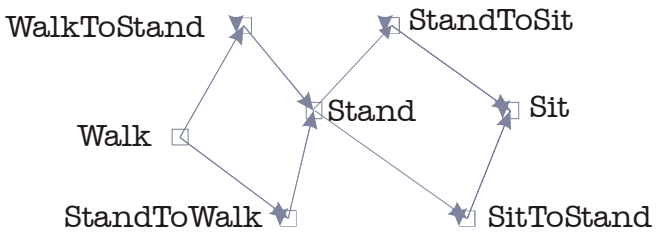


figure 19.a simple 'verb' graph of animations

graph formalism below, the task of making and maintaining these transitions grows to occupy much of the animators' time. Additionally, the creation and maintenance of the graph structure itself becomes a burden, and it is a burden that is hard to ease by automation given the intimate connection between the motor and behavior systems.

adding animation blending

Animation blending offers the chance to lighten the load on animation staff to produce transitions, to create far less repetition and far more plasticity in the output of the motor system. Animation data can of course be processed and combined in arbitrary ways before being put onto the character's body. In fact software products and motor system paradigms exist that explicitly encourage such manipulation within data-flow like languages [83].

Of course, only some operations make sense. A useful subset can be obtained if we constrain the space of operations that we perform to combinations of animations:

$$a_i(t) = \text{blend}\left\{a_i^1(t^1), a_i^2(t^2), \ldots, a_i^n(t^n); \beta^1, \beta^2, \ldots, \beta^{n-1}, \left(1 - \sum_{m=1}^{n-1} \beta^m\right)\right\}$$

with $a_i^1(t^1)$ representing animation data from source animation 1 on joint i at time t^1 .with weight β^1 The function "blend" is a combining function suitable for the underlying algebra of the representation of joint angles.

Three ways of blending seem particularly useful:

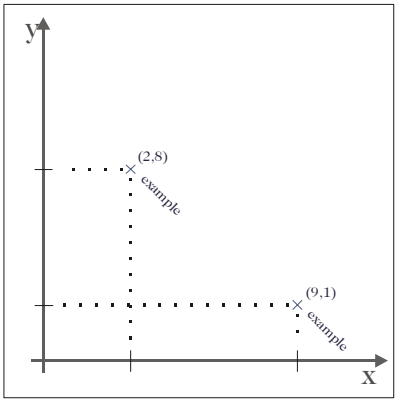
- **mutually exclusive layering.** For example, upper body motion from one animation (say, a hand wave) gets layered onto an animation that contains no upper body motion (say, just the legs walking).

  This might be considered a generation of the blend weights $\{\beta_i^n\}$ from spatial position of the joint.

- **general layering – *adverb parameters*.** Here animations that control the same joint angle degrees-of-freedom are blended, typically with different *blend-weights*, to produce the resultant animation. This style of blending has been termed the *verb-adverb* style of

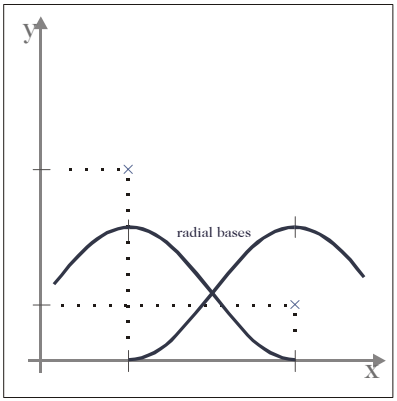
motor system [77] especially when the weights are used to blend between similar animations each in a different style, an animation of walking *sadly* and an animation of walking *happily*.

Here we generate blend weights $\{\beta_i^n\}$ that are independent of the spatial position of the joint. Instead we might generate these weights from higher level *adverb parameters*. These might be dic-

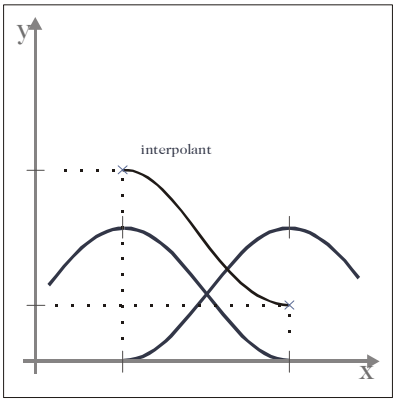


1. Consider the task of interpolating between two (one dimensional) examples "8" and "1" located on a single axis (at positions x="2" and x="9").

Radial basis function interpolation seeks to interpolate (that is, strictly go *through* the example points) examples by finding a weighted sum. The weights are given by a single *basis* function only, and this is a function of "distance from example".



2. If we choose a basis function with compact support and our examples are evenly spaced then we can easily make sure the interpolant does indeed go through the examples. We do this by ensuring that the basis functions go to zero before they reach the neighboring example. Here we use cosine basis functions (which have the added benefit of not needing normalized).

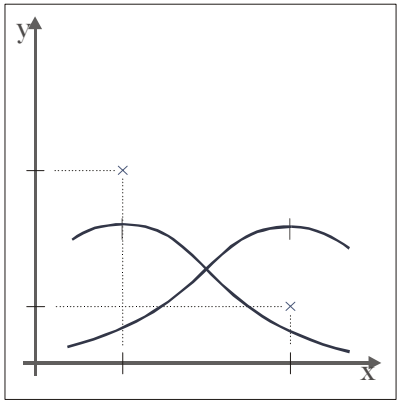


3. Now we can create the interpolant by performing the weighted blend. Note that the blend is not guaranteed to be linear (although, here, it is guaranteed to be symmetric).

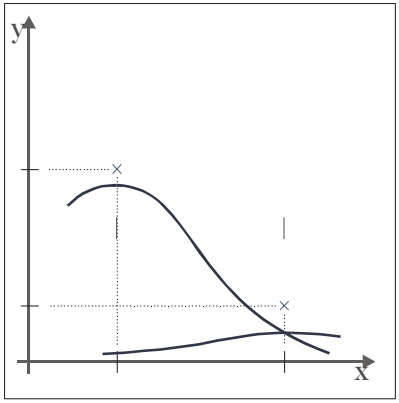
figure 20.radial basis functions for interpolation

tated directly by the behavior system – for example emotional parameters '*happiness*: -1 → 1'.

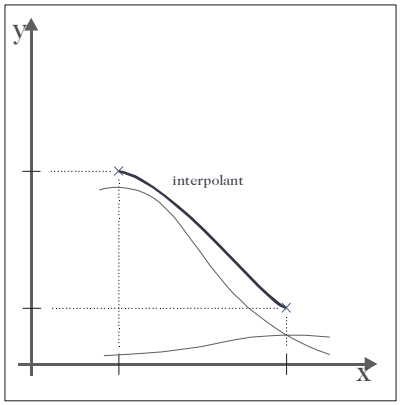
A number of subtleties must be present in any robust implementation. Firstly, care must be taken in this case to interpolate in a "neutral time". Consider the blending of those two animations `walk(happily)`, an animation lasting, say, 2 seconds and `walk(sadly)`, an slower animation lasting a full 4 seconds per cycle.



4. Sometimes our examples will not be evenly distributed, or we might desire basis functions without compact support.



5. In this case we need to work out what scaling of the basis functions will make the interpolant go through the examples. This is equivalent to solving a linear equation.



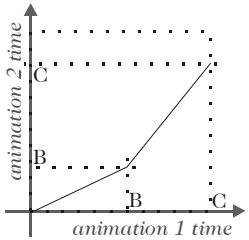
Clearly a 50%:50% blend of the two ought to last 3 seconds, and while we are half the way through this generated animation we should be blending data from 1 second into `walk(happily)` and 2 seconds into `walk(sadly)`.

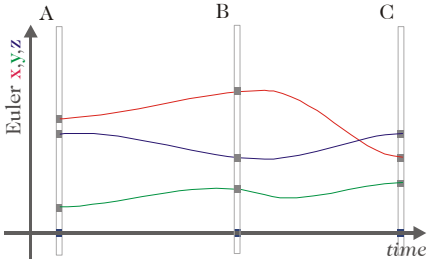
For higher accuracy we can define *correspondence-points* other than the start and end frames of the source animations that ought to line up (for example, the times that feet touch and leave the ground). Using these points we can perform inverse piece-wise linear time-warping on the source animations to take them into a normalized time and then a blended forward timewarp back to real time for output.

Also care must be taken not to move through this blend space too fast. A naïve implementation will have temporal aliasing problems (animations will slow down, speed up or 'hitch' if we are blending between animations of different speeds) – although this problem can be avoided. Any implementation however will produce joint position and joint velocity artifacts that will be very noticeable if the blend weights change on or above the timescales of change inside the source animation.

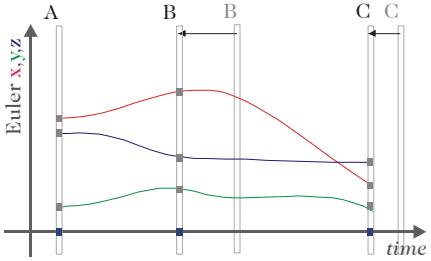
Regardless, this approach is very useful in producing animations on-the-fly. Later we will see how the automatic generation of adverb parameters can be used to solve a variety of motor system level problems (see page 114.)

we can describe
the warping of time
be a piece-wise linear
function

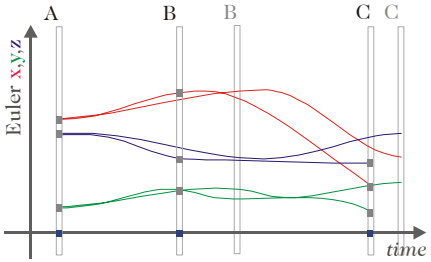




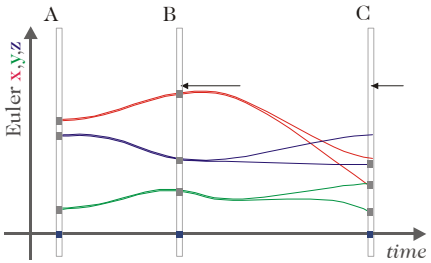
animation 1



animation 2



directly overlayed,
a blend would produce
artifacts



after timewarping
into a neutral time
a blend makes much
more sense

figure 21.time-
warping necessary
for blending

- **temporal blending**. Finally we can generate $\{\beta_i^n\}$ based on time – akin to a crossfade in audio. Similar to such a crossfade we might pick a single blending function $f([t\text{-}t_{start}]/[t_{end}\text{-}t_{start}])$ to govern the progress of the crossfade.

  If this works (and it often does) then these crossfades can take the place of the hand crafted transition. Indeed, if we are especially courageous, then whole verb graph structure can be dissolved. Instead we rely on these crossfades to take us from whatever position (and velocity) the joints happen to be at to wherever the behavior system wants to go.

  Blends like this have little to do with original source data, and therefore fail to maintain physical plausibility in many cases. A crossfade generated between arbitrary poses, travelling at arbitrary velocities by an arbitrary blend function, while guaranteed continuous, is not guaranteed plausible.

  Other implementation choices include: the length of the crossfade – perhaps generated by an heuristic look at the initial and target poses; weather or not to start playing the destination animation during the blend or at the end of the blend; the choice of interpolation technique – a linear interpolation (effectively reparameterized by the blend function), or perhaps a cubic spline of some sort, etc.

  Other, far more sophisticated approaches exist to the automatic creation of transitions between poses than a simple blend [76], however they are all too computationally expensive to compute on demand. One possible (and as yet unexplored) solution might be a hybrid approach, where transitions are automatically generated, off-line, and inserted into the graph. These transitions would contain data for the joint angles involved in the animations they link, and temporal blending could care of the remaining supporting joint angles not present in the animation.

using adverb parameters to solve problems

a baseline motor system - so far so good...

Implementing a motor system that contains all of the above concepts is a somewhat involved task. Not all of the styles of animation blending fit well with each other. It is not, for example, clear how to maintain a verb transition graph with heterogeneous adverb parameters. Eventually one ends up relying on the temporal blending to smooth out the seams, or the patience of animators to supply a large number of example materials.

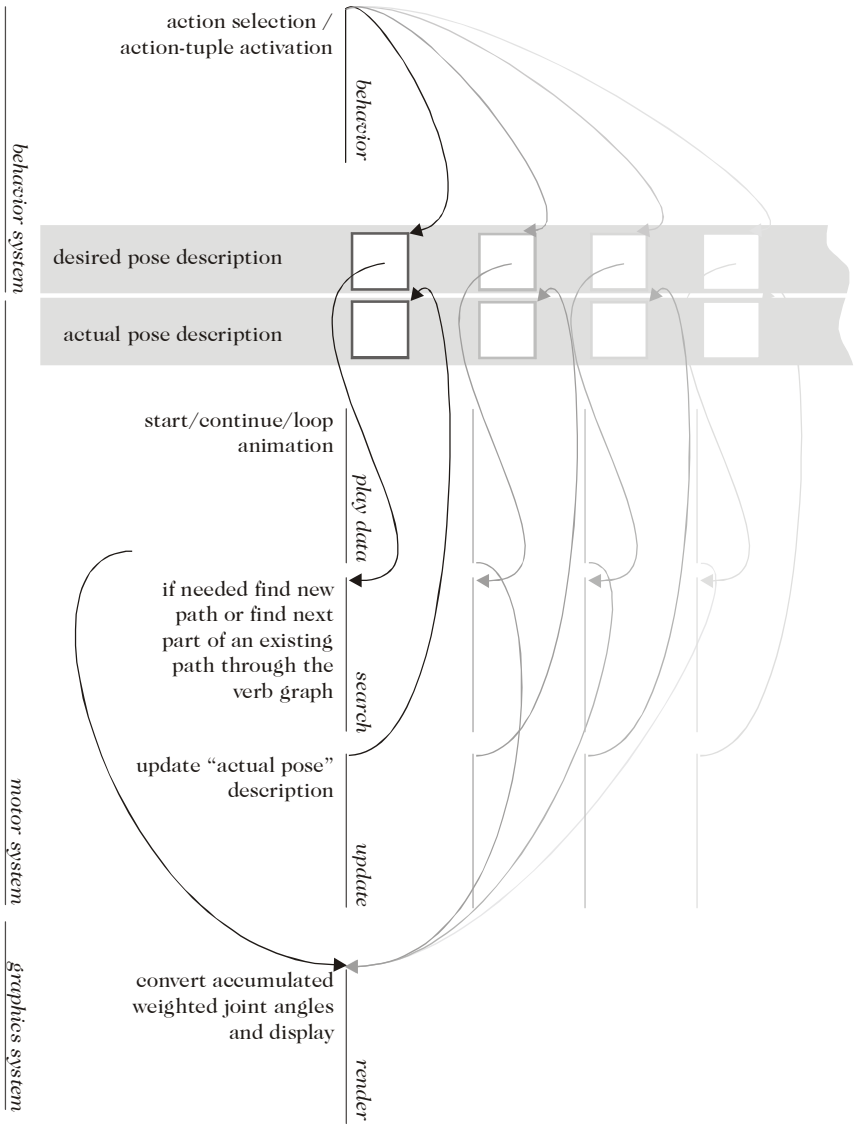


figure 22.flow of command in a blend based motor system

Creating a motor system that can layer animations on top of one another, produces the ability to do more than one thing at once, allowing different active parts of a behavior system to tell the motor system different things at the same time. Sometimes this is benign – walking and waving – sometimes it isn't – walking and sitting for example. Ideally, such clearly conflicting 'goals' should never arise from a well written behavior system. Demanding this from a behavior system confines it to communicating one command over perhaps several orthogonal channels – both an unreasonable limitation in the longer term, and an unreasonable 'leakage' of motor system structure into the structure of the behavior system.

The simplest, and harshest, way around this problem is to ignore the latter of two mutually incompatible commands – if the creature is actively sitting then it cannot begin to actively stand up as well. Mutually incompatible animations can be detected from the simultaneous presence of animation data for the same joints in animations, or particular joints can be labelled as 'necessary' for an animation to run. But the bottom line is that the behavior system has to have access to the current state of the body of its character, and behaviors must be prepared to factor the current body pose into description into their decision making process.

in joint representation blending & processing

As discussed above the simplest representation for the body that we might use for the motor system is the same representation that is used in a graphics system: that of the initial tree of coordinate systems affected by whatever joint rotations we have applied to them. As parts of the motor system write joint angles into the tree, previous joint angles are overwritten. After the motor system is complete, the graphics system is left to turn this tree into geometry, and then pixels.

Much of the need for the mutual exclusion detailed above comes from problems created when motor systems write into this tree more than once before the graphics system renders a frame – obliterating the data

that was there. This might be seen as purely an engineering problem, but things can be simplified if we give joints in the scene graph the ability to preform blending on all the data incoming each tick.

Two things fall out of this. One is the ability to weight blended actions, that is to accumulate a collection of joint angles and their weights each frame and perform the averaging just prior to the rendering stage. The normalization occurs automatically, thus simplifying the design of the motor system considerably (no direct lateral communication between concurrently active animations) and enables animations to ease in, ease out, or mask out less relevant parts of their data, fine tuning their blending behavior.

Second is the ability to collect statistics inside the joints themselves and act upon these statistics. For example, `sit` with weight 50 can ask all the joints that it really needs for their recent average total blend weight. Upon discovering the blend weights of 100 (put their by an ongoing `walk` animation) it knows that it shouldn't even try to enter the fray. If previously active, it can fade out its involvement in the joints and shut down gracefully.

Finally, we might be tempted to introduce arbitrary signal processing here. For example, by feeding back a low weighted amount of the previous joint angle back into the joint, we can obtain some of what temporal blending did for us earlier. Using this mechanism, and making newly active actions 'fade in' their blend weights we can always exchange a zeroth order discontinuity for a first-order velocity artifact.

Other kinds of signal processing can be conducted at this level[4] (e.g. [17] or for emotional manipulation of existing animations, see [97]) but

4. one further thing to note: joint angles are not quantities that are well expressed in simple vector spaces, so traditional signal processing ideas carry over but the engineering doesn't. This work is conducted using *quaternions* (page 215) as the representation for joint angles. Multi-target quaternion interpolation in these spaces is the work of colleague Michael Patrick Johnson [47].

there is a *caveat*: we run the risk of violating physical constraints. Since low-pass filtered feet slide when they walk, we know that we cannot globally apply such signal processing at such a low level.

simple shaping

But even without these extensions we can, using only the forms of blending already discussed, begin to solve some of the problems stated in the introduction. It is here that we begin to put together behavioral elements, in particular the *action* payload of action-tuples, and percepts and percept models associated with them.

Take, by way of an example, a one dimensional adverb space. We can construct a mapping from such a space to an arbitrary number of example animations using a radial basis function style approach (see page 104). If given, say, `beg(low)` and `beg(high)` example animations, this one axis represents a quantity h' that is related[5] to the height of the beg. Using this axis we can create new animations on-the-fly, `beg(`h'`)`.

Putting this access under behavioral control, we can reuse those same statistical models that have been built for *Tr* percepts to model this parameter, and place these models into percepts for *Ac*.

The simplest approach continues along this strategy: when this action becomes active we can *sample* this percept thus generating a target $\hat{h}$. We can send this to the motor system along with the command `beg`. We then feed this model with $\hat{h}$ itself, and of course the value of the next active behavior. Then, just as when *Tr* percepts were being updated, we begin to form a model of reward value versus $\hat{h}$.

Choice of model is, of course, critical to the success of this venture. A good choice in practice is a simple rolling second order statistical model, coupled with a simple Gaussian prior. This model offers an opportunity

5. although this relationship is not necessarily a linear (or even a one-to-one) mapping. A way around this is discussed on page 116.

to specify an initial tendency through the prior, and control the plasticity / rate of forgetting, through the rolling buffer length. While this prior is specified in terms of $\hat{h}$ rather than h, any non-linearities present in this mapping do not affect the success of the technique.

replacing inverse-kinematics

We can extend this approach to mimic the functionality supplied by inverse kinematics based solutions. The example here is of a *shepherd* character with the ability to point with a stick. Given such a long linear chain of rigid coordinate systems – end of the stick, start of the stick, hand, wrist, elbow, shoulder, spine, pelvis – we would seem to be on classic inverse kinematics solving territory.

Such solvers are typically no more than constrained, non-linear optimisers that exploit the locally smooth properties of chains of coordinate system transforms. A solution from such a generic mathematical technique will not be particularly lifelike in this case. For example, we will not see a shift in lower body stance to compensate for an extension of the stick from such a solver.

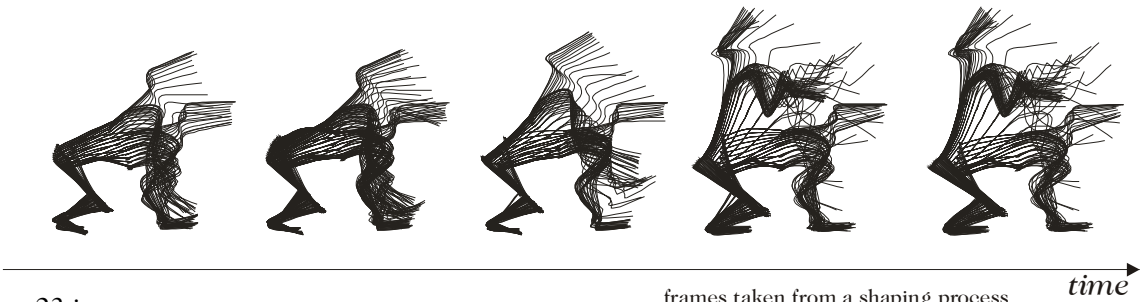


figure 23.increasingly high "beg" models (skeleton only) for dog

frames taken from a shaping process on dog (rewarding for higher "begs")

Instead we revisit the technique used for simple shaping, the generation of adverb parameters. Here given a nine example space of almost static poses arranged to span the typical range of stick movement, we can map desired stick position x (a real space 3-vector) to $\underline{\chi}$ (a sum-one, 9-vector). We do this again using the RBF style mapping. This can, with good example material, capture far better the subtleties of weight balance.

learning to replace inverse-kinematics

One might think that the above technology is all you need to have characters walk smoothly around the world. By blending together examples of walking in a variety of directions (say, `walk_turning_left`, `walk_forward` and `walk_turning_right`) along a similar one-dimensional space characters can turn arbitrary amounts within this space in order to get places. The blending for this problem works pretty well – both in our blend-based motor system (and in [76]).

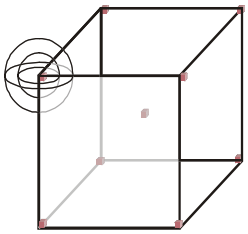


figure 24.arrangement of basis functions in 3 space

For walking, this must be augmented with a servo mechanism to insure that we actually get there. This is because the mapping between internal adverb parameter and external direction is not necessarily accurate. Just as h' does not correspond to h in the shaping of `beg`, the desired turning velocity and the actual velocity of `walk_turning`(ω) will not be the same. To make this more accurate we would have to work through the mathematics. Tying this and the engineering inextricably to the specific source animations is not an easy or scalable approach.

A better approach is to learn the mapping ([76] discusses this outside the context of behavior). And we can do this while maintaining a viable, interactive creature. Here we perform a very similar trick to the `beg` shaping. Here, however, we construct a lower level action-tuple that monitors the skill in question and filters the incoming parameter from higher up in the behavior system.

An example of this at work is the task of learning how to look at something; how, that is, to map a desired pair of polar body coordinates (θ, ϕ)

to a two dimensional adverb parameter space perhaps filled by a five-spot pattern of examples. There, blend weights are provided by a similar radial basis function approach, and the model assumes that no gross body movement is needed to achieve the goal pose. The model we choose for this 2-dimensional to 2-dimensional mapping problem is the *self-organising map* (SOM)[51]. This map has a number of properties that suit it well to the task, not least of all the ability to provide the good initial guess - that the mapping between (θ, ϕ) and adverb parameters are linear.

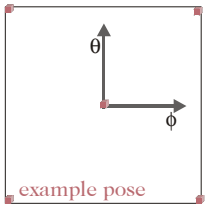
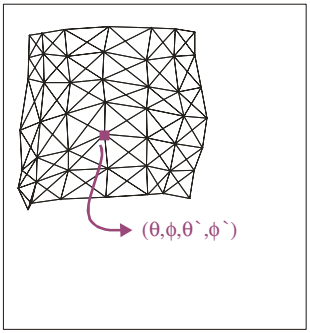
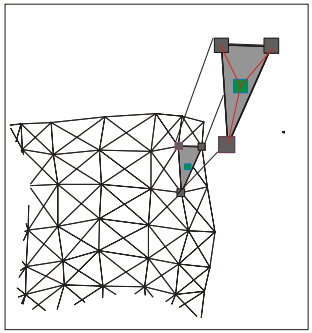


figure 25.a five-spot basis function pattern for head interpolation

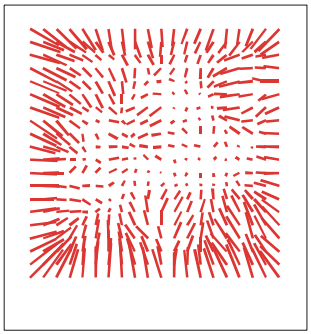
Another approach involves carrying out this learning in a *micro-world* where the dog concentrates solely on the task of learning to look at things. After we have converged sufficiently on a solution, this ability is save out into persistent storage to be reused as part of more complex creatures (with the same bodies). This is largely an engineering task, but it pays engineering dividends. Both approaches have been implemented and support convincing looking behavior.



1. For learning the mapping between desired head orientation (θ,ϕ) and adverb parameter (θ`,ϕ`) we use a *self-organising map*.

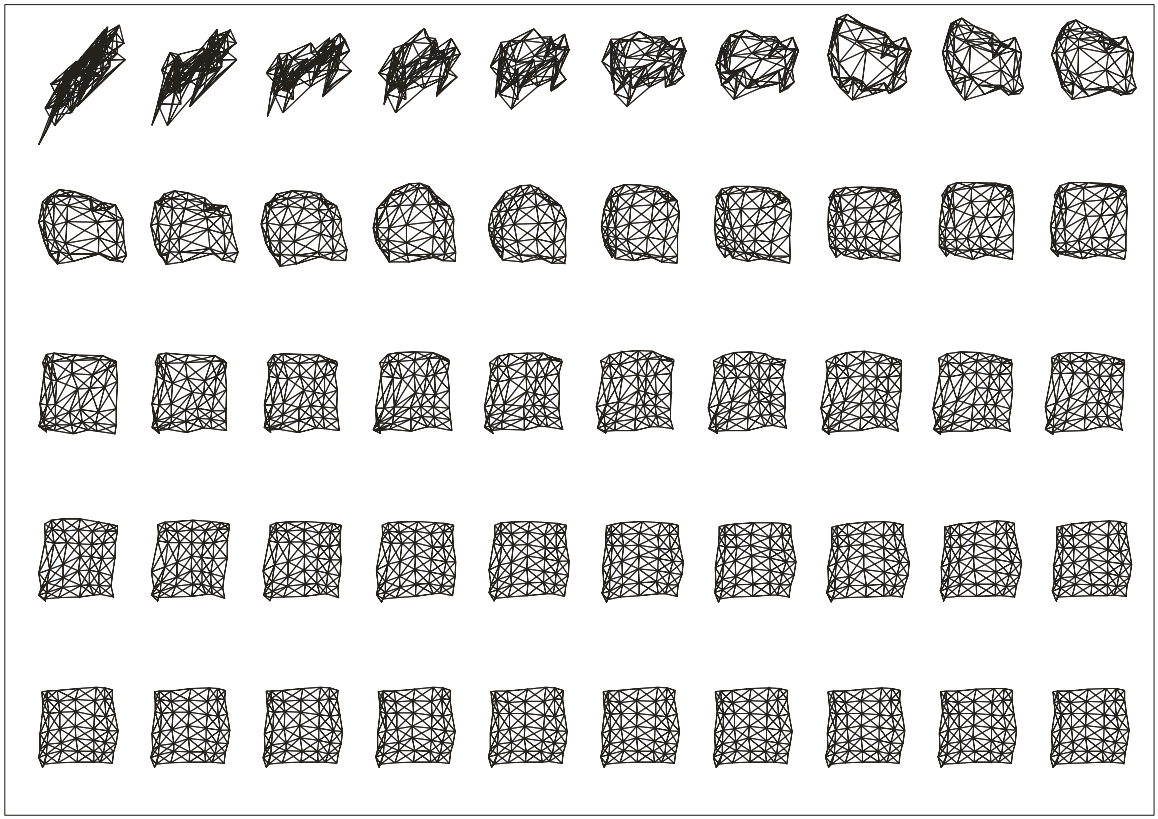


2. By modeling the (2d) mapping surface by a collection of (4d) nodes we can, through interpolation, approximate the function that maps between these coordinate systems



3. Above we show a vector field between (θ,ϕ) and (θ`,ϕ`). Note that the mapping is weakly non-linear but highly dependent on the details of the source animations and the blending strategy.

figure 26.SOM learning to look at objects



3. SOMs come with a learning strategy that can refine initial approximations or initially randomized networks

For, given this structure, our test dog was able to refine this initial guess down to a more accurate mapping that corrects for the superposition of several non-linearities (the basis function interpolation and the multi-target quaternion interpolation). Even better, it was also capable of learning how to look at things despite a variety of incorrect pieces of linear algebra surrounding the algorithm. This code was responsible for generating the (θ, ϕ) that the blend actually obtains, and is sensitive to the graphical model of the character[6]. This might be an good example of learning making the task of engineering a system easier.

But we can push this idea further and have creatures that not only learn what parameters (adverbs) to send to the motor system but which animation to play (verb). What we have to build here is a model of what spacial domain the motor system maps an adverb range onto given a particular verb. In one angular dimension (the only case that has been studied so far) we can learn to choose between 'tight turns', 'turns' and 'walk forwards' in order to turn a particular angle.

We achieve this by introducing behavior system structures into the motor system itself and building a family of action-tuples, each triggered in part by the requirement to turn, but also triggered each by a percept modelling and learning the range that the turn produces. Then a description 'turn 60 degrees' triggers the action-tuple that most likely can handle such a request. While this idea has only been verified in the simplest case, it is hoped that techniques like these can eliminate the complex coupling between animation code and animation material that develops while building a character and that can become a barrier to change and development.

where does this leave us?

The three experiments above show that we can use adverb parameters to solve motor problems. We can do this either in domains where the inex-

6. For example, the coordinate system of the head and of the root of the joint tree.

act mapping from adverb space to solution space isn't a problem, or we can learn our way around that problem, and do this within the context of a viable character. What none of the above experiments provide is any guidance as to how one should *move around* that adverb space to generate movement.

Again, in some domains this is not a problem. In the shepherd stick control domain, movement was being generated by an actual human participant moving a stick (primitive motion capture if you will) – the chances of generating implausible motion for a humanoid character are therefore diminished. In the shaping domain, we never move around the adverb space during the course of an animation, so this never becomes an issue.

But in other domains, having no guidance as to how to change adverb parameters becomes a problem. Since we are creating animations by blending static poses, we automatically violate the condition that the adverb parameter should change slower than the animation data it governs. The pragmatic solution is to filter the desired head position adverb parameter down into the actual head position adverb parameter. One can choose a filter and edit filter time constants, or one might choose to velocity limit, etc. One might go as far as to start changing these filter constants based on high level emotional parameters (high *valence* emotional states correspond to short time constants and fast movement and perhaps even under-damped oscillatory movement). Then to this *ad hoc* approach we need to add eye / head coordination for there is no way that this can be automatically regenerated in this approach.

These techniques might look acceptable (and they do), but they make me nervous. They are taking us away from the example based nature of this program, away from the data and thus away from the animators' skill. Worse, they take us closer to the control theory used in robotics, and a 'robotic look' is certainly not what we are trying to achieve. Perhaps we might back project example animations onto our adverb space and learning from these animation how to move around the space?

A second prominent issue concerning this style of motor system comes from the temporal blending techniques. While we use correspondence points to help adverb blending, we need to be able to supply similar guidance for blending transitions. Consider blending from a walk animation into a run animation – specifically, a blend from half way through a walk animation into the start of a run animation. These two animations are at this point exactly out of phase, it is easy to see that a crossfade could produce arbitrarily unrealistic motion.

Thirdly, all of the forms of blending violate other physical constraints – this is usually most noticeable on the feet. During temporal blending feet slide and feet can go through the floor. *Ad hoc* approaches to solving this do not seem particularly robust in practice. We'll see a closer analysis of, and a limited solution to, this problem later (page 147) but the integration of general physical constraints into a blend based motor system remains an open area of research.

Fourthly, although we have an approach to the problem of simple shaping, no progress has been made towards the more complex problem of learning new animations.

I feel the solution to all these issues lies beyond the scope of a blind, example blending paradigm. Instead we must actually pay attention to the example animation data.

graph-based motor systems

overview

So far we have manipulated the data (by blending, or layering) blindly. To get any closer to solving our motor problems we need algorithms that actually look at the animation material.

To this end a completely new motor system was built, based on a different representation of animation data. The approach here is to take source animation material, break it up into fragments, or *nodes*, annotate and associate pre-analysis information with each node, and re-form animations in real-time by stitching nodes back together again just-in-time.

These nodes, termed here `BodyPose`, are connected together in tangled structures called *directed, weighted graphs*. Forming a such a graph of such simple nodes could be seen as an extension of the original verb-graph idea into far higher resolution domains. Considered like this it is a logical way of decreasing the size of the atoms of a motor system. The resulting representations are also similar to structures found in gesture modelling techniques. This is no accident. An approach to the complex shaping problem will require a synthesis of animation production and analysis techniques.

technical development

The contents of `BodyPose` will determine the kinds of manipulations that we can do with them. We'll want a representation that will let us re-generate animations; a representation from which we can generate useful weights (or here, distances) between nodes; and we'd like a representation that lets us automatically work out what node should be connected to which.

In this project we are mindful of the constraints imposed by the need for real-time interaction with our characters. These constraints are very re-

al, and they limit the kinds of analysis of the data that we can hope to achieve at run-time. It is true that for us computer memory is far more abundant than computer time. Time that we can save by pre-analysing the data at the expense of the size of BodyPose can be time truly saved. If, that is, the pre-analysis burden is distributed throughout the representation. If it is not, and global computation is required that scales with the size of the representation, then we run the risk of producing a structure that cannot be extended at run-time, and thus cannot support the learning of new material at run-time.

building up the node representation

So, what should go into this BodyPose? The first ingredient is a snapshot of joint angles. But will that be sufficient to enable good graph topologies to be generated automatically?

Consider that many animations that we encounter are *biphasic*. For example, think of a hand waving. The hand covers the same positions and joint angles twice during the animation, first one way, then the other:

$$\ldots \rightarrow \text{left} \rightarrow \text{center} \rightarrow \text{right} \rightarrow \text{center} \rightarrow \text{left} \rightarrow \ldots$$

The two nodes labelled 'center' are not the same and should not be confusable. Their difference is reflected in the velocity (of the joint), therefore we include velocity information in BodyPose.

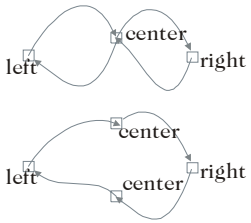


figure 27.the wrong and right segmentation of a biphasic 'wave' gesture

The second complication concerns the timing of poses. One thing we would like to do is to recover the timing of a source animation. Should we happen to choose a path through the graph that goes along all the nodes from an original animation, we would like to recover the timing (i.e. the relative times of the key-frames) of that source animation.

The easiest way to do this is to include this information in BodyPose. To do this we include both a time and something which identifies which animation produced this node. When moving between nodes from other animations we have to guess how long this should take. Here the velocity

information helps - we can make an educated guess based on the joint angles and velocities at the two nodes.

Together then, the primary representation in these motor systems is the directed, weighted graph made up of `BodyPose`'s for nodes each consisting of:

- **a complete set of joint angles (*unit* quaternions[7]).** We use these at run-time as essentially key-frames to interpolate between.
- **a complete set of joint velocities (quaternions).** We can also use these to make the interpolation smoother by using a higher order interpolant.
- **a time and source-labelling.** This notes where this `BodyPose` came from and when. Storing this information will enable us to do a better job of getting the timing right between key-frames.
- **optionally labels, translational information (position) and precalculated world-space positions of joints.** We will see a use for these annotations later.

To form a directed, weighted graph from these nodes, given some source animations, we must define a distance metric that produces the weights for the graph edges and a scheme for creating the edges between nodes.

developing the distance metrics

On a directed, weighted graph we can define two distance metrics that give meaning to this 'length'. The first is the distance (or weight) between two nodes that share an edge. The second is the distance between any two nodes, connected or otherwise. Desirable paths to follow minimize accumulated distances along edges (first metric, the *intrinsic* metric), and it makes sense while trying to find paths to go first along edges

7. quaternions are our representation for joint angles. For more information, page 215.

that take you 'closer' to the goal pose (second metric, the *extrinsic* metric). Both metrics will come into play when we start to decide how to move around the graph.

Some of the intuition that goes into the metric is the same as the intuition that helped us pick what was to go into the representation:

- **pose difference.** It makes sense that the more different two joint angle configurations are, the further apart their respective `BodyPose`'s ought to be.

  In a quaternion joint implementation we can break down the pose into joints j and start with a term:

$$\sum_j \text{acos}\,(1\text{-}(q_1^j \cdot q_2^j))$$

- **acceleration.** We ought to include the velocity information into the distance metric – transitions between poses that require little acceleration to complete should be favoured. That is, there should be no penalty for continuing to head in the same direction.

  We can capture this by taking the average of all (absolute) joint-joint *transition times*. Consider a path that begins at q_1^j, and moves along the velocity tangent (e.g. reaches a point v_1^j in unit length). How far along this path (e.g. what multiple of this length) do we come to the point that is closest to q_2^j? When we get there, how far are we away from q_2^j?

  The first of these questions can be answered by the formula below (see page 215). We travel α way along this line, with:

$$\alpha = \frac{1}{\phi}\text{atan}\left(\frac{v_1 \cdot q_2}{q_1 \cdot q_2}\text{-}q_1 \cdot v_1\right)$$

with $\phi = \text{acos}(q_1 \cdot v_1)$. The second, from the distance metric given above:

$$\sum \text{acos}((v_1 q_1^{-1})^{\alpha} q_1 \cdot q_2)$$

It is only slightly more work to show that this is related to the curvature of the line interpolating these two quaternions. So, we add the term above to our 'distance metric', multiplied by a scaling factor.

- **recapture source material.** If we can, we ought to go down paths that were either present in the source material, or we have some other evidence to suggest that they are 'good' (e.g. behavior system experience that performing this path is a 'good thing to do'). Further we also want to penalize movement between nearby, but not subsequent frames – the main result of this local inhibition is fewer irrelevant paths being explored by path searching algorithms on this graph[8].

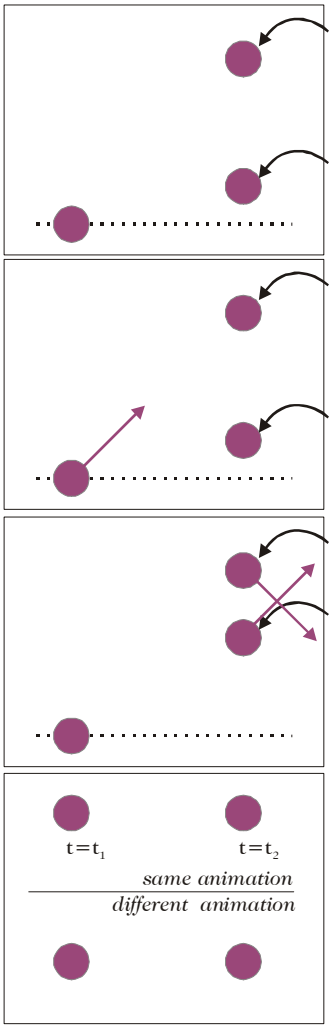
Some of this intuition is played out in figure 28. What we are left with is a distance metric that contains at least two somewhat arbitrary scale factors – one each for the acceleration term and the time curve. It might be possible, although it is as yet untried, to derive these factors *by example*. Given, that is, the connectivity information of some example animations.

Finally, we might be able to improve the performance of the distance calculations if we actually took note of the likely joint positions themselves – to have an idea of what is and what is not a reasonable difference in joint angle. Some joints (e.g. the elbow) have highly anisotropic angle distributions. Differences along some axes are more significant than others. Work on useful modelling (quaternion) angle distributions from

8. It is possible to envisage more exotic inhibition metrics – hand crafted metrics and metrics with multiple zero crossings (see *exchange,* page 185) can be used with close control over the source animation to help craft more abstract movement patterns.

source animation material is taking place in the Synthetic Characters Group. When these models become available, they can be incorporated into our distance metrics.

- *prefer similar joint angles*
- *prefer joint angles that follow on from velocity*
- *this can be symmetric*
- *include time preferences between animations of same source matieral (or when timing in known)*

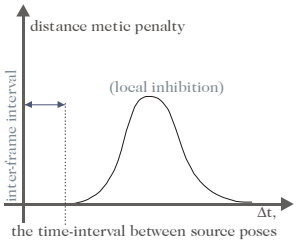


Quaternion distance metric:

$\text{acos}(1-q_1.q_2)$

Use time to closest transit (see over)

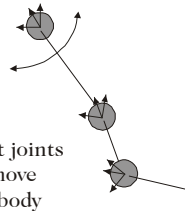
Use time curve



Each aspect is weighted and added to provide the distance metric from one *joint* to the other.

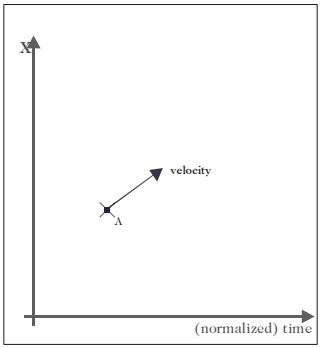
These (rather arbitrary) weights perhaps should be extracted from the data itself and vary slowly across the graph.

Once we have an interjoint distance we sum all joints in a pose:

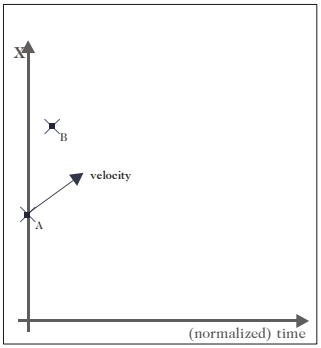


- weight joints that move more body mass (i.e. higher up the chain) than others
- animator or learnt preference global weight multiplier

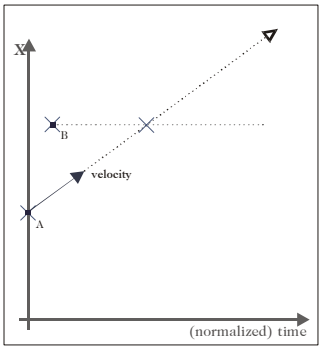
figure 28.building up the distance



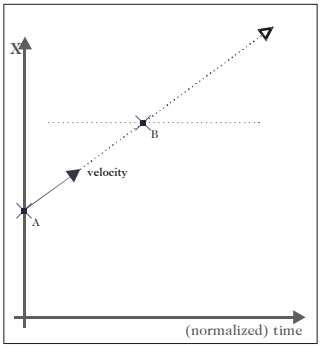
1. Consider a simple 1 dimensional example first.



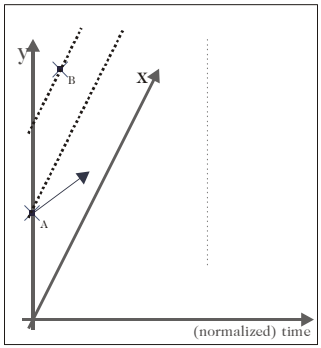
2. We wish to calculate where point B ought to be placed in time so that we can move through it starting at point A.



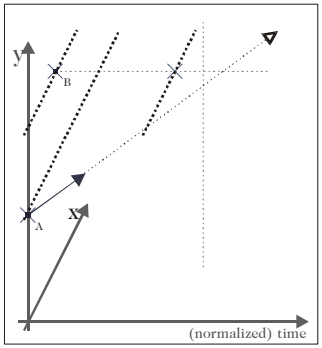
3. Obviously we can solve the problem with a very simple geometric construction.



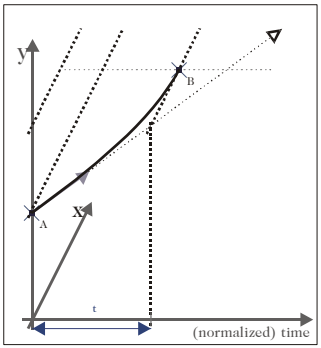
4. And, thus, create an interpolation with the minimum curvature (here, zero curvature).



5. Now, in 2 dimensions. Here we can solve the problem with a similar construction



6. ... but there is no guarantee that we will actually be able to *intersect* point B. So we get as close as we can.



7. And again we can interpolate out to point B in time t.
We can pose this technique in such a way that it can be used on the 3-dimensional quaternion manifold.

figure 29.calculating closest transit time

These distance metrics will affect which paths get taken through the graph. But before the graph can be built we must decided which edges we should create, i.e. which nodes we should connect together. Paths can be built only by moving along these connections.

But we need ways of building up these graphs from source material – perhaps many separate animations, perhaps a few longer animations. It is unreasonable for this to be a manual process. The simplest thing to do is to connect nodes to their closest n other nodes. Perhaps setting n locally – we know that we ought to connect to a subsequent key-frame, so we can set n to insure that.

By doing this we can read in whole animation files and automatically turn them into graphs. This is good, because we've already built the engineering to write and read animation files (since they are the building blocks of the blend-based motor system). Better, we know that the key-frame animation is the base-currency for all forms of animation generation techniques. But the animator is no longer tied to creating many small animations (endless types of `walk_turning_tight_right`, `walk_turning_left`, `walk_into_stand`, etc.) Previously the source ani-

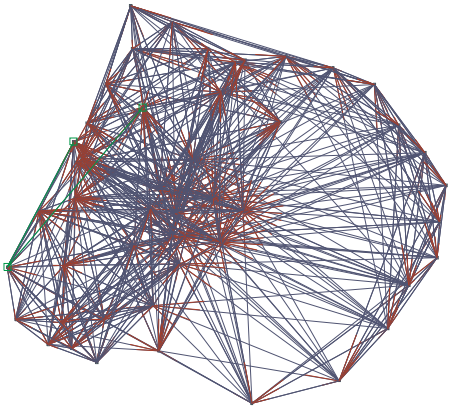


figure 30.a highly connected motor graph generated from 3 animations – sniff, play and beg

mations were strictly tied up in the actions a character can perform. Here instead they can be much more freely specified – a continuous animation of a character running, jumping, rolling over can feasibly be taken in.

The above connection criterion is not optimal. Several things prevent this from being a sin. Firstly, a missing connection might not be *that* bad - at the very worst the motor system can always interpolate the joint angles between two nodes to form an edge on the fly. The worst case scenario is exactly the kind of blending that blend-based motor systems perform all the time.

Secondly, we expect that the graph will change and grow during the course of the character's learning - so it is unlikely that we will have all the data that we will ever see at the time when we initially prepare the graph, and it unlikely that we will have the clock-cycles left for a complete reorganisation of the graph at run-time.

re-editing animation data

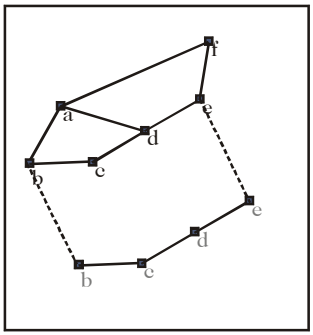
Finally the graph itself can be, and ideally will be, edited by the animator directly. Paths that look bad will be removed (and subsequently forbidden), or re-edited, or have their timing changed. In general we can identify for forms:

1. the topology of the graph can be changed, connections can be forged or explicitly removed and forbidden (e.g. in the case of a self-penetrating interpolation);
2. connection weights can be altered making certain routes less or more likely, or whole nodes themselves can be made more or less favoured (by applying a constant multiplier to the weights of their outgoing edges, we avoid the need to cache all blend weights);
3. the contents of the `BodyPose` nodes can be altered or rather blended towards the new material;

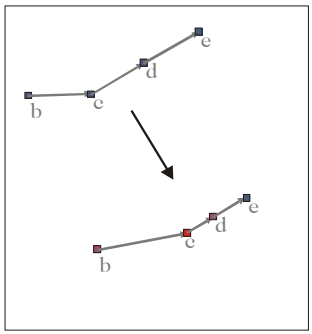
4. or new nodes can be added.

At present, any path that occurs through the graph during the course of an installation can be saved out as (re)sampled animation data, reloaded back into the animator's favourite software (presently 3D Studio Max) and edited. This data, when loaded back in, can be either be merged into the nodes that created it (3, above) or patched into the graph (4, above) as new material.

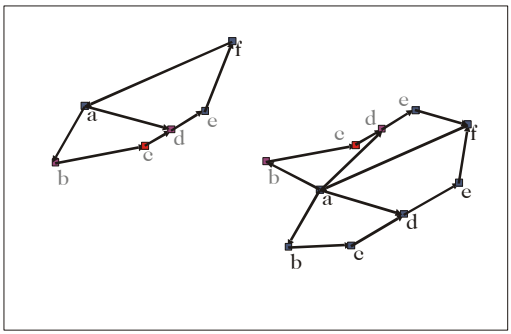
Perhaps characters start off as a series of connected static poses – literally, the 'character book' – and are built up in this manner. This might insure a structural coherence (poses lining up, timing relationships etc.) throughout the wide variety of animations animators are called upon to



1. Elements of the motor graph can be saved out to other kinds of storage (typically an augmented 3D studio MAX file). These chains come from the actual movements of a character, that take place during (the development) of an installation



2. Once saved they can be re-dited (both in content and in graph topology)



3. Once edited they can be reincorporated into the graph, as new material, or blended into what was there before

figure 31.graph operation for reincorporating animation data

create – a project-wide coherence that is difficult to maintain with tools today.

It is not yet clear how this ability will change the way that complex characters are authored – this technology is yet to be deployed in a large multi-person effort. What is clear is that this is at least some compensation for the increasing the risks that the motor system takes with their precious source material. The ability to "save out" material from an actual running installation, edit it, and load it back in such that it will be used in subsequent similar situations offers up a new relationship with the creative process. It is my feeling (and the feeling of the animators in our group) that the result will be to bring animators *closer* to the authorship process than they have been in the past.

towards a complete motor system

To explore what we can get out of this representation we need to build a motor system. One interesting note: the interface between the behavior system and this motor system is very similar to the interface to our previous, blend-based motor system. Just as before, the behavior tells the motor system a 'desired pose/animation' and the motor system tries to get there, keeping note of where it currently is as an 'actual pose/animation'. The initial difference here is the range of ways that the 'desired animation' can be specified and created by the motor system.

moving around the graph

How does the motor system take the body from an arbitrary `BodyPose` in the graph to a particular 'desired pose'? To do this, recall, we need to search for the shortest path along the edges of the graph.

A popular graph search algorithm goes by the name of the *A* search algorithm*. [55] It is applicable here because it can take advantage of both the intrinsic distance metric between connected nodes, and the extrinsic

metric between any two nodes – it knows to try to head 'in the right direction' when finding the shortest path.

The use of the A* algorithm will generate search results on demand - rather than a statically computed 'all pairs shortest path' algorithm. This allows us to change the distances of edges and change content and topology at run-time without an expensive recomputation.

Graph based motor systems can use the A* search algorithm to find the shortest paths from a current body configuration to a target body configuration, and hence can find out the minimum distances body configurations. We can travel along those paths by interpolating between the joint angles contained in `BodyPose`. We could perform this task in exactly the same way as we'd interpolate key-frames from source animation material[9], and exactly the same way as we performed 'temporal blending' for our blend-based motor systems.

That gets us the data for the key frames, but what about the times of those frames? If the two nodes originated from nearby each other in the same animation, we're in luck – we know the time difference between the nodes. Failing that, we can estimate the time it might take to interpolate between two frames based on the current joint positions and velocities of the two nodes – this calculation is similar to those that we performed to calculate the distance metric. Finally, it's worth noting that even these calculated time deltas might be modulated or overridden by a human eye during the re-editing process.

With this representation in place several problems dissolve. One problem originally discussed above, simply stated for a dog, is 'how do I get my nose there'. This general problem is similar to the head orientation, or

9. but while we can usually get away with a (spherically) linear interpolation between the finely sampled frames of an example animation, the nodes of a graph might not be sampled at such a high resolution. Therefore a higher order interpolation method makes more sense – thankfully one exists, referred to as *squad* in the literature. see, [82,100].

the shepherd stick positioning discussed above. But given the wide variety of strategies that the dog might employ neither inverse kinematics or a blind animation blend will work. We have to get the dog's nose 'there' *in style*.

If we have precomputed the position of its nose in many of the nodes in the graph we can search the graph for the pose that gets us as close to 'there' as possible. Given a graph the current body configuration is either at a node or travelling (interpolating) between two nodes along an edge, we can search for a path that takes the dog from where-ever is in this graph, to that desired pose. This means that the graph representation can be used to solve motor problems.

complex labels

In the current implementation a graph motor system contains the ability to, under behavior system command, find paths from the current `BodyPose` to any labelled node. But consider a `MotorProgram` which can be executed to return a graph node:

$$\texttt{desiredState} \leftarrow \text{motor-program}.\texttt{exec}(t)$$

These programs can act as labelled nodes that generates other nodes as targets – becoming a dynamic alias for other nodes in the graph.

We can use motor programs to complete common tasks that motor systems are called upon to achieve. In particular they handle the generation of cycles in graph based motor systems.

A simplest example program might be:

```
walk:
    getTo "walk start"; wait
    getTo "walk middle"; wait
    cycle
```

and this might generate a walk cycle. In each case the labels "*walk start*" and "*walk middle*" refer to labelled nodes, but they could refer to other motor programs – in which case

```
getTo "walk start";
```

would set a course for whatever node program "*walk start*" decides to go to, and:

```
wait
```

will wait this program ends or until it cycle's. Therefore we keep the ability to have a stack of programs and sub-programs active at any one time.

A better, more sophisticated program might be something like this:

```
walk:
    getToClosest {"walk start", "walk middle"}; wait
    getToFurthest {"walk start", "walk middle"}; wait
    cycle 1
```

This would have the property that it would get to closer of either of these two points of a walk animation (remember the biphasic nature of a walk cycle).

The point here is not to replace the functionality that could be better put into a behavior system, or even a behavior style layer above the motor sys-

tem. Motor programs serves as the replacement for fundamentals like animation cycling.

Multiple nodes can be associated with the same label. In this case the motor system chooses the closest – either by finding shortest path lengths or, if we're short on time, by just using the extrinsic metric.

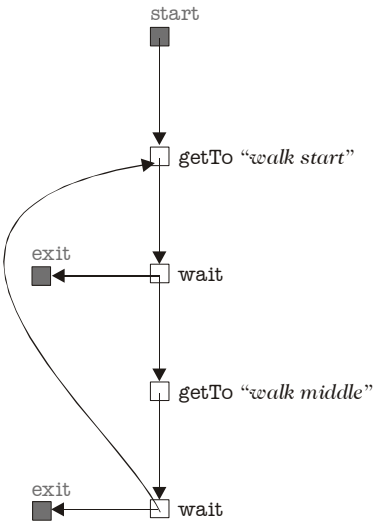
An immediate result of this is that we can write programs like:

```
beg:
        getTo {"beg middle"}; wait
        forcedGetTo {"low energy state"}; wait
        end
```

motor program graphs

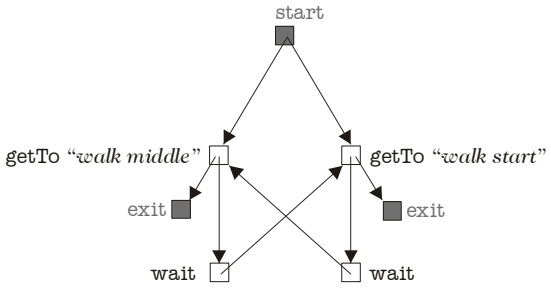
These 'language' that motor programs are written in is left as an option in the current implementation. While the above style of writing is easily translated into a simple stack machine, we can envisage other kinds of "program". One other way of describing motor programs is worth mentioning, because it more succinctly handles a number of program flow scenarios that arise in such motor programming and begins to highlight the relationship between this motor work and other motor control work.

We can create motor program *graphs*, where the program statements are laid out as vertices and program flow moves along edges:

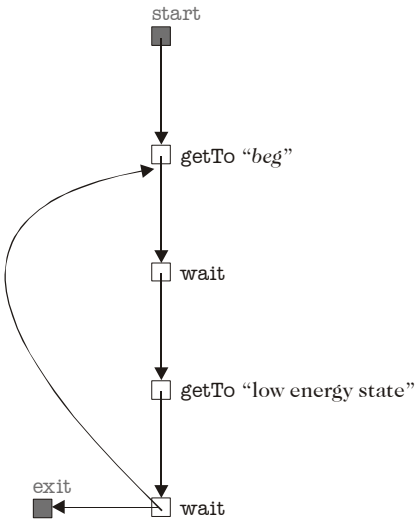


Flow begins at the special (and unique) node "`start`" and continues along nodes, taking the *shortest path* in each case. If a program has to finish (because the behavior system, or a program further up the stack has decided to move somewhere else) the behavior system can calculate the shortest path to a (not necessarily unique) node "`exit`".

But we can quickly draw more complex graphs:



Finally, the "forced low energy state example" again:



So far all three of these kinds of models have been built and used. It is a matter for future work to refine the interface for the creation and debugging of these graphs.

learning to chase your tail – part 1

With these techniques a complete body description can be much more compact – communication 'upwards' from the motor system to the behavior system is much lighter. The current state of the body can always be described as 'this far between node *n* and node *m*', or, if more that one graph is operating, one such statement for each graph.

This finally puts us in a position to approach the complex shaping problem – the dog learning to chase its tail. To achieve this the dog must build up a model of what motor actions to perform (typically, what motor actions it should perform in order for food or other rewards to appear).

Building up this model, in real time, using a representation consisting of raw, sampled joint angle positions is a daunting task for a blend based motor system. Firstly the amount of data involved is too large to conduct real time gesture 'recognition'. This pattern matching is required if we are to build models of these actions. We get models with sizes of around 625 quaternion values[10]. If we are to perform techniques such as discrete time warping not only is this number too large, but the quaternion distance metric is not a Euclidean distance metric – thus highly optimised off-the-shelf recognition primitives cannot be used[11].

Secondly, lets say that we are successful in learning a new action - one that leaves you with your nose in the air. A blend-based motor system, that operates on the level of animations rather than poses, might have no way to get the dog from this pose to any other pose (other than blind interpolation). There is insufficient contextual information stored in a verb graph to enable the creation of new parts of the animation-level transition graph automatically on demand.

Instead we can build a model consisting of node-node transitions. This model that does not look at the contents, but the identities of the nodes. What we had before was some 625 quaternions – "a set of all joint positions sampled over time". What we have now is a few node references – "node A, moving to node B, moving to node C".

Working with this representation is far faster. It is trivial to see if "node A" is the same as "node A" – we do this by pointer comparison. We are, in short, simplifying the gesture recognition and modelling problem by exploiting the fact that we ourselves are generating the data. Since we generate the data, we already know a good categorisation for the data

10. For example, if a dog has 35 degrees-of-freedom (each a quaternion) sampled at 5 times a second. For actions lasting around 5 seconds this results in a model too large to process live (around 2500 floats).
11. For example, the Intel recognition primitives library [44].

(the nodes) – thus, there is no need to approach this as a blind gesture recognition problem.

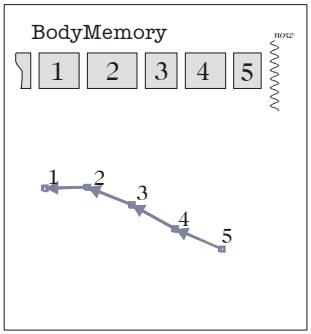
gesture graph generation

To build this model we can reuse the engineering created to handle directed-weighted graphs of the motor system – for these models themselves will be directed weight graphs. We choose a model in the style of Augmented Markov Models [37]. These are capable of capturing movement by the motor system as transition probabilities, and timing information (in a more flexible manner than straightforward Markov Models [72]). One final twist arises from the need to handle loops in the incoming data.

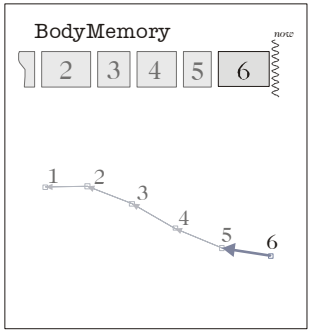
What we expect to happen, as we are learning a new gesture, is the appearance of a model that has a good, simple core of node transitions surrounded by non-causal and uncorrelated node transitions. Using this model we can define a confidence measure for a node – confidence, that is, that the node is part of the *core* of the model.

At present this measure consists of two terms. The first captures the increase in confidence associated with a node being present in many of the input data-sets. A second term reflects a decrease in confidence associated with having a range of outward transitions (i.e being part of a tangled mess) – this can be formally captured in the entropy of the transition probability distribution.

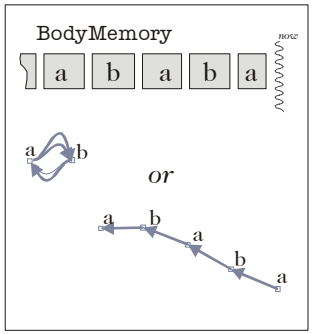
The segmentation technique attempts to identify the core of the gesture by starting at the node with the highest confidence and working outwards, both forward and backward in time along the paths of highest probability, until the confidence drops below a threshold for two transitions. This is by no means an optimal segmentation algorithm and others are being investigated. Incidentally, this is the segmentation algorithm used to support the simultaneous activation models discussed previously (see page 52).



1. This is a simple demonstration of learning a new “gesture”, a new sequence of motor BodyPose’s. Initially the model starts of blank. Then the contents of a BodyMemory is added to it. Here it is the sequence “12345”. Above, arrows point backwards in time. In these graphs the distances are related to the transition probability - not the transition time (although we also keep this information in the GestureNode representation)

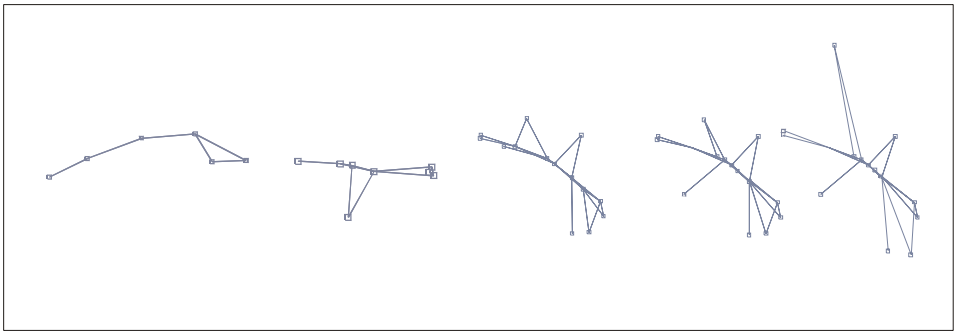


2. In order to know how to add data to this model, we need to know how to recognize where to put it. Here we have the sequence “23456”. In the light of the model this is well explained by the substring “2345” - therefore we know just to put “6” on the end of the model

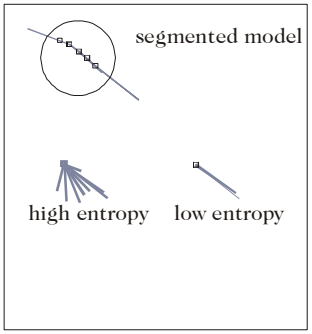


3. One trick: the need to handle loops correctly. It is insufficient to produce a tangled graph like the one on the left. Instead we must generate new nodes for repeated poses, and, when matching, try to match the most likely subsequence.

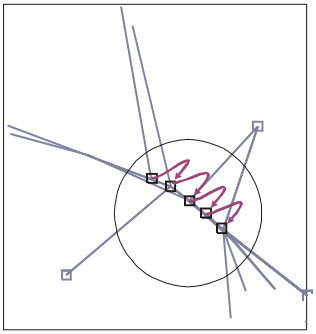
figure 32.learning and segmenting a gesture



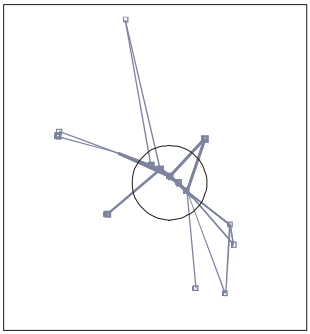
4. Here we show the evolution of a simple gesture model as it receives different but related subsequences from a BodyMemory.



6. We can separate out this core, by selecting connected areas pools of core nodes. Presently, membership of the core is dictated by two heuristics: transition distribution entropy (i.e. how tangled is the stuff coming from the node), and visit count (i.e. how many times have we actually seen this node).



7. This heuristic is fast to compute. At any time (before or after segmentation) we can *execute* the model, that is play out the node transitions as a sequence of desired body poses to the motor system, complete with timing information collected in the nodes. Thus this model represents a probabilistically stored animation sequence that is fast enough to work with at run-time



5. What we see here is typical of what we see in more complex models. A tight *core* of simple gestural material surrounded by low probability, uncorrelated / uncausal noise.

Regardless, this model can be sampled to create an executable `MotorProgram` – it should be apparent that it is already quite close to an executable motor program graph. Because of this we can attack the 'learning to chase your tail' style, complex shaping. We do this by embedding these gesture models into an *Ac* percept (compare with the simple shaping solutions). Note here that these gesture models posses all that is required of a percept model – they can produce categorization probabilities of incoming data (here data from a `BodyMemory`), and they can be sampled from (here in order to be executed).

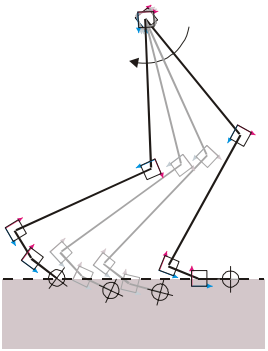


figure 33.blended joint angles are not equal to blended end-effector positions

There is a twist involved if we want to complete the problem as specified; a twist that is explicitly supported by our behavior system. Now, if we have an ability to guide our dog's nose with the training sick; and this ability will be wrapped in an action-tuple. Say,

moving stick | follow stick with your nose | around 10s; V = 10

gets created (by hand or automatically). We want to conspire to create another action-tuple:

whenever | execute *Ac*'s motor program | around 10s; V = 0

The twist is that we give this action-tuple a proximity of 1 to the "follow stick" action-tuple. This has two effects in a dog that follows the stick and is constantly rewarded for it: one, this action-tuple's *Ac* model will begin to build up a model of the contents of the bodies memory, after execution of "follow stick with your nose". With consistent training, this will be a model of, say, going around in a tight circle.

Secondly. this action-tuple will begin to grow in value as (and if) "follow the stick" grows in value. So soon, the second action-tuple might take over – eventually our dog's action selection will choose the latter action-tuple to execute. Then, if this experimentation is successful, a few executions with sufficient reward, we'll have a model in *Ac* that looks like chasing its tail.

After a certain point, we can pull out the core of *Ac* and 'dispatch' it down to the motor system – treating it as new animation material to be added to the graph in the same way that we treat new animation material from the animators. This material will either be stitched into the graph, or we will perturb the distances between the nodes involved in the new material present in the original graph. And leave behind in its place an alternative action-tuple, say,

whenever | execute `MotorProgram` new_gesture_1 | around 10s; V=50

creating a MotorProgram that walks the motor system through the new material.

Finally, if we like, we can place the *Ac* payload "execute `MotorProgram` new_gesture_1" into an *Ac* percept tree. This means that the rest of the behavior system can try out this motor command as material from which to form action-tuple hypotheses.

communication with the behavior system

The communication between the behavior system and the motor system becomes simpler and easier to maintain. We are getting closer to a motor system that can communicate with the behavior system in the "problem solving" ways that we'd like:

- **different resolutions.** While the behavior system can talk in terms of 'sit down' (coarse resolution) it can also talk in terms of 'get my nose here x' (fine resolution). The graph supports the solving of the problem of how the body should get there.
- **different kinds of labels.** The richness of the representation lets us build different kinds of labelled states into the system. These include small programme like nodes (see below), special markers like 'low energy state' and pre-computed end-effector positions.
- **pre-categorized body description.** Now that we have a graph of `BodyPose`'s we can express the state of a body in a far more com-

pact fashion. Instead of returning a set of all joint angles and velocities the motor system can describe its state as ‘halfway between node 32 and node 47”, and we can compute connected distances along the graph between nodes. We’ll see how this can be used soon.

enforcing physical constraints

The graph structure detailed above gives us a few opportunities to improve the physical plausibility of the animations that the motor system produces. Consider what happens once the dog succeeds in getting his nose as close to the stick as possible. If he happens to be precariously balanced on his hind legs then he shouldn't stay there, he should return to the nearest 'low energy' state. Low energy states can be hand marked (by animator) or annotated automatically (where we have zero velocity poses for long periods of time). All this occurs without behavior system intervention, and more importantly, without behavior system management.

Another example: precomputed end-effector positions enable the stabilizing of feet position. Taking them closer, that is, to positions given by a Euclidean interpolation (e.g. on the floor) rather than where interpolated joint angles on articulated chains put them (e.g. through the floor). You can do this by exploiting our ability to move the entire creature (e.g. upwards) to compensate.

We can proceed along the following lines: blend the end-effector positions in Euclidean space, perform the joint angle interpolation, work out where those end-effectors ended up and compute the difference. Then we attempt to reduce the difference between the Euclidean blend and the pose blend by translating the root node by the average of the difference.

In the specific case of a creature-on-floor we can do a little better than this – by weighting the average by the reciprocal of the distance from the floor. This makes sense because people are using the floor to judge where the character is in 3-dimensional space – it is more important for a crea-

ture to keep his feet exactly on the floor than it is for him to keep his feet exactly at some arbitrary location in space.

future extensions

self-intersection removal

Another 'physical' problem that plagues blend-based motor systems is that of self penetration. The graph structure gives us mechanism where self-penetration can be reduced. Of course, for this we need an algorithm that can detect self penetration. Such algorithms are slow (with arbitrary geometry intersecting with arbitrary geometry, far too slow for real time). But if we wanted to, we could run the entire graph through such an algorithm off-line, removing the edges that result in self intersections.

persistent 'mergeable' graphs

Such off-line, path based approaches have the advantage of parallelizing well – especially when a character's motor graph is seen as a single group-wide persistent entity. With some engineering I believe the merging and concurrent versions management abilities offered by existing software (already used by groups of developers) could be leveraged towards this end.

Further, it is already the case that the graph upon which motor systems are based is persistent – it has to be to support the addition and editing of animation by animators. A construction of a text based external representation that allows merging and simultaneous editing of graphs completes the reintegration of the animator into the development cycle. The challenge remains to provide the necessary user interface components to explore and maintain such graphs[12].

node models

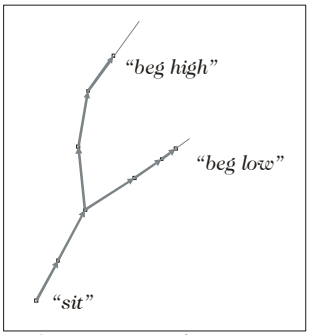
Above we used self-organising maps to learn smooth surfaces fitted to some source data. The benefit with such a representation is, of course, the ability to evaluate the surface at points not in the original example set. We can perform exactly the same task with basis function interpolation. We cannot however, easily create in-between poses with a directed weighted graph – unless those poses happen to lie along the edges of the graph. Although the self-organising maps (SOM) and the directed-weighted graph are at two ends of a continuum, SOM's don't learn high dimensional, non-simply-connected manifolds very well – we can't replace the graph with a SOM and train it up on some animation data.

In losing the ability to provide a continuous variety of motor movement we've lost a lot. It was this interpolation ability that let us walk smoothly around, and perform simple shaping. Thankfully, we can envisage the reintegration of these things back into a directed-weighted graph representation with more complex vertices.

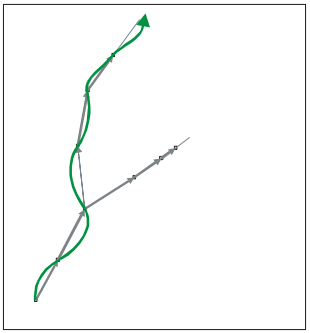
Two techniques can be used to recapture what blend-based motor systems could do, while keeping what graph-based motor systems have won us. Firstly, with minimal engineering, we can create paths that are the result of interpolating paths through the graph. We can do this by construction a new kind of connection – called here the *spoke*. While animation usually flows along edges, it flows perpendicular to spokes. The following diagram shows this in action. Vertices that perform this task would still contain `BodyPose`'s, but augmented with some additional logic.

---

12. some steps towards these components have been taken – not least the graph visualization software developed during this thesis (which is responsible for all the directed weighted graph diagrams in the printed matter).

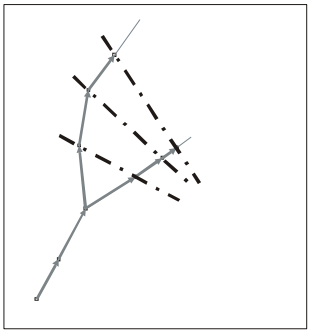


1. above is a part of a motor graph. Marked is a pose corresponding to "sit" and two nodes corresponding to the apex of "beg" animations. Other material (not shown) will take the character back to a sitting position.

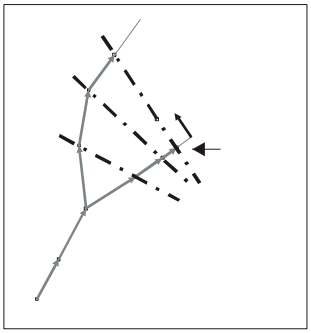


2. Paths that the motor system can take are limited.

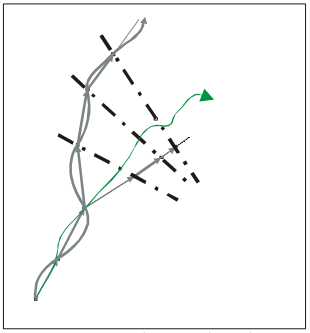
figure 34.adding blending back into graphs



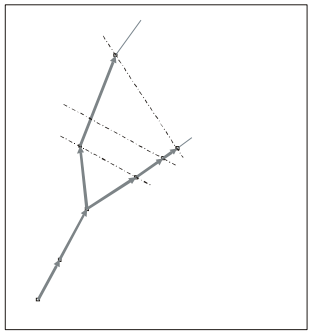
3. In order to recapture some of the benefits of the blend-based motor system we define cross linkages between nodes.



4. Searches for solutions to motor programs can take place at higher resolution with no extra cost if the interpolated solution is located on a cross-link that has the closest node as one of its ends.

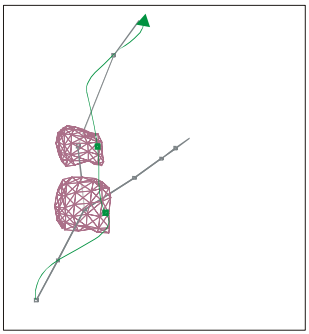


5. A motor path can then be made from interpolating paths either side of the cross links.



6. Since interpolating between paths is so trivial, there are few constraints on the layout of the paths. This graph representation can recapture the piecewise linear timewarping used for adverb blending, and can attempt to automatically generate correspondence points through the `BodyPose` metric.

Note that we are essentially performing automatic correspondence point finding, in this representation.



7. Finally, we can embed little self-organising maps into the graph nodes themselves. These nodes are, of course, trained on animation data.

Note our ability to search for solutions on graphs containing such structures can be maintained if a few conditions hold. For example, if we are looking to put our end-effector nose as close to a particular place $\bar{x}$ then the real solution may lie along a spoke. Since we cannot search the entire space created by all the spokes (that would take to long) we will only be able to find the correct solution if the closest node lies at one end of the spoke containing the best solution. We can then quickly find the blend parameter along the spoke that we require analytically (for a taste of this mathematics, see back to the derivation of the distance metric page 123, or page 218).

An alternative would be to keep these linear blend extensions internal to the nodes, thus removing the ability to solve problems with this extra dimension. This approach is more applicable when the blend access is accounting for emotional change – if the dog is sad then it cannot use happy dog animations to get its nose nearer the food. Looking back over the two approaches we can see much of the ground-work for blend-based motor systems is present in the graph based work. Here the spokes framework take the place of the timewarp into neutral time rather naturally.

For other applications this additional blend dimension will be insufficient. The next level of complexity consists of incorporating statistical models into the nodes themselves – models which learn the relationship between new style adverb parameters and `BodyPose` material (including timing information).

The only fear here is whether increasing the complexity of `BodyPose` will cause the complex shaping approach to fail – recall that this is tenable only because we can quickly state that the distance from "node A" to "node A" was zero using pointer comparison. With more complex nodes, this might no longer be the case – we might have "node A(a=0.2,b=0.5)" to "node A(a=0.7,b=0.1)"; with 'a' and 'b' adverb parameters. But these fears are groundless, for we can still compute fast metrics between these nodes by using the adverb parameters and I ex-

pect these metrics to be no less well behaved than those in use already. This is, after all, a space that we have learned from the original example animation data – what better space is there to define a distance in?

concluding remarks

In this chapter we have built two different kinds of motor system, one of which (the graph-based) is completely new. While the blend based motor system has precedent in the literature (i.e. [75]) we've been able to push it further because of the facilities provided by the behavior system. There can be no doubt that the graph-based representation is very simple – but we are constrained by both the example-based and the real-time natures of our chosen domain.

Both serve different ends. The next stage is to work out a hybrid motor system that supports both. A real solution here is less a hybrid graph/blend system than a programming environment that makes it easy to move between the two paradigms as the character authorship situation demands.

Having such a malleable motor system is of great benefit to our work. It lets us build with hierarchies of temporal constraints. Our experience of previous architectures and installations shows that it isn't sufficient to have a behavior system that is ignorant of temporal constraints and leave it all up to the motor system, nor is it sufficient to have a motor system that know no constraints and leave it up to the creators of the character's behavior.

Although this work stops short of demonstrating a solution to the complex shaping problem – a dog learning to chase its tail – we've made a significant approach. We have our problem solving ability to guide the dog's attention towards the 'training stick'; our gesture learning techniques to learn the chase tail; motor programs to play out the animation; the behavior system modelling to support the learning and eventually to

reuse the learnt gesture. It is now, for the first time, feasible to create a synthetic character who can learn new animation material at run time.

music

This section serves two rôles. Firstly it attempts to justify the approach: the creation of music with synthetic characters. It locates this work in the wider context of interactive works with musical components, the authorship environments used for such works and the recent emergence of a community focusing on the biological roots of music.

Secondly, it details some musical applications of the structures discussed previously. These applications have typically taken the form of installations or short videos or performance works and here we put them into context of work pre-dating this thesis in the Synthetic Characters Group.

This chapter, like those proceeding it, is a chapter of unfinished ends, of works-in-progress. Having decided to make interactive musics with virtually embodied reactive creatures, it is important to first sketch the shape of the aesthetic space this idea contains. These works are that sketch.

why do music this way?

(void *): a cast of characters

At SIGGRAPH '99 the Synthetic Characters Group premiered an installation entitled, (void *): *a cast of characters*[7]. More complete descriptions of the work have been given elsewhere [104] – but the work contained the first example of music from a character, so we'll briefly discuss the environment. The piece contained three characters – *fast Eddie,* the "dude"*; Elliot,* the "salesman" and *Earl*, the "trucker". Without human participants these characters would sit at lonely bar (with no chance of service). However, equipped with a pair of forks stuck into bread rolls, participants could select one of the characters to *possess*, forcing their legs, perhaps against their will, to dance[1]. By gesturing with the bread rolls, the characters on screen mimicked your dance moves.

Initially Elliot and Earl will dance very reluctantly, but, if the participant treats the characters well (criteria include not making them fall over and not making them do painful dance moves) they will begin to open up and enjoy themselves. This will be reflected in their facial expression (created by two target vertex blending); the quality of their animation (a two example, one axis adverb space between happy-nervous/sad-angry); the behavior of the characters around them (for example, they'll begin to applaud if they see somebody 'loosen up'); whether they keep dancing by themselves when you put down the bread rolls (they will perform the dance moves that they've discovered that they like); and finally, their starting position along the happy-nervous/sad axis the next time they get possessed changed to reflect their previous dancing experience.

It was the intention that the piece should have a cinematic feel. Since there is no navigational component to the piece there was no obligation for the camera to maintain a constant orientation with respect to the

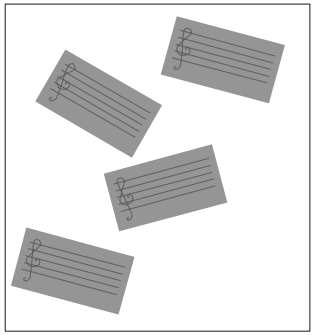
1. inspiration: interaction, taken from Chaplin [18], set taken from Hopper [41].

controlled character. The camera is therefore much freer – to pan, cut, select close-ups. The creation of the this cinematic 'camera creature' is described in detail in [93,94].

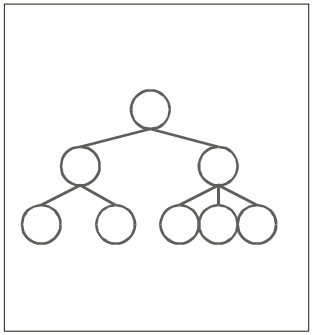
The missing piece of cinematic tradition is the film score. Hence the desire for a "music creature". Its rôle: to help ensure this cinematic mood by the addition of a suitable film score. The problem simply stated is how should one write a film score when there is no script? Going over the outline of the interaction above we realise that this is a rich interaction to score. Short and long term emotional changes take place, discrete behavioral decisions occur, each by three characters. In addition there is a complex camera system that is not always showing every character on the screen.

The prime consideration for the music in (`void *`) was that it should never sound *bad*. It should never really become the center of attention, and it should certainly never do this by sounding broken. In this sense the music had to *support* what was happening on the screen but not come in front of it. Many people never noticed the dynamic quality of the music – this is probably a symptom of success.

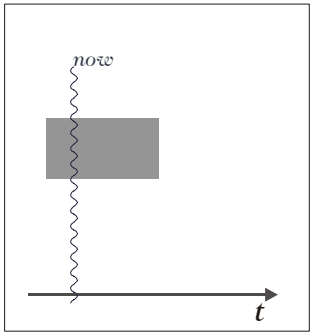
figure 35.images from (void*)



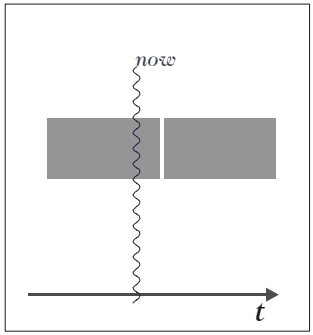
1. low level output system built from *tiles* of pre-composed music.



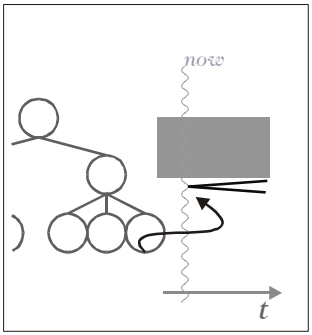
2. the layout of these tiles are placed under behavior-system control (using the *Scoot* behavior system). This behavior system runs on a separate machine from the output layer.



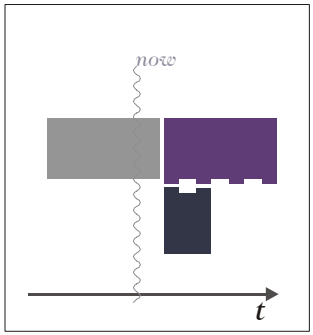
3. tiles get layout out in time, under behavior-system direction...



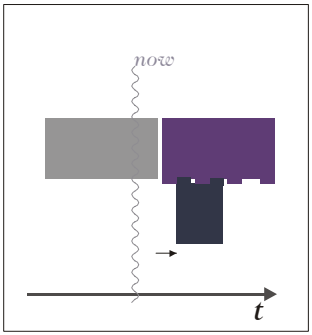
4. ...just in time to get played.



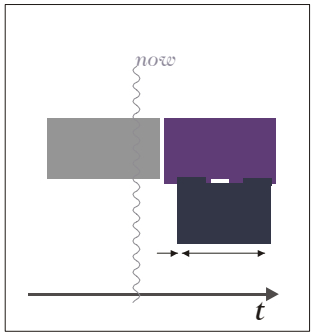
5. in addition to changing which tiles get laid out when, the behavior-system can also modulate the contents of the tiles (changing volume, and MIDI controllers).



6. constraints, such as beat alignment and tempo matching can be solved automatically by the output layer (see, correspondence points for adverb blending)

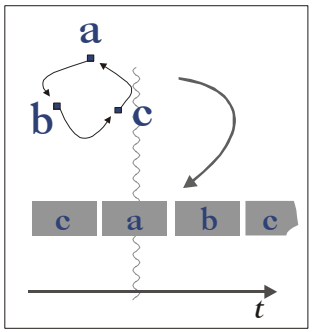


7. beat alignment through time shifting.

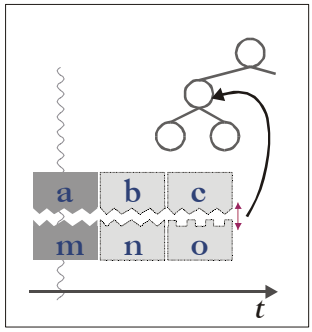


8. beat alignment through time stretching.

figure 36.the (void*) music system



9. *pattern generators* can act as intermediaries between behavior system and tile layout. These can encode simple sequencing tasks better than *Scoot*'s reactive behaviors.



10. *pattern generators* can be used to predict the future. Therefore conflicts can be "reacted" to well before they get played.

11. All this can be recaptured by the new action-tuple based behavior and graph-based motor systems.

- *behavior system* → *action-tuples*
- *output layer* → *motor system*
- *temporal constraints* → either the action-tuple's *Du context*, or *motor system*
- *pattern generators* → *motor programs*

While this output layer (and large parts of the behavior system) was custom built for (`void*`), only the very lowest level representation (the content and metrics for `BodyPose`) need be changed for the new motor system to produce notes.

interactive music score

The use of music to aid and reward the voluntary suspension of disbelief is traditional. Music is present today as a background to the most pervasive story media – television, films etc. It seems clear that an unfamiliar medium without many pre-established conventions might benefit from the inclusion of such non-diegetic music.

Events in the world often cause and are caused by emotional changes in the characters – their actions occur in a rich emotional context. Participants become immersed in a scene only if they can make sense of that scene. Our core goal – that participants must be able to 'read' characters – can be aided by music.

Interactive music has precedent in computer games. But the problem there is a far less interesting one. Computer game characters typically have much less internal state, no emotion system to speak of and little no long term memory. There is, frankly, much less to score, and you can achieve the 'never sound bad' criterion by the looping of CD audio (with loops attached to environmental locations) and the occasional fade to and from silence (between locations). Indeed, as soon as game engineers discovered how to stream CD quality audio during gameplay, the industries brief flirtation with note-level in-game composition all but ended. Sound is now commonly limited to crossfading safe, environmental background music.

But when the characters become complex these static, looped approaches break down. In failure, either the integrity of the music or the readability of the characters is threatened by a disconnect between the sound and the visuals – violating those very conventions that we hoped to borrow. To go beyond this is to go beyond uncommitted and possibly irrelevant background music; to provide a musical score for a medium with no script; to rise musically to the challenges offered by the richness of the characters.

In (`void *`) the music was tied to the fate of the characters in the diner in a number of simple ways. Rather than building the score from individual notes, small segments of music pre-composed by a human musician were dynamically ordered, stretched, overlaid and arranged just before it has to be played.

- Thematic development was driven by developments on screen on a character by character basis - this is nothing more than *leit motif*.
- Key events, for example a character becoming possessed, have specially written music - just as film scores are build around 'spotted' music created for key scenes.
- A character's stance towards the user's control is connected to the timbre of the characters' music - the orchestra is not deaf to the emotional content of the scene.
- Thematic balance is biased to take into account what characters are visible on screen - similar to the final mixing and composition process in, again, film score.

Ideally, all of this control acts to subtly but robustly reinforce the user's interpretation of the scene.

In the finished piece one finds, anecdotally, that it is the good stories that make the most interesting music. Interactions in which many things happen at a well ordered pace and seem to fall into some sort of 'arc' result in the most varied but coherent music. An alternative outcome - where the best music is generated when nothing happens - would have suggested a return to the static looped background piece, and would have been much simpler to construct. It is clear that by motivating music change by events onscreen the two most common complaints of 'algorithmic composition' - that the music never changes, and when it does it isn't clear why - has been avoided.

after (void *) — lessons learned

Since the music was another character, a common currency between characters echoed the sort of information flow that we expect to need. This eased the flow of information between all the characters - including the camera - to the music.

This architectural decision points towards the idea of an *autonomous* music. A music that is serving and protecting its own (musical) needs - including never sounding 'bad' - while surviving and serving in the dynamically changing virtual world of the emotions and decisions taken by the creatures it provides the scores for. The goals of a good action selection mechanism – of coherency and relevance – apply equally to this music. It must survive through conflicting goals, sudden changes, and constrained actions.

the case against direct control

In creating the music system for (void *), I ruled out the option of giving participants direct control over the music. In general people aren't, don't want to be, and certainly shouldn't have to be, competent musicians. What was needed in (void*) was a music that took care of itself. Secondly it had to do this with an apparent transparency of intention - exactly the same interpretable depth and presence that we seek for our synthetic characters. In short participants must be able to 'read' the music in the same ways as they must be able to 'read' the characters and the scene as a whole. These are precisely the same reasons behind the decision not to give direct control over character animation to participants.

In general terms, this rejection of direct control, or even literal control, leads us back to a character based music approach. To support transparency throughout the complexity we need for a layer of intelligence somewhere. What sort of relationship should the participants have with the music? There are particularly appealing modes of interaction that take place between embodied agents – conversations and dialogues, people

can *play* with creatures, duet with them, people recognise and can be recognised - and conversations occur between embodied things. Other kinds of relationships are less rewarding.

However, this character metaphor remained in the background throughout the piece. In (`void *`) never was the music *center stage*. Nobody, out of the hundreds of people who interacted with the piece interacted with it for the music (at least consciously). Had it have been missing, it might have been missed, had it have been broken, it would have received much more attention. Any further investigation of what it means to make a 'music creature' really had to put a music creature at the focus of an installation, otherwise we might stall the aesthetic ambitions of this project.

the case against low risk music

This project then returns to another of the themes of this thesis: risk. The music in (`void *`) is low risk – it plays pre-composed music, bent, layered and faded into shape, but pre-composed all the same. To do anything else is a risk by comparison.

We can alleviate some of the pressure by changing the aesthetic – by moving away from the interactive film score goal towards an interactive music installation. The risk now becomes the lack of author control over the piece, threatening the artistic integrity of the whole venture.

Splitting up animations into graph fragments is a similar risk, indeed giving up conventional ideas of author authority when entering an interactive medium is a risk. In the end however, I believe that this risk is necessary, interesting, and inevitable if we are to go any further.

Presently, the artifacts detailed below mark a move *away* from what one might call an example driven approach to computer music. I believe this apparent step backwards is only an initial step. Just as we had to weaken the rule of the example when developing the graph-based motor systems, here we reduce our reliance on the precomposed segment of music. Just

as we dissolved the example animation with the intention of building having creatures competent enough to put them back together, we do the same here for music – replacing the role of a human crafted musical structure with the first steps towards creatures capable of recapturing those structures by themselves.

the case against radically different music creatures

The music creature for (`void *`) was built with some of the same underlying structures as the other, on screen, characters. This was the *Scoot* behavior system augmented with a completely custom output layer. Rather than playing animations, the music creature's 'motor system' plays small lines of music; rather than using its 'motor system' to move around the world, the music creature moves around a score.

Where the reasons, intentions and emotions must be conveyed in the characters' motor systems so too the music must convey them through its 'motor system' - the sound. Where the motor system shouldn't fail on the technical level, the sound system also shouldn't fail – notes and keyframes both must be delivered 'on time'. Where the animations shouldn't fail stylistically, the elements of the composition shouldn't fail. Similar transparency and robustness is required by the author of the composition. Otherwise, the (necessarily interactive) composition process for an interactive piece is untenable.

While we were reusing the behavior system for the creation of the music, the music creature's 'motor system' was a motor system only in name. After the finalization of the (`void *`) piece it became clear to me that many of the problems that the music creatures "output layer" was attempting to solve were common to other creatures as well. These included the synchronization of different elements that had extension in real time; the specification of temporal constraints; the patterning (or sequencing) of smaller routines into larger routines. It was low down in this layer that *Scoot*'s inability to deal with time was being compensated for. The behavioral problems discussed in the introduction to this thesis were

being solved locally inside the music creature – why shouldn't all creatures benefit?

The reverse is also true – if we increase the sophistication of the other creatures and we might leave music creatures deficient. So many of the problems surrounding the creation of music with our creatures were either problems faced (and perhaps dodged) by the behavior system or were problems that were going to be faced soon enough that to keep their development separate did not make sense. This is an argument for a parsimony of effort – common problems should have common solutions (and common engineering). But it is biased by a belief that movement and music are not unrelated, that it makes sense to think of 'the temporal arts' [31] – especially when building the core competencies of creatures up from nothing.

sketches

common themes

This section details some of the installations and videos created after (`void *`), with the express purpose of sketching the shape of music creature based music. Common themes throughout these sketches include:

- **motivation based authoring**. Creatures that have their behavior described at the highest level in terms of motivations. There is a nice progression through these motivations to an installation design pattern that starts from agreed *texts* and what might be called musical *character books*. We'll see below creatures that 'get lonely when nobody sings to them' and 'get angry after a while of being shut out' etc. Evaluation of the 'truthfulness' of these statements are irrelevant. Instead I hope to have shown that this thinking is a powerful way of approaching the authorship of installations with musical components – an area with traditionally few handles for the author to grasp.
- **virtually embodied simple creatures**. Music creatures with bodies – be they graphically represented, tangibly present in an installation or altogether hypothetical.
- ***proto*-musical abilities**. The recent interest in the biological origins of music research focuses on a top-down, deconstructive approach to this problem. Some of these installations can be seen as bottom-up, constructive reaches into these fields. Here we combine in creatures a small number of very simple perceptual abilities that are aspects of music perception and composition and see what we can get from them.
- **complex systems created by a number of simple creatures**. Can we take our texts and our ideas and create installations (and musics) this way?

- **equivalence of live performers**. There is not an obsession with ground-truth fidelity in these creatures – we are, after all, not building robots. But in some of the pieces detailed below there is an opportunity for live performers to communicate with creatures in exactly the same manner as they communicate with each other. This is a place to locate the human performer that aids transparency and interaction rather than fostering a process of conducting or even editing the output of complex systems.
- **gestural and 3-space control metaphors**. Gestural control techniques seek to provide mappings between gesture and music – typically gesture and synthesis parameters, or gesture and algorithmic composition parameters.

  There is now something close to a rich tradition of this kind of work in the interactive music community (for a review see [66]). It might sound obvious when stated – that there is, after all, something rather special about 3-dimensional space – but it is worth stating nonetheless. Gestural control and other spacial mappings leverage authors' and participants' ease and natural abilities in real space.

  If it makes sense to map the gesture of humans to music this way, if it makes sense for composers to think about music this way, perhaps it can make sense to map the gestures of virtual creatures to music as well.
- **interactive authorship of non-interactive works**. Finally, if we're looking at using music creatures to make interactive pieces, to use music creature metaphors to author complex worlds, perhaps we can use them to create finished pieces. There is a tradition, more present in music than elsewhere, of constructing systems, winding them up and letting them go. If characters are useful for interactive pieces then they are useful here too.

sand:stone (installation)

introduction

sand:stone was an small interactive installation [95], the first after (`void *`), and the first to approach some of the themes mentioned above. Music here is a major component of the installation and the most interesting component of the interaction.

The piece is loosely a response to Shelley's poem *Ozymandias*.The interaction interface consists of six rocks on a bed of sand. On screen, a single character – a statue. The proximity to the center of the bed of sand and the identity of the rocks is sensed[2] and forms the basis of the interaction – firstly by directly modulating the pose of the character on screen, and secondly by controlling the colony of music creatures.

music creature design

Each rock "contains" a simple musical creature. Each creature can sing – decide to play out individual, precomposed music – and the decision to sing is based on a simple motivation mechanism. The creature's presence in the center of the sand bed increases its motivation to sing, the rate of increase is governed by a boredom variable. Singing is suppressed on hearing uncorrelated music from other creatures. Finally, singing uses 'breath', and when a creature runs out of breath it must pause.

Such descriptions might be hard to describe (and understand) in English but they are well modelled by "synthetic chemistry" – [39] a model of (linear) 'chemistry'. Here we use the expressive power to create interactions between motivational (the desire to sing) and sensory variables. The internal time and how this time progresses through the creatures precomposed segments is governed by a final piece of motivational state.

I met a traveller from an antique land

Who said: `Two vast and trunkless legs of stone

Stand in the desert. Near them, on the sand,

Half sunk, a shattered visage lies, whose frown,

And wrinkled lip, and sneer of cold command,

Tell that its sculptor well those passions read

Which yet survive, stamped on these lifeless things,

The hand that mocked them and the heart that fed.

And on the pedestal these words appear --

"My name is Ozymandias, king of kings:

Look on my works, ye Mighty, and despair!"

Nothing beside remains. Round the decay

Of that colossal wreck, boundless and bare

The lone and level sands stretch far away.'

Ozymandias —

Percy Bysshe Shelley

2. sensing technology developed by the Responsive Environments Group at the MIT Media Lab. [67]

Timbre is modulated directly by the boredom model, and by the time of day in the installation world.

Although this piece, like (`void*`), is reliant on precomposed passages, it moves towards characters that have an awareness of the contents of the the progress through those passages. The motivational model, perceptual abilities – based on a 'circle of fifths' distance metric – and action selection create simple, but complete, character. A colony of such characters is capable of producing interesting musical forms from a tangible interaction. Once revealed participants find the creature metaphor compelling and understandable musically.

status

This installation is on tour internationally with the 7th New York Digital Salon.

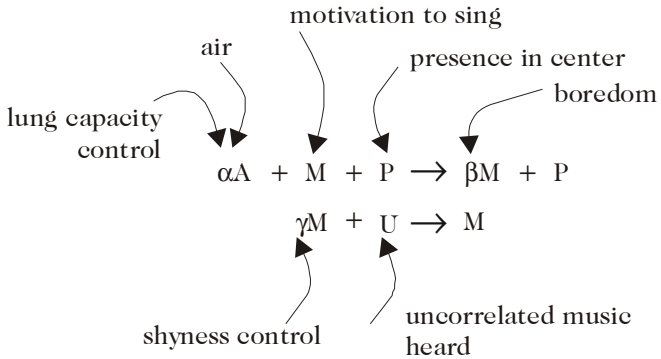


figure 37.synthetic chemistry for motivational variables

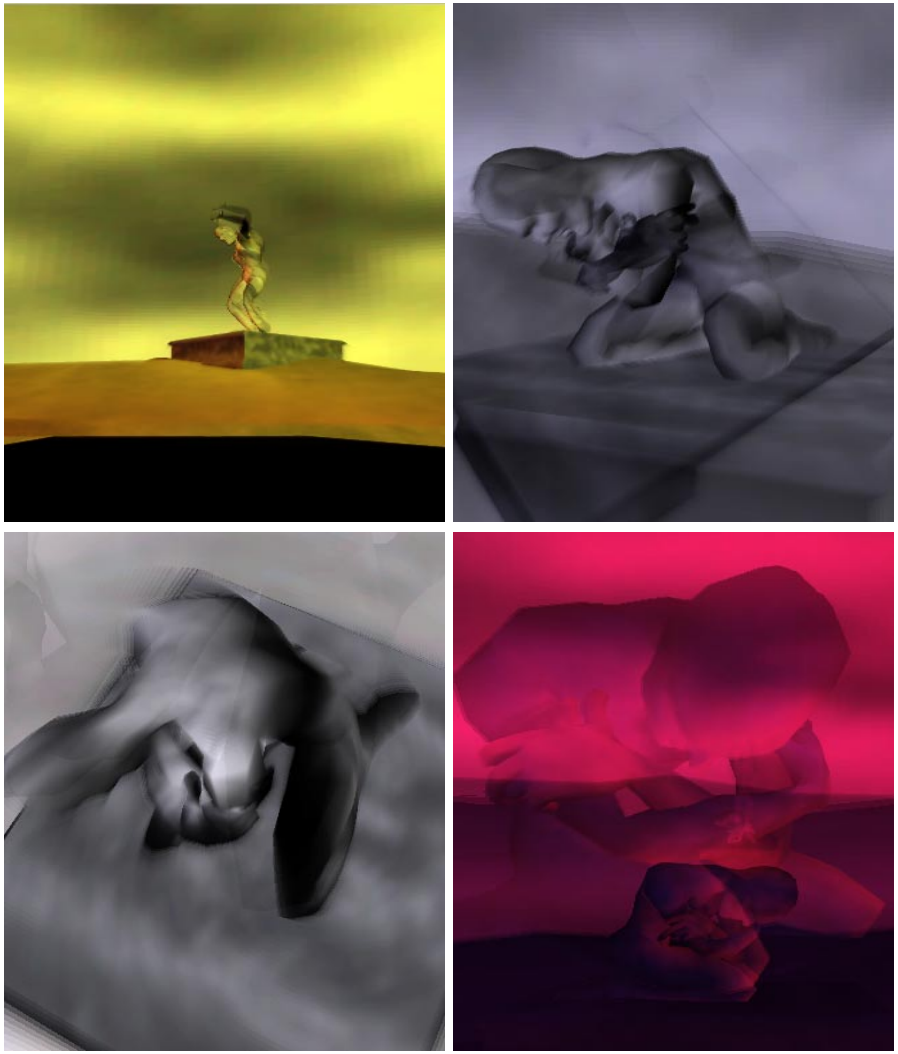


figure 38.images from *sand:stone*

three creatures (performance)

introduction

three creatures is a performance piece for piano and two independent musical creatures. The title is, of course, driving home the idea of equivalent communication between creatures.

The creatures in the title are simple 'creatures' coded entirely in *csound* [24]– and run on physically separate machines. Without graphical instantiations, they communicate through a loudspeaker each, and hear through a microphone. They communicate, therefore, with each other, in the same manner that the performer communicates with them.

Each has a simple motivational model – controlling the sound that they hear, and their willingness to sing it back out. This is driven in part by a simple correlation metric across what they hear and what they remember. Their memories were periodically flushed with pre-selected samples by an external "score providing" computer.

This installation was an exercise in the design of a system with a "creature metaphor" outside the context of the behavior, motor and graphics systems of this work and instead inside a dataflow language increasingly used for real-time interactive sound.

Although interesting (and enjoyable) to play, the piece was most intimidating to develop and it is almost impossible to consider extending it further. It serves now only to disprove a null hypothesis – that we might be able to create character-like music without the burden of behavior-based AI.

But the dataflow representation demanded by the language is simultaneously at too low a level to create an illusion of intelligence in the "creature" and, bizarrely, at too high a level (that of information flows rather than sample manipulation) to construct any perceptual abilities.

What the installation suggests is a hybridization of the lightweight data-flow synthesis paradigm with the rest of the work of this thesis – but it also illuminates the necessity for a connection between these worlds that is both *bidirectional* and surprisingly high bandwidth. This is less surprising in light of idea that the complexity of the interaction (from the musician and other creatures) must come from both the perceptive abilities of a creature (*from* the low-level system), and the complexity of the creature's action (*to* the low-level system). If either part of the whole is deficient or mismatched then the metaphor breaks down.

soundcreatures (installation)

introduction

soundcreatures is an installation for one or more creatures. Here, each creature has a graphical representation, in addition to a microphone-as-ear and a loudspeaker-as-mouth. This installation is a study using some of the aural percept models previously discussed to analyse auditory events and the use of action-tuples to represent musical events.

creature design

soundcreatures stores their sonic repertoire in *Ac* percept trees using our cepstral aural model (page 65). Persistent aural models (that is models that can be saved out from one instance of a creature and reloaded into another) offer an ability to seed the creatures with musical material. Key elements of their behavior are:

- **the perception of sound rewards behavior**. Their preference towards exploitation versus exploration is controlled by a motivational variable – a motivation to hear things – which in turn is increased during times without sound. Therefore creatures are learning what sounds they need to produce in order to hear sounds back.
- **sounds they hear get incorporated into their repertoire**. Sounds that they hear are classified into trees by the percept tree mechanism. This common tree is shared in both *Tr* and *Ac* areas of the action-tuples. Attended sounds can then become material for use in creating new action-tuple hypotheses. These *Ac* trees are discussed below.

- StateObject **processing allows the filtering out of self-made sounds**.One technical thing that these creatures need to do is to not listen while singing – we cannot expect that the creature's ear (microphone) will not pick up the creature's mouth (loudspeaker). This inhibition of hearing while singing is easy to produce with StateObject's ability to process the saliency of data before passing it on to the Percepts for evaluation (see page 39).

  Although such *half-duplex* communication does not strictly occur in humans auditory systems, it does occur in their visual systems – optical processing is inhibited during a saccade (a rapid, ballistic eye movement). It is possible within the framework to simulate this with a trivial amount of engineering.

- **sound processing and playback through graphical movement**. It is the creature's *body* that produces the sound – the creature literally breathes out the notes. The body here is manipulated using our blend-based motor system. Labelled positions on the animated body control parameters such as amplitude modulation (e.g. rib cage)

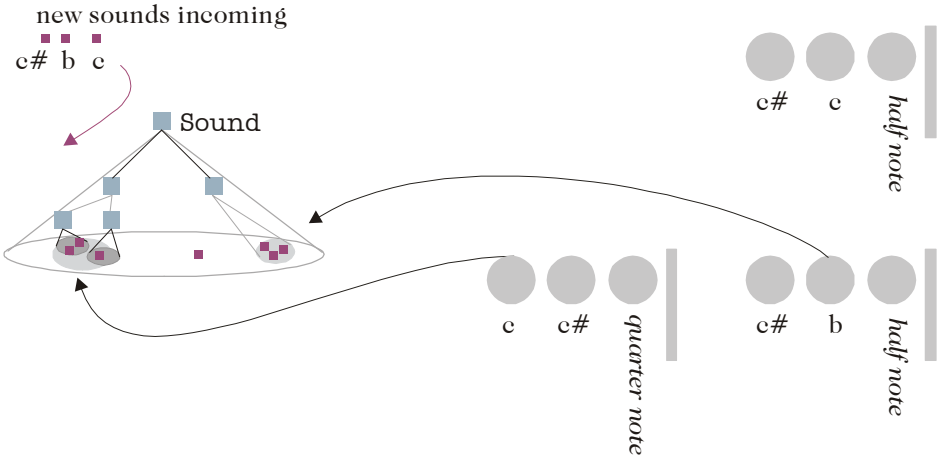


figure 39.new sounds are incorporated as triggers

and a simple low-pass filter (e.g. here, back angle). Finally, the amplitude of the resulting sound "blows" the camera around the creature.

Some time was spent building the tools to enable the connection of positions, distances, angles and velocities constructed from end-effectors, blended-vertices and planes to be connected to real-time sound processing through common signal processing elements[3]. The result is an open framework for processing sound with movement.

aural *Ac* percept trees

Recall that we can arrange *Ac* percepts into trees, just like other kinds of percept. The use of *Tr* percept trees to guide hypothesis generation is pervasive in our creatures. We create children action-tuples with *Tr*'s referring to the children of percepts. *Ac* trees creates a parallel mechanism by which, in response to increases in value, new *actions* can be generated. Here it is simply used to create new, and more specific, sonic material; the dynamically growing tree becomes a templates for the sounds that the creatures can produce (as well as being reused by *Tr* percepts to model the sounds that we can react to).

Action-tuples start at the top of the tree, carrying a payload that plays out a wide variety of sounds (here, a wide variety of notes) and slowly refine what it is that they should be playing on the basis value propagation through the behavior system.

For this we might not use cepstral models alone – without phase information we couldn't reconstruct model samples perfectly. So here we cheat and also keep the raw samples around[4].

---

3. although a better (and future) approach is to off-load all the sound processing onto separate hardware, communicate via a network protocol, and use existing real-time sound manipulation software.

A better space to produce and categorize musical sounds, especially if we want to include extra causal dimensions (including value, but also perhaps some motivational state of the creature) might be found in [81]. This work combines a data-driven methodology necessary for accepting raw sound; a cluster based model ideal for putting into Percepts; and an incremental update algorithm that might perhaps be put into a on-line creature.

musical structures from behavior

Creatures like these are platforms for directly investigating what musical structures can be obtained from the behavior system that we've built. The current installation only scratches the surface.

Looking towards the future we can see a continuing exploitation of the action-tuple structures for musical ends. As hinted at in the opening discussion of the behavior system, the behavior system is capable of representing, through action-tuples with forward transition *Du* models, the network structures used in musical analysis and data-driven music composition. But the tangled, hierarchically natured arrangements that are possible – both structures in the behavior and motor systems possess the computational power of structures like *augmented transition networks*. [21][5].

The hierarchies of actions and the goal directed value modulation developed in the behavior system can echo the hierarchical organisation of some musics. Properties, uses and shortcomings of this view of musics

4. More investigation is required to ascertain properties that blending in this space possesses although it is probably interesting to blend raw samples *after* a discrete time warp.
5. Augmented transition networks are a tool from computational linguistics. Of course we cannot mimic the parsing abilities of these networks with behavior system structures directly – in particular there is no way to back-track in real-time. However we can mimic the *generative* abilities of such labelled arc transition networks rather easily.

have been discussed at great length in the literature. Gone, however, from our representation are the hard hierarchical boundaries. No hard edges (we can have multiple membership in hierarchical actions and soft goal distances in value modulation) and no hard coded dogma (goal distances can be learned and action hierarchies generated in response to changes in value). I am confident that these structures are a proper superset of many algorithmic composition techniques.

status

A single soundcreature was shown in April 2000 at the Digital Life open house at the MIT Media Lab. Multiple sound creatures have played together only informally, and await public performance. Finally, there is interest in giving a sound creature or two to one of the "parrots in residence" at the Media Lab – bringing our inspiration from nature full circle.

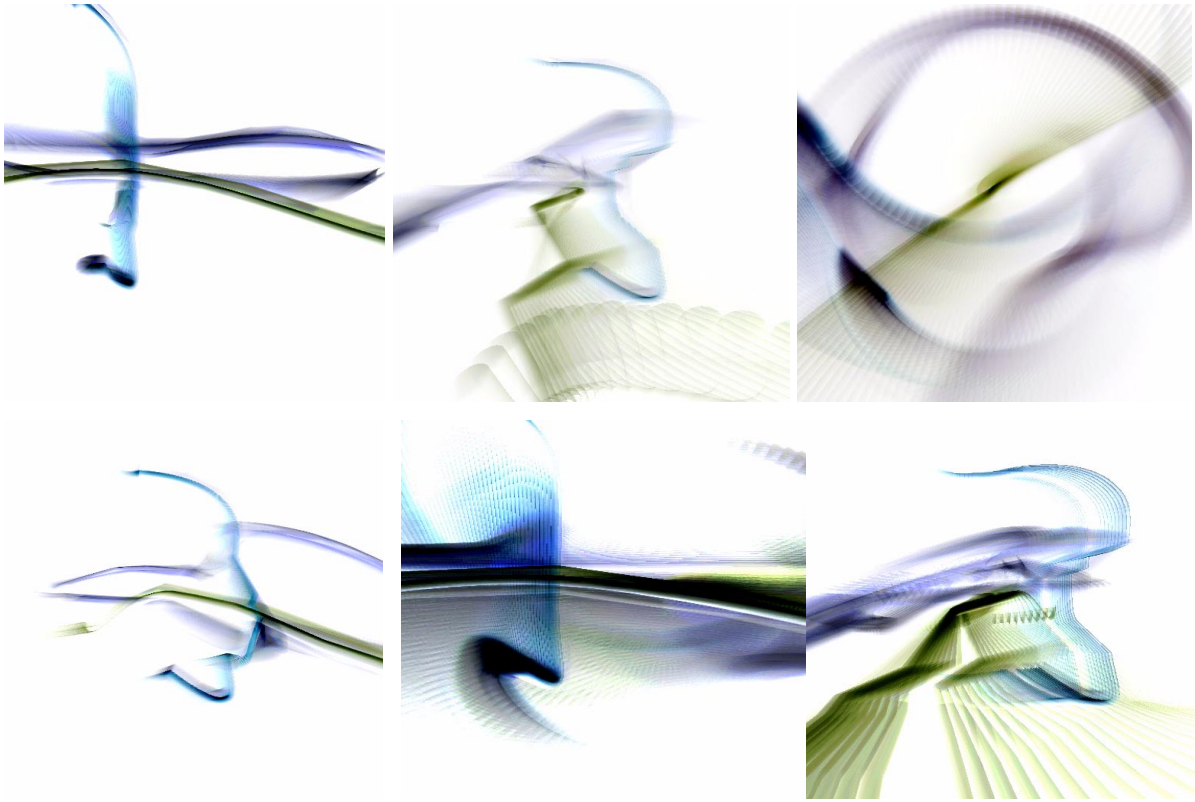


figure 40.images of a sound creature (inverted)

ozymandias (video)

introduction

Ozymandias is a piece for video reusing images generated by the *sand:stone* installation. Audio for this video is supplied by a driving a colony of simple sound creatures with the spoken text of the poem.

Simple creatures are trained on the text of the poem (by repeatedly playing the text at them). These creatures are similar to those in *soundcreatures* although the reward strategy is different - here the author controls the reward of these creatures with a simple mouse-click. The resulting creatures act as a stimulateable delay line, or some complex echo chamber, becoming excited by the text of the poem, and reading parts of the audio material back out of time.

While in training creatures can produce their sound live, but instructions concerning the manipulation of sound are also stored to file – equivalent in use to an *Edit Decision List* (EDL)– for higher fidelity off-line manipulation.

off-line image processing – image creatures

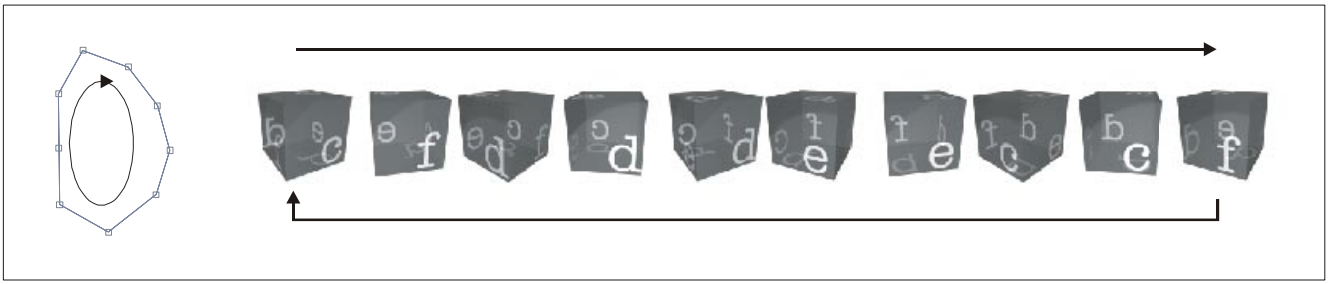
Of all the pieces discussed in this chapter, this one is unique for the off-line processing of graphics. The source material for this piece was around 1200 frames taken from the *sand:stone* installation. From a low resolution sub-sampling (120 frames) of this video stream a tangled, directed graph was created. This structure is directly analogous to the one used in the graph-based motor systems[6]. The distance metric is based on the sum of the squared differences between frames.These graphs capture a complex structure of ambiguities present in a video stream, in a way that allows graphical exploration.

---

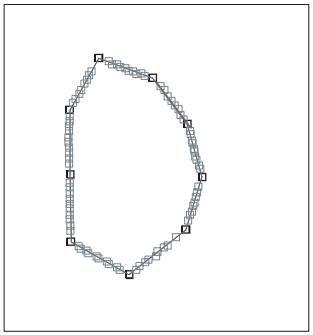
6. although this work actually pre-dated the graph-based motor system.

The creation of the video stream comes down to another, trivial, behavior system – which explores the graph in response to the commands read in from the audio EDL. Its purpose, using the "problem solving" capabilities of the graph, is two-fold: to move along paths created to get to video material corresponding to the *time* that the sonic creature has taken audio material from; and to modulate how these paths are played out according to the number of times that the nodes have been traversed – specifically the amount of motion blur applied along the path.

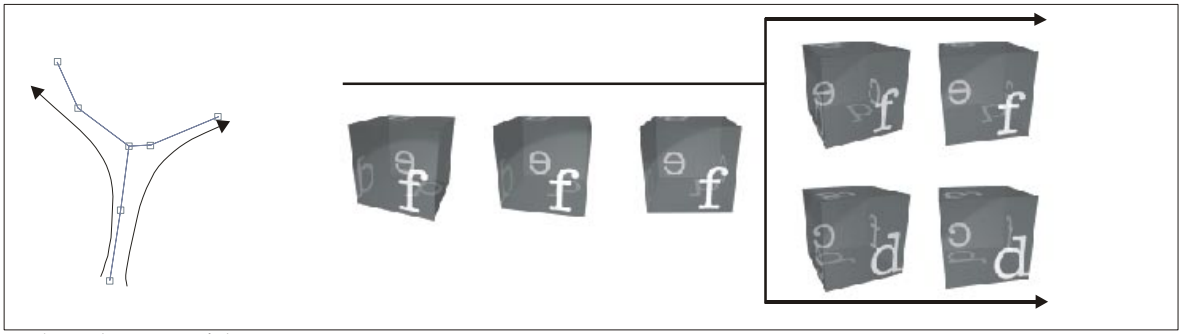
This extends our ideas of characters into rich *image based* rendering. Similar structures, without the benefit of a unifying motor system framework, are explored in [80].



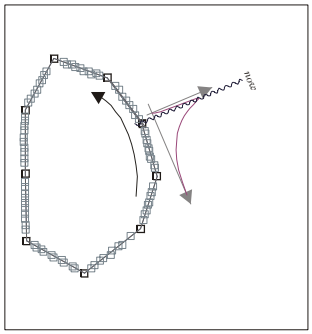
1. Image-graphs can be used, just like motor graphs, to represent common structures found in video. They are interesting because they can also recognize flow ambiguities in this footage. Here we represent a looped animation.



2. Unlike motor-graphs, computation of distances and the initial generation of the graph would be prohibitively expensive if we fed, in this case 1200, frames of animation into the graph generating algorithm. Instead we subsample the images by a factor of, here, 10; compute the graph based on that, and add in the in-between frames to the graph. These structures are not too dissimilar to the *spokes* discussed previously.



3. Above shows one of the structures present in a graph of the rotating cube footage. This simple (sum-of-differences-squared) distance metric is dominated by the mass of the cube, but the symmetry of the animation structure is broken by the textured faces. Lacking in the original implementation was any equivalent of the velocity information present in the BodyPose metric, although an easy step towards this would come from *optic flow* techniques.



4. Finally we can redefine common video processing techniques along the paths generated by the "motor system" surrounding these graphs. Of particular interest for the Ozymandias piece, which deals with the destruction of material over time, is motion blur. Instead of generating material by interpolating (by weighted sum) between nodes in the graph, we can take weighted sums of groups of nodes along the path.

exchange (installation)

exchange is an installation for a small colony of creatures. Like the *soundcreatures* these creatures have virtual bodies that effect the production of sound. Here, the emphasis is on the arrangement of notes, not necessarily the content of the nodes themselves.

The exchange creatures have the ability to play notes (through external MIDI[7] communication) and to hear notes that other creatures (and people) play (again through MIDI, or, more robustly, though a UDP protocol[8]). They are also equipped with microphones.

These creatures can play a variety of very simple "games" with musical material that they hear. The simplest is quickly stated by two "rules":

- every time a creature hears a note, it places a copy of that note "in space"
- every time a creature strikes a note in space, it plays.

The notes in space are represented in *note objects* discussed below.

creature design - behavior

These rules are played out by a character with a prepared graph based motor system – created with source animation material from, and re-edited in 3D Studio Max [30] (see page 132).

In the absence of any modulating motivational state a slow progression around this graph is enough to produce organised sound that is sometimes startlingly "musical". Played out in a spatial representation, it is

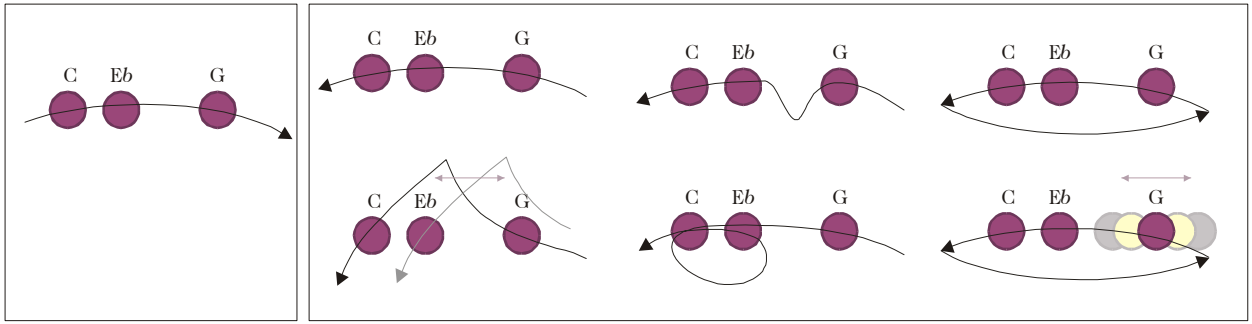
7. MIDI – *Musical Instrument Digital Interface*, is an interface for passing notes and other synthesis parameters around among computers, keyboards, synthesizers etc. [61]
8. UDP – *Unicast Datagram Protocol*, is a lightweight protocol for the exchange of packets of data over standard physical transports (including local ethernet and the internet at large)

clear that passages will get exchanged between the creatures and then distorted – retrograde variations, repeated subsegments are swapped back and forth in an accidental recombinant evolution.

But the graph based motor system that we have built enables characters to do much more than blindly bump into these note objects.

The first extension we can build in is a "hunger" for notes. The motor system knows how to answer questions like "what is the closest note object?" (in a distance metric appropriate to the motor system). Then, having answered, the motor system knows how to take us there. Once a character can find note objects in space (i.e. the behavior), we can control from a high level the desire to find notes (i.e. the motivation).

To complete the control of this motivational state we tie this desire for notes to the *hearing* of sound. If a creature gets "bored", it is more likely to head directly for any musical material around.



1. Notes can be encoded in space and played out by moving objects.

2. Several, musically interesting, transformations can be conducting within this spacial domain.

figure 41.note-space metaphors are musi-

One final twist offered by the graph-based motor system for the authoring of these non-representational characters is the ability to perturb the distance metric of the `BodyPose` structure. Here we disrupt the time-difference component of the metric to unusually favour low-integer-ratios of the source animation length. This distorted metric echoes that found in sound perception, and enables the animator to carve the temporal equivalent of the note triads out of animation material.

A brief look back over the last description will reveal that we've introduced a large number of "constants" into the description of the creature – the signal processing of all these quantities, the expected ranges of the "boredom" state variable, etc. Such constants are the bane of any character designer. The setting of them is, in short, a tedious and not particularly interactive search through a complex high dimensional space – exactly the worst excesses of producedural authorship that I have argued against.

One part of this problem will remain a genuine user (or author) interface problem, and the current environment goes some way to try to ease this aspect. But we can go some way to dissolving the problem by including some of the adaptive models discussed before. The relationship with the motivational boredom variable and the behavior can be mediated by a *model* of that motivation variable and hence specified in abstract terms – that is to say, terms free of the specifics of the numbers. This is no more that the Gaussian model discussed earlier (page 50). Similar *scale-free* techniques can be implemented elsewhere. In this example the connection with perceived note density and the motivational state was similarly mediated.

This motivational linkage helps keep the music on a certain edge between excessive complexity and complete decay. We can then *script* this linkage overtime to create a waxing and waning of a creature's musical complexity. This kind of high level control (well discussed in [8]) is a natural feature of character-based design.

The above discussion of this musical game refers to "placing a note in space". What does that mean?

Theses spacial notes are represented by virtually embodied note objects. These note objects have a specific role to play in shielding the rest of the installation design from the underlying musical synthesis paradigm. This is a good thing because the most common paradigm is still MIDI[9]. In this architecture, each note is treated as an independent entity, orthogonal to all other notes, including notes of the same pitch. This is not a good model for anything other than (perhaps) keyboard instruments – it would not be appropriate for this unbiological, uninspiring, and unmusical representation to "leak back" into the design of musical creatures.

More useful models can be made out of the things that MIDI offers. Note objects can support the idea of "being stuck" and closely subsequent striking events can perturb (the synthesis of) a note already in progress. The idea can be extended into cases where the objects themselves have behavior – sliding into the situation of inter-character interaction through physical contact.

Actually the geometry, and some of the behavior, of these objects has been explored in other fields; most notably the interactive dance community has had an interest in virtual tangible objects (be they 3-dimensional objects, or planes projected out of 2-dimensional camera space) and their relationship to tone generation (for example, the Palindrome dance group [65], [20] or [3]).

It is here that we can begin to leverage 3 dimensional spaces for the purposes of synthesis control. Here the possibilities are, trivially, endless:

---

9. a transport layer to the software synthesis programs SuperCollider [87]and Max/MSP [25] has also been constructed that does not suffer from these limitations.

distances between characters limbs and objects; plane crossings and collision geometries; physics simulations on note objects etc.

The difference here, of course, is that these possibilities are being played out in a world without sensor problems or soldering. I propose that, while fun to explore, these possibilities only become interesting in a virtual environment if the entity at the center of all this interactive opportunity has the ability to interact convincingly with it – in short, if that entity is a compelling synthetic character, equipped with appropriate perceptual and behavioral intelligence.

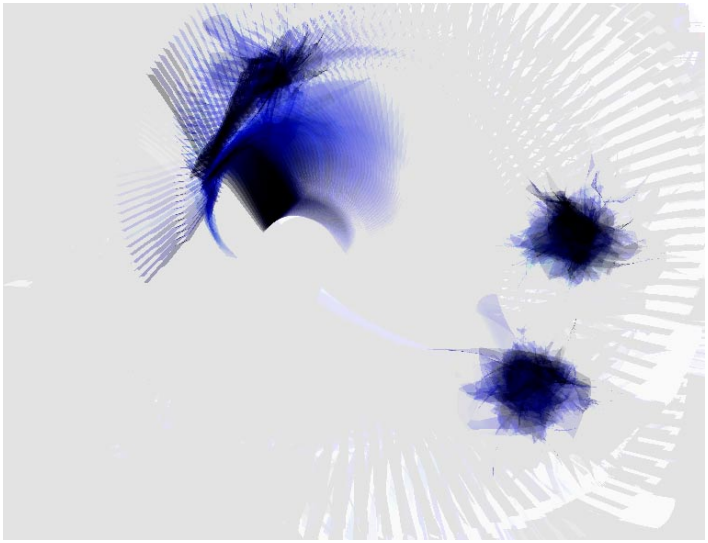


figure 42.a music creature chasing down two notes (inverted)

listen (installation)

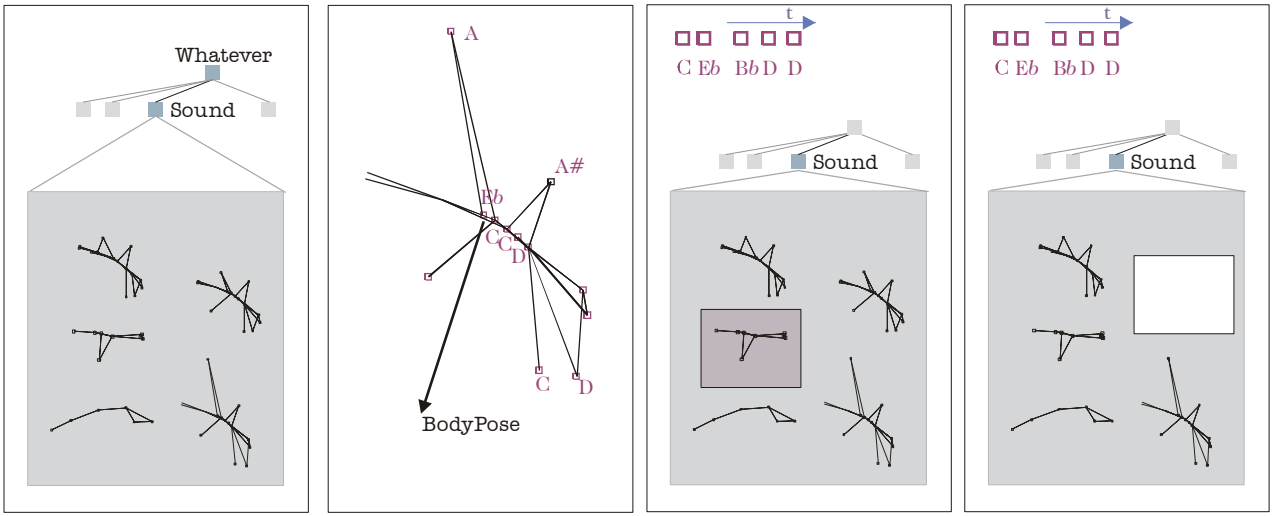
introduction

The idea: create a creature that can dance, by using very simple musical perception. The methodology: repurpose many of the ideas that we've discussed above, use the body representations to represent music, and use musical ideas to represent space.

creature design

listen creatures possess a body – a similar body to the *exchange* creatures. Creatures can listen, again over MIDI or over the same UDP protocol as the *exchange* creatures. Over these channels they obtain labelled note events (time, pitch, volume).

While active the bulk of the creatures computation is spent maintaining a *population* of percepts modelling sequences and timings of notes. These models are nothing other than the gesture recognition models from the graph-based motor system, page 123.



1. the percept tree consists of a population of "gesture" models.

2. Here however, the nodes keep track of notes, and keep references to body poses.

3. At any one time there is a model which best fits the memory of the recently heard nodes. This model dictates the movement of the creature, and, with behavior system interaction, can get updated with this recently heard music.

4. Models which fall below a certain standard are removed. When the recently heard model is not matched by anything in memory, new models are created.

figure 43.keeping a model population in a percept

Although here, the nodes do not just contain references to `BodyPose` they have been repurposed to contain also references to `Note`'s that have been heard by the creature.

Since these models have classification abilities (page 142), at any time there is a model that is doing the best job of matching a buffer containing our running memory of musical events. This model is "played out" through the animation system using the stored references to `BodyPose`.

Movement is therefore created out of the creatures modelling of the music it hears – a dance originating in both reactive and predictive responses to music.

extensions

The seed idea for this piece trivially opens up to a large number of possible treatments, and there are a number of versions of this work that could be realized.

The most promising of these considers an internal musical space – an abstract inversion of the *exchange* creatures (where space is turned into notes) – mapped out by the music that the creature hears and remembers.

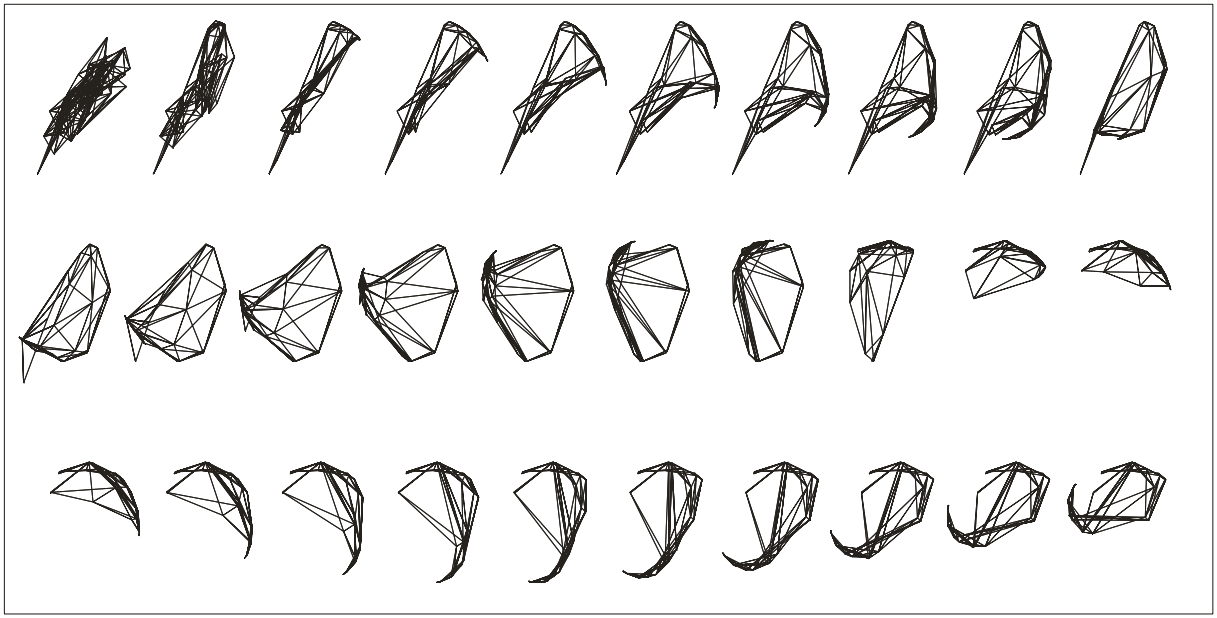
To perform this, two self-organising maps are used to simultaneously learn two distributions. Firstly, the temporally local distribution of musical material and secondly a map of body movement derived from an animation playing continuously in the background (providing a background cyclic structure to this graph). SOM node plasticity can be connected to the saliency of note events – memorable notes result in memorable poses. In the current implementation we use note velocity as a salience signal, but in the future, this can be driven from a expectation violation event.

Together this provides a platform for testing the self-organising representation extensions of the directed-weighted graph structures. The rest of the behavior system is unaltered – the predicting population of models

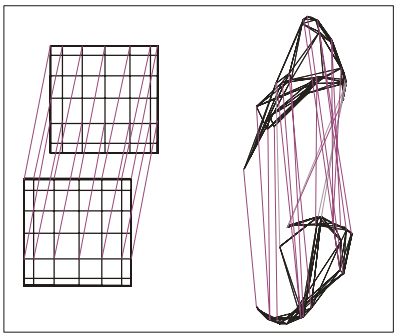
can now dance by finding (paths to) body pose clusters related to note clusters in a dynamically changing music-pose mapping.

status

Both *exchange* creatures and *listen* creatures exist as a video – *exchange: listen: exchange*. This was obtained as a single continuous take without any external interaction other than the injection of three musical fragments into the "ear" of one exchange creature. Motivational changes were pre-scripted. *listen* listens to the two creatures play with this material. *listen* premiered at SIGGRAPH 2000 (listening to music from [67]) and an *exchange: listen: exchange* installation premiered at the opening of MediaLabEurope, July 2000.



1. here a self-organising map (with BodyPose nodes) is being trained on a slowly changing animation. This use of the self-organising map is in some ways an *ab*use – such maps are typically used with data presented in a random order to prevent the patterns seen above. Here however, we are interested in tracking *non-stationary* distributions – and seek to use the artifacts above as a memory of pose configuration.



2. Here a SOM is modeling recently heard music can be connected by strict topology to the pose SOM.

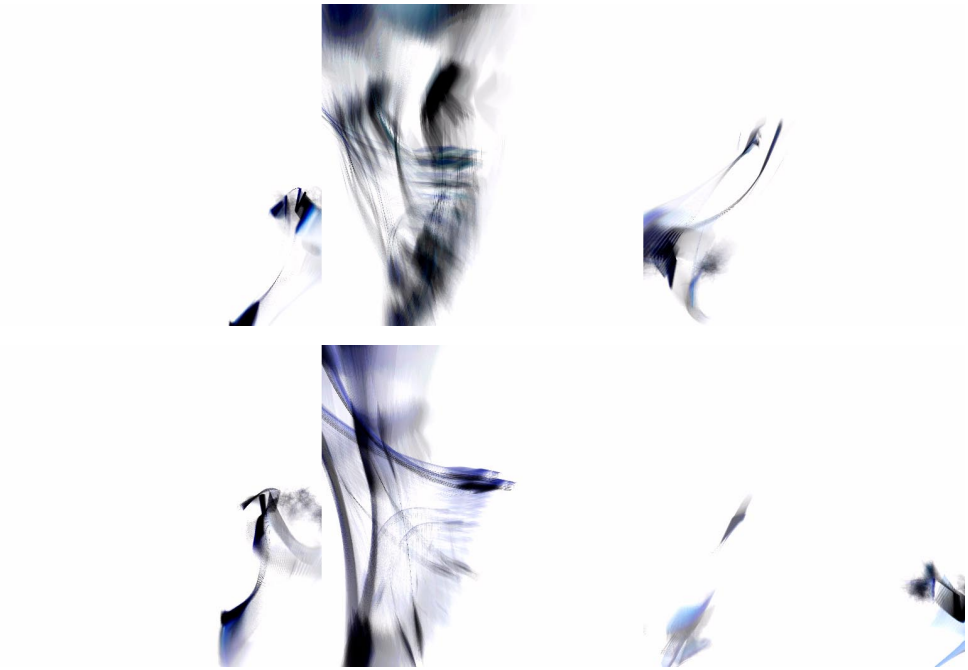


figure 44.two stills from *exchange:listen:exchange*

figure 44.*listen* dancing frenetically

curve (technology)

introduction

curve is an platform for experimenting with characters with bodies made not with triangles but with lines. Up until this project the perceptual worlds that characters inhabit focused on external phenomena – distances from objects, sounds, etc. – rather than movement directly. *curve* seeks to help create line drawn characters that perceive and respond to spatial configurations of other such creatures.

Other than a certain freshness of the aesthetic, concentrating on abstract lines as a body enables us to explore simple physical constraints with the introduction of 'spring physics' into the world. The graph-based motor system allows characters to understand the effect of physical constraints and physics based interaction on their bodies – albeit currently in very simple ways. Limited learning of static differences between `BodyPose` and pose realization can be offered by reblending `BodyPose`'s with contents of `BodyMemory`'s.

It is hoped that by moving along this path we can create visual creatures with an awareness of their own and others' shapes. This could extend the graph motor work towards creatures that learn through imitation.

The project's engineering includes a new `BodyPose` representation suitable for encoding 3 dimensional splines; spring system and optic flow based dynamics; and three new renderers for displaying these splines – either in the current 3 dimensional graphics system, a 2 dimensional Java 2D[10] based system, or direct to Postscript[11]. With the addition of the ability to read Postscript the animation tools now become incredibly simple – a piece of paper, a pen and a scanner.

---

10. Java2D: an application interface for drawing and imaging on screens for the Java programming environment. [38]
11. Postscript: a document interchange format, originally designed for interpretation by computer printers. [1]

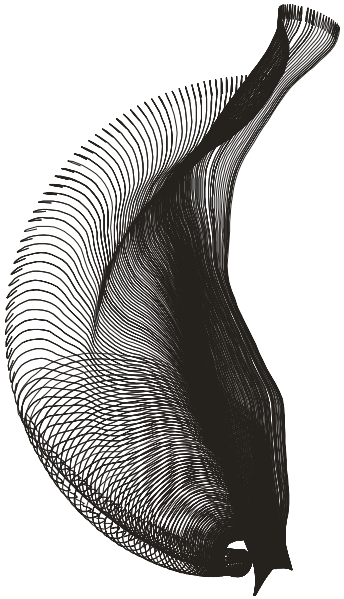


figure 45.example animation from *curve*

1. source material for this curve animation

figure 46.eight
creatures from
curve

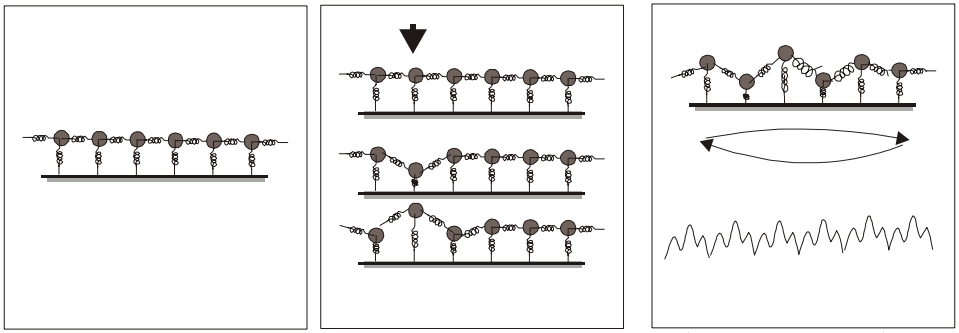


figure 47.scanned synthesis models

1. many scanned synthesis models start with a finite element system or a spring system.

2. forces can be applied to this model, which as vibrational modes below 15Hz

3. these structures are then "scanned" repeatedly at audio (kHz) rates

scanned synthesis models (technology)

However, there are musical applications of this line representation – we can introduce *scanned synthesis models* into our world. So far the installations described here have either dealt with raw audio or music made by off-board MIDI controlled synthesizers. Here we consider a direct connection between the characters and the synthesis technology.

Scanned synthesis models are a class of low-level synthesis algorithms for creating sounds with musically interesting timbres. Although the term was coined in [11] there are quite a few other synthesis techniques that could equally be described as involving "scanned synthesis" (e.g. [56]).

Such techniques are all interesting to us here because of their particularly *gestural* control. Indeed they are (perhaps retrospectively) motivated by considerations of such control mechanisms. Here we consider these techniques as an addition to the instruments we can place in the virtual musical worlds discussed above.

Scanned synthesis models rely on the simulation of a dynamical system with natural vibrational frequencies lower than 10-15Hz – for example, textbook finite element models of strings and sheets. This frequency cut-off is chosen to place the vibrational modes in the domain of haptic or gestural control – and we note that motor timescales are roughly the same as timescales on which musically interesting timbres change. These inaudibly low frequency patterns are turned into audible sound by repeatedly *scanning* the model along a particular path at audio rates.

artificial gestural control

The relevance here is that this is a synthesis algorithm (and one which is particularly cheap to evaluate) which has a direct and obvious interface with movement. What is lacking in this community is any tradition of creating, composing or working with artificially generated movements. Hence there is a obvious hybrid to be made between our character animation technology and this synthesis model – to connect characters' bodies to such a synthesis technique.

There are two ways that we can achieve this connection. Firstly we can have the nearby movement of a character's limbs affect a number of synthetic strings. We can do this by embedding both character and string inside a motion flow simulation – a particularly cheap fluid dynamics system which discretely simulate a dispersion and propagation of movement throughout a plane or space.

Secondly we can relate the body of a creature to the body of the string

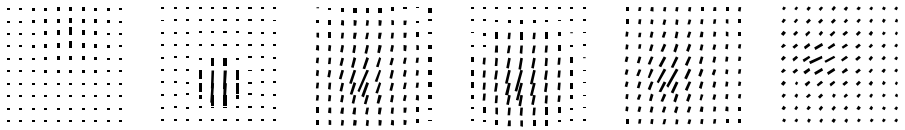


figure 48.motion flow fields generated by an animated limb

itself through the *curve* extensions to our system – either directly or through a spring system. Thus our body pose representations become timbral representations; our ways of orchestrating movement through pose space become techniques for changing timbre – the whole machinery, from animators tools through the behavior and motor systems gets repurposed for software synthesis.

beginnings and endings

Here we conclude this work, both by locating some of work that influenced it and by summarizing what has been achieved.

technical influences

This work draws from a diverse array of traditions. Several ideas have been borrowed from other implementations; several others have been more inspirational than proscriptive; still more are ideas that have been kept in mind during the development of the system, but are as yet only the horizon. Either way the heterogeneity of all aspects of this work is intentional – we have actively sought to build not dogma, but framework.

behavior-based AI

Perhaps the most important influence after our previous work is the *behavior based artificial intelligence* community of which we are a part. In general we agree with Brooks [14,16] that the deliberative AI tradition has failed to live up to its promises: promises of usefulness, especially when placed into dynamic, unpredictable, noisy, real worlds; and promises of insight into organic systems. We note that the interactive worlds that we wish to build with our system are, if they are anything, dynamic and unpredictable (they are, after all, *interactive*); and the creatures and conventions we seek to reference are biological. This difference in style is most succinctly expressed in the title of one of Brooks' papers – *Elephants don't play chess*. It should be clear by now that the chess playing abilities of our characters are irrelevant to our goals.

However, the architecture described here departs from the now traditional behavior based approach in many ways. The most significant departure is how we represent time – indeed the very fact that we represent time at all in our architecture differentiates us from mainstream reactive AI. In the original (and many contemporary) reactive architectures 'time' was never handled explicitly and temporal coherence left to become an emergent effect. The result: the creature, typically a robot, encountered each slice of time almost afresh.

Controlling an emergent phenomenon is hard and unnecessarily indirect. Such architectures make both actions which last sizable lengths of

time, and the representation of perceived objects (that are persistent over time), unnecessarily difficult to achieve. While this short-coming is understandable given the community (mainly robotics) that were the driving force behind this approach it was not clear how we were to build either musical creatures or virtual dogs without better tools to create temporal coherence.

traditional machine-learning

There is a long, rich tradition of machine learning techniques. A common theme is the desire to take other people's ideas and embed them inside our framework. Growing character complexity demands that we find a way of embedding the best of this tradition into characters in a way that lets the character address more open domains. Subtle, complex, and successful modelling technologies exist and nowhere more than here do we seek to reuse and repurpose other people's work.

Previously this statistical modelling was kept away from the core of the creature system. For example, previous installations required the statistical recognition of gestures performed with the participant interface [102]. This classification obviously had to happen in real time and the results relayed to the other characters in the world. But the approach there was to treat the recognition as an entirely separate process – encapsulated as a server of labelled gestures. This was by no means just an engineering fact – it was an enforced conceptual reality of the system. The variety of actions and the subtlety of the action selection was advancing beyond the perceptual intelligences that our creatures could support. As the development of (`void*`) progressed this decision made less and less sense. What was needed was a closer coupling between behavioral structures and gesture recognition models and processes. This lead to the question: why can't creatures do gesture recognition?

Throughout the development of this behavior system attention has therefore been paid to embedding such modelling techniques into the behavior system. Throughout there are other, clear inspirations taken

from other "limited domain" techniques. For example, we've seen shades of temporal difference learning, speech recognition and gesture recognition throughout this thesis.

deliberative AI

Some of these choices were also made with reference to deliberative AI. There are three aspects from this style that are of interest here.

- **explicitness**. Deliberative approaches can possess a clarity of statement lacking elsewhere. Formal logics or hybrids (for example, [34]) when packaged well are readable by humans. We have tried to share this clarity with the design of the action-tuple – a single statement of "when this, do this, until this". In deeply tangled trees of behaviors we can lose sight of this simplicity, and, in previous work, character authors have been driven to creating such impenetrable forests of behaviors. However, with the constrained format of the action-tuple, we have a formal mechanism for introducing new hypotheses into the creature – a useful mechanism is absent from arbitrary statements of formal logic. I hypothesize that this unexplored *middle-ground* between hard complex formal logic and amorphous low level behaviors will turn out to be fertile ground.
- **temporal logic**. The work of Pinhanez [69] (and Allen [2]) is an approach to constructing action representations that know about time and temporal relationships.We look to areas like temporal logic to verify and reassure us of the representational power of the structures that we have built.
- **computer music work.** Several musical works also serve a similar purpose – we seek to mimic their power but not to use them in their original domains. We can see structures like augmented transition networks in both the forward transition modelling extensions to the behavior architecture and the graph based motor systems and programs. These are used to musical effect most notably by Cope [21].

We also see shades of Desain and Honing's statistical models of rhythm induction [28] in our temporal models.

Finally, transcending categories (and section headings) are the seminal works by Minsky on the relevance of music to artificial intelligence and vice versa – [62,63] – his comments are topical and relevant today, some 20 years later.

interactive music

In addition to the above artificial intelligence work there are other interactive music techniques that are of interest here. In particular, there are two works that have consciously created music from embodied creatures – Pete Rice's *Stretchable music* [74] and Toshio Iwai's *music insects* [45]. Although both very compelling works, both lack the scope of a full character metaphor – neither of the graphical creatures involved have significant autonomy, internal state, perceptual abilities, or adaptation. However, our goal could be seen as the retaining of which works' transparency after the inclusion of these aspects.

origins of music

There has be recent resurgence of interest in what might be described as *biomusicology* – a field that seeks to give neurophysiological, psychological and evolutionary perspectives to the origins and purposes of music (for a review see [99]). What is music for? Why does every human culture posses music? What meaning can be given to the proto-musical structures found in animals?

Music from characters is in a unique position with respect to this work – work which looks at how music is grounded in emotion and behavior, music's communicative rôles and a willingness to consider animal behavior as containing some musical elements. In retrospect many of the "game like" installations detailed in this thesis seem to target an "animal level music". Of all the work in this field one idea that stands out as particularly relevant is that our human language competencies are in fact para-

sitic on our musical competencies. This justifies our study of the musical applications of our 'pre-verbal' synthetic characters, and, should we wish to create one day characters with language, characters with musical abilities are a necessary step.

We might also look to this literature to support my claim that music and movement share common competencies sufficient to merit a joint investigation [31]. Considerable anthropological evidence suggests that the tendencies of Western culture to regard movement and music as separate are recent and incorrect (e.g. the inseparable cultural rôle of dance and music in aboriginal societies). Studies conducted with children – who, for example, find it hard to sing without movement – supports this work [86]. Finally, in a more computational vein, we have work like [29] investigating the relationship between performance tendencies (here physical motion is used as a model for understanding the timing of the ends of pieces of music). Other work (e.g. [90]) goes further and seeks to explain other aspects of performance in terms of the functioning of the vestibular system, identifying such a connection as having *generative* influence on the structure of written music. Finally we might turn to the arguments presented in the gestural control community (most notably, [57]) and run them in reverse to suggest a functional connection between gesture and music production.

In this light, future work using the ideas contained in this thesis may complement biomusicology's almost exclusively top-down approach (for an exception see [92]).

previous work inside the group

The Synthetic Characters' Group have completed several large-scale installations in recent years and a variety of smaller sketches – the most relevant of these are discussed in detail throughout the text. The complete list reads: (in chronological order) *Swamped!*, SIGGRAPH '98 [9]; (`void *`): *a cast of characters*, SIGGRAPH '99 [7]; *sand:stone*, 7th New

York Digital Salon [95]; *sheep|dog: trial by Eire*, at the opening of MediaLabEurope.

The lessons learnt from these installations, of which I contributed to all but the first, run too deep in this work to be fully disentangled. One point worth noting: the creatures in (`void*`) totalled some seven thousand lines each, of a custom built language for describing characters. Despite the fact that the characters appeared very similar each character description was significantly different. These files consist of an intimidating number of numbers; character design often proceeded by changing these numbers.

While it is hard to compare characters from different installations, and the creatures from (`void*`) have not been reimplemented with the work here, the size of the behavioral atoms (now action-tuples) suggests that we can now make more complex characters with less work. While it is not true that we have no more numbers – we have values, filter constants, a free parameter in the action-selection mechanism, saliencies, etc. – we have the twin abilities to quickly raise character scaffolding and to *learn* what many of these more concrete numbers should be.

where this leaves us

We began with three problems of synthetic character design:

- our synthetic characters needed awareness of their own bodies.
- our synthetic characters were naïve about time and the temporal extent and location of actions. They were unable to represent or predict future states – fundamental to representing the passage of time.
- modelling, learning or adaptation were not well integrated into our characters – neither conceptually or in the engineering.

To approach these issues I introduced a new behavior system for synthetic characters; one with learning integrated into the core that has been shown to be robust in publicly displayed installations. This system can in-

corporate statistical models and can change complexity, in response to it perceptions, at run-time.

Through the action-tuple structure, this modelling capability provides sophisticated temporal relationships essential for the creation of musical structures. These models, which can learn from the action itself, provides an explicit representation of the temporal extent of actions. We can also reuse these models as models for adaptable parameterized actions.

The organisation of percepts which are containers of these models, into dynamically growing trees allows character authors to guide the creation of new action-tuple hypotheses. The learning capabilities allows these hypotheses to be tested and the value to the creature, learnt.

For these structures, a new action-selection strategy was proposed that exploits the information present in action-tuples to produce coherent and relevant behavior.

A variety of new models and techniques for incorporating textbook models were presented that fit well into the framework. These extend the capabilities of the creatures. All of these models have been used to some extent in the production of the installation work for this thesis.

A reuse of the action selection strategy to synthesize perceptual abilities was suggested. This is particularly useful – it is important that the perceptual abilities of our characters are well matched to the abilities of their actions. What has not been discussed is an extension of the perceptual structures to allow a variable number of generic objects to be attended to by the same percept tree at the same time. This has not been discussed because it has not been required by the installations detailed here. This remains work-in-progress, but I believe that the action-tuple intermediate layer will play an ever more important rôle in representing a dynamically changing number of persistent objects.

Several mechanisms for generating and using motivational state in characters were discussed; some of which, including the expectation genera-

tion and the emotional tag learning, are uniquely easy to incorporate this framework; others provide a focus of attention for the action-selecting capabilities – potentially allowing strong motivational or emotional state to enable creatures to make decisions faster.

However, no unified approach to the modelling of emotion or the integration of the variety of ways of generating emotional content has been given in this thesis. Finding such an approach is a topic for future work, but it remains significant that we now possess an *over*-abundance of options and mechanisms for grounding emotional change in our characters – the elements are clearly there for a principled approach to motivation and emotion modelling.

Next we looked at the design of motor systems for synthetic characters – since there is no reason to have a complex behavior system if it can never be expressed by the characters.

We begin with an extension of some more traditional "blend-based" motor systems to solve some "simple shaping" problems – here we combine the example blending capabilities of our motor system with the parameterized action modelling of the behavior system. By further exploiting this modelling we demonstrated that we can use learning as an engineering tool (with the learning to look at things, page 116) – here to engineer an ability to move the head to correctly attend to objects.

However, these blend based motor systems ultimately fall short. In particular they offer us no insight into the "complex shaping" problem, nor do they offer a representation that is amenable to fine-level learning. To go beyond this a new class of motor system was developed using directed weighted graph structures. There we showed that there can be synthesis of animation creation and gesture recognition paradigms fast enough to support learning at run-time in our creatures. These graphs have the advantage that the dialogue with the behavior system can take place in terms of problem solving.

In this way, the behavior system is well shielded from the competencies of the motor system. "Desired poses" may be statically computed keyframes, small problems to solve, animations, complex motor programs or involve entire subsystems. We can change radically the style of graphics system (see *ozymandias*, page 181 or *curve*, page 196) and still maintain a behavior system that is capable of modelling and moulding the body of the creature.

Although these graph structures take far greater risks with the animator's source material, these graph structures also promote a new way of designing characters – one where the animator remains far closer to the development loop. Finally, not only does it becomes easier to enforce a small (but common) class of physical constraints automatically, but also when other constraints are seen to be violated there is something that the animator can do to prevent it happening again.

This approach, when fully hybridised with blending capabilities, and taken with the behavior system structures, points towards a day when the question most frequently asked of our characters – "but can it learn anything new?" – can be answered with a simple "yes."

In the future we will explore these motor systems further – it is imperative that they are incorporated into a large multi-person installation of the size of (`void *`). We have yet to really explore the recognition or storage capabilities of graph based motor systems. Here I expect this to have applications in learning by imitation and the recognition and learning of the meaning of body poses of other creatures. These are fundamental features of social behavior between animals. Social behavior between synthetic characters is now the next time scale frontier, and a rich area for musical exploration.

Next we turned to concentrate on the musical instantiations of this work. The overarching goal is to use the small structures and competencies as the fundamental building blocks of organised sound. In discussing the (`void*`) work, we showed the equivalence with the new structures, that

one can start with larger pre-made musical segments within this framework.

These installations are presented here to serve as a sketch of this future interactive music paradigm – that of character based music. The goal remains that of the introduction, how to create engaging, long term interactive music pieces. The reality is that the only things participants have a tradition of complex long term relationships with are, in the broadest sense, characters – they are our friends, our pets, and our stories. Characters are particularly appropriate for this new medium.

To conclude, I have attacked all three of the problems we initial started with. These are real, relevant problems; publicly shown works that people have found interesting were created to explore these areas; and at the end of the process, we find ourselves in a more interesting position – intellectually, artistically and technologically.

The promise of the behavioral and motor techniques developed here far exceeds what could be exhausted within the time frame of this thesis. So we have failed to rise to the challenge posed by the behavior system and the motor system for musical ends. However, it was an ambition for those musical ends that drove the conceptual content of those systems. The future therefore is exciting, and artifact-rich.

(quaternions)

Quaternions have been shown to be mathematical objects that are good at representing rotations. They do not suffer from the well known problems of Euler angles – problems that include "gimbal lock" from their singularity, and strange interpolation behavior. Quaternions are better discussed elsewhere and the world probably doesn't need yet another introduction to quaterions. But they are not well integrated into the literature or into engineering maths, so I'll introduce the notation for the concepts discussed in this thesis here.

Of particular interest here is how we interpolate between orientations – this forms the basis for almost all of our character animation capabilities.

preliminaries

First some definitions:

$$q = \{w, x, y, z\}$$

For which we can define a norm, from an inner product:

$$\|q\| = \sqrt{q \cdot q} = \sqrt{w^2 + x^2 + y^2 + z^2}$$

"Unit quaternions" have unit norms:

$$\|\hat{q}\| \equiv 1$$

They can always be written as:

$$\hat{q} \equiv \left\{ \cos\left(\frac{\theta}{2}\right), \underline{\hat{v}} \sin\left(\frac{\theta}{2}\right) \right\}$$

With $\hat{\underline{v}}$ a unit vector. Vectors can enter this representation as:

$$\underline{v} \rightarrow \{0, \underline{v}\}$$

And, obviously, may or may not be represented by a "unit quaternion". Quaternion multiplication is defined:

$$q_1 \equiv \left\{ \cos\left(\frac{\theta_1}{2}\right), \hat{\underline{v}}_1 \sin\left(\frac{\theta_1}{2}\right) \right\} \equiv \left\{ w_1, \hat{\underline{v}}_1 \sin\left(\frac{\theta_1}{2}\right) \right\}$$

$$q_3 \equiv q_1 q_2 \; = \; \{w_1 w_2 - (\hat{\underline{v}}_1 \cdot \hat{\underline{v}}_2), \, w_1 \hat{\underline{v}}_2 + w_2 \hat{\underline{v}}_1 + \hat{\underline{v}}_1 \wedge \hat{\underline{v}}_2\}$$

$$q^{-1} \equiv \{w, -x, -y, -z\} \;\; \text{if} \;\; q = \{w, x, y, z\}$$

$$q^{\alpha} \equiv \left\{ \cos\left(\frac{\alpha\theta}{2}\right), \hat{\underline{v}} \sin\left(\frac{\alpha\theta}{2}\right) \right\}$$

From this definition we can show that quaterions can rotate vectors using the following construction:

$$u \; = \; q \underline{v} q^{-1} \equiv q\{0, \underline{v}\} q^{-1}$$

An operation count shows that this is computationally more efficient that a 3x3 matrix representation.

interpolation

We can define a spherical linear interpolation (in the literature, *slerp*) between two unit quaternions:

$$\text{slerp}(q_1, q_2;\alpha) \equiv (q_2 q_1^{-1})^{\alpha} q_1$$

This has all the properties that we'd like it to have – it is a smooth, continuous, well-defined, parameterization of a geodesic (here, a great circle) on the 3-sphere on which unit quaternions live.

The above representation is great for analysis, but bad for practical use. Better is the extrinsic form:

$$\text{slerp}(q_1, q_2;\alpha) \equiv \frac{q_1 \times \sin((1-\alpha)\phi) + q_2 \times \sin(\alpha\phi)}{\sin(\phi)} \quad \text{with } \phi \equiv \text{acos}(q_1 \cdot q_2)$$

Using this definition of "slerp" we can built up a higher (third) order spline interpolant. Both interpolants are used in the motor system discussed in this work.

distance metric

Finally, we can define a distance metric for use with quaternions. ϕ from the discussion above is one posibility. Easier to handle is:

$$\text{distance}(q_1, q_2) = 1-(q_1 \cdot q_2)$$

By minimising this metric we can compute:

$$\tilde{\alpha} = \arg\min_{\alpha}\{\text{distance}(\text{slerp}((q_1, q_2;\alpha), p))\}$$

namely, what α takes this geodesic as close to quaternion p as possible. By working through the mathematics, we find the answer:

$$\tilde{\alpha} = \frac{1}{\phi}\text{atan}\left(\frac{q_2 \cdot p}{q_1 \cdot p}, -q_1 \cdot q_2\right)$$

We'll find use for this, and the distance metric, in the task of engineering and analysing `BodyPose`'s.

(video contents)

To complement this printed document, a video figure is presented. It contains:

> (void*)**: a cast of characters**. Post-SIGGRAPH video showing characters, interaction & demonstration. Music to video is one continuous piece 'written' by the interactive score system – although it may not be connected to the edited visuals.
>
> **sand:stone**. Video from premier gallery installation in the New York School of Visual Arts. Background music by earlier version of music creatures.
>
> **Isle of man's best friend**. Video showing Duncan (the dog) in action – here there is a training scenario, value learning, simple shaping and limited speech learning is demonstrated. Shown at the Game Developers Conference 2000.
>
> **Graph-based motor work**. Demonstration of problem solving within a graph based motor system. Duncan tries to get his nose near the red marker using a graph derived automatically from three hand crafted animations.
>
> **exchange:listen:exchange**. Short video showing two *exchange* creatures (left and right) in action, with a *listen* creature between. One continuous take.

It is my intention to make this material publically visable at the following address: http://www.media.mit.edu/~marcd/thesis/video.html

references

[1] Adobe Systems Incorporated. *PostScript Language Reference, third edition*. Reading, MA. Addison-Wesley Publishing Company. 1999.

[2] Allen, J. F. and G. Ferguson. *Actions and Events in Interval Temporal Logic*, in: Journal of Logic and Computation, vol. 4 (5), pp. 531-579. 1994.

[3] Bahn, C. Streams. information: http://www.music.princeton.edu/~crb/Streams/streams.htm

[4] Balaban, E. and O. Laske, editors, *Understanding Music with AI: Perspectives on Music Cognition*, Cambridge, MA, MIT Press, 1992.

[5] Baron-Cohen, S. ed. *Understanding Other Minds*, Oxford, OUP 1999.

[6] Billard, A. and Mataric, M. J., *A biologically inspired robotic model for learning by imitation*, in: Proceedings, Autonomous Agents 2000, Barcelona, Spain, June 3-7, 2000.

[7] Blumberg, B. *(void*): A Cast of Characters*. Proceedings of the conference on SIGGRAPH 99: conference abstracts and applications, p. 169. 1999

[8] Blumberg, B. and Galyean, T. *Multi-Level Direction of Autonomous Creatures for Real Time Virtual Environment*. In Proceedings of SIGGRAPH 95: New York, NY. ACM SIGGRAPH. 1995

[9] Blumberg, B. Swamped! *Using plush toys to direct autonomous animated characters*. Proceedings of the conference on SIGGRAPH 98: conference abstracts and applications, p. 109. 1998

[10] Blumberg, B., *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD. Dissertation, MIT. 1996

[11] Boulanger, R.. *Scanned Synthesis: An Introduction and Demonstration of a New Synthesis and Signal Processing Techniqu*e. In Proceedings of the International Computer Music Conference, 2000.

[12] Brand, M. and A. Hertzmann, *Style machines*. In: Proceedings of the 27nd annual ACM conference on Computer graphics, SIGGRAPH 2000.

[13] Brooks, R. A. *Intelligence without reason*. In Proceedings of the 12th International Joint Conference on Articial Intelligence, pages 569–595, Sydney, Australia, August 1991.

[14] Brooks, R. A. *Prospects for human level intelligence for humanoid robots*. In: Proceedings of the First International Symposium on Humanoid Robots (HURO-96), 1996.

[15] Brooks. R. A. *A robust layered control system for a mobile robot*. IEEE Journal of Robotics and Automation, RA-2(1):14–23, April 1986.

[16] Brooks. R. A. *Elephants Don't Play Chess*. In Robotics and Autonomous Systems 6: 3-15. 1990

[17] Bruderlin, A. and L. Williams. *Motion Signal Processing*. In: Proceedings of the 22nd annual ACM conference on Computer graphics 1995.

[18] Chaplin, C. *The Gold Rush*. (film). Charles Chaplin Productions, United Artists, 1925.

[19] Chowning, J. M., *The Synthesis of Complex Audio Spectra by means of Frequency Modulation*. in: Journal of the Audio Engineering Society, vol. 21, no. 7, pp.526--534. 1973

[20] Close, B. *MASS ensemble*. information avaiable : http://www.light-works.com/MonthlyAspectarian/1998/November/1198-03.htm

[21] Cope, D. *Computers and Musical Style*. Wisconsin, A-R Editions. 1991.

[22] Cormen, T. H., C. E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. Cambridge MA, MIT Press. 1990.

[23] Credo Interactive, Lifeforms (software). Information at : http:// www.lifeforms.com/

[24] Csound. *A software synthesis language*. information: http:// www.csound.com

[25] Cycling '74. *Max/MSP*. (Software) published by . information : http:// www.cycling74.com/index.html

[26] Dawkins, R. Hierarchical organization: A candidate principle for ethology. In: *Growing points in ethology,* ed. P. P. G. Bateson & R. A. Hinde. Cambridge University Press. 1976

[27] Dennett, D. *The Intentional Stance*. Cambridge, Mass.: MIT Press. 1987

[28] Desain, P., and , H. Honing. *The quantization problem: traditional and connectionist approaches*. In M. Balaban, K. Ebcioglu, & O. Laske (eds.), Understanding Music with AI: Perspectives on Music Cognition. 448-463. Cambridge, MA. MIT Press. 1992. available: http://www.nici.kun.nl/mmm/papers/dh-96-e.html

[29] Desain, P., and H. Honing. *Physical motion as a metaphor for timing in music: the final ritard.* In: Proceedings of the 1996 International Computer Music Conference. 458-460. San Francisco. ICMA

[30] Discreet Software, *3D Studio MAX V3* 1999. Information http://www.discreet.com

[31] Dissanayake, E. *Antecedents of the Temporal arts in Early Mother-Infant Interaction.* In [99].

[32] Dolson, M. *The Phase Vocoder: A Tutorial, Computer Music Journal*, Volume 10, Number 4, pp. 14 - 27, MIT Press, 1986.

[33] Ekman, P. *Emotion in the human face. 2nd Edition*. Cambridge University Press, Cambridge, UK. 1982

[34] Funge, J., Tu, X. and Terzopoulos, D. *Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters*. In Proceedings of SIGGRAPH 99, 29-38.: New York, NY. ACM SIGGRAPH. 1999

[35] Game Developers' Conference, San Jose, CA . 1999. information: http://www.gdconf.com/1999

[36] Gleicher, M. *Retargetting Motion to New Characters*. In: Proceedings of the 25th annual conference on Computer Graphics. 1998.

[37] Goldberg, D. and M . J. Mataric. *Augemented Markov Models*. USC Institute for Robotics and Intelligent Systems Technical Report IRIS-99-367. available at : http://www-robotics.usc.edu/~dani/amm-tr.pdf

[38] Gosling, J., B. Joy, G. Steele. *The Java Language Specification*. available online : http://java.sun.com/docs/books/jls/html/index.html

[39] Grand, S., D. Cliff, and A. Malhotra. *Creatures: Artificial Life Autonomous Agents for Home Entertainment*. Proceedings of the Autonomous Agents '97 Conference. 1996.

[40] Hodgins, J. K. and N. S. Pollard, *Adapting Simulated Behaviors For New Characters*. In: Proceedings of the 24th annual ACM conference on Computer graphics 1997.

[41] Hopper, E. *Nighthawks*. (painting) The Art Institute of Chicago; Oil on canvas 30 x 60 ins.1942.

[42] Horner, A., J. Beauchamp, and L. Haken. *Machine Tongues XVI: Genetic Algorithms and Their Application to FM Matching Synthesis*, Computer Music Journal, Volume 17, Number 4, pp. 17 - 29, MIT Press, 1993.

[43] Hsiao, K. and J. Paradiso, *A New Continuous Multimodal Musical Controller Using Wireless Magnetic Tags*. Proc. of the 1999 International Computer Music Conference, October 1999, pp. 24-27.

[44] Intel, Inc. *Intel® Recognition Primitives Library V4.0 Documentation*. Santa Clara, CA.: Intel Corporation. 1998 Available at http://developer.intel.com/vtune/perflibst/RPL

[45] Iwai, T. Music Insects. Permanent collection at the Exploratorium, San Francisco. 1992. information: http://www.iamas.ac.jp/~iwai/artworks/music_insects.html

[46] Jackendoff, R. and F. Lerdahl, *A Generative Theory of Tonal Music*, Cambridge, MA, MIT Press, 1985.

[47] Johnson, M. P. *Multi-Dimensional Quaternion Interpolation*, In ACM SIGGRAPH99 Conference Abstracts and Applications, page 258. New York, NY. ACM SIGGRAPH. 1999

[48] Johnson, M. P., A. Wilson, B. Blumberg, C. Kline, and A. Bobick. *Sympathetic interfaces: using a plush toy to direct synthetic characters*. Proceeding of the CHI 99 conference on Human factors in computing systems. 1999

[49] Kline, C. and Blumberg, B. *The Art and Science of Synthetic Character Design*. In Proceedings of the AISB 1999 Symposium on AI and Creativity in Entertainment and Visual Art, Edinburgh, Scotland. 1999

[50] Kline, C. *Observation-based Expectation Generation and Response for Behavior-based Articial Creatures*, M.S. Thesis, MIT 1999

[51] Kohonen, T. *Self-Organizing Maps*. Berlin. Springer-Verlag. 1995.

[52] Kronland-Martinet, R. *The Wavelet Transform for Analysis, Synthesis, and Processing of Speech and Music Sounds*, Computer Music Journal, Volume 12, Number 4, pp. 11 - 20, MIT Press, 1988.

[53] Lansky, P. and K. Steiglitz. *Synthesis of Timbral Families by Warped Linear Prediction*, Computer Music Journal, Volume 5, Number 3, pp. 45 - 49, MIT Press, 1981.

[54] Laszlo, J., M. van de Panne and E. Fiume. *Limit cycle control and its application to the animation of balancing and walking*. In: Proceedings of the 23rd annual ACM conference on Computer graphics SIGGRAPH 96. 1996.

[55] Latombe, J-C. *Robot Motion Planning*, Norwell MA, Kluwer Academo Publishers. 1991

[56] Longo N., *Gesture Synthesis Research and Development*. document online: http://www.cesiumsound.com/GestureSynthesis.html

[57] Marrin Nakra, T. I*nside the Conductor's Jacket: Analysis, Interpretation and Musical Synthesis of Expressive Gesture*. PhD dissertation, MIT, 2000. available: http://www.media.mit.edu/~marrin/HTMLThesis/Dissertation.htm

[58] Marsden, A. *Representing Musical Time: A Temporal-Logic Approach*, Lisse, Swets & Zeitlinger. 1999

[59] Mataric, M. *Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior*. in: Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents, 9(2-3), H. Hexmoor, I. Horswill, and D. Kortenkamp, eds., 1997. available: http://www-robotics.usc.edu/~maja/bbs.html

[60] Mataric, M. *Interaction and intelligent behavior*. MIT Articial Intelligence Laboratory Technical Report AI-TR-1495, Massachusetts Institute of Technology, 1994.

[61] Midi Manufacturers Association. *General MIDI Level 2 Specification*. available: http://www.midi.org/about-midi/gm/gm2spec.htm

[62] Minsky, M. *Music Minds and Meaning*, MIT Artifical Intelligence Laboratory. Internal Memo 616, MIT Cambridge, MA, 1981.

[63] Minsky, M. *Steps towards Articial Intelligence*. Proceedings of the IRE, 149, 8–29. 1961

[64] Minsky, M. *The Society of Mind*. Simon and Schuster, New York, 1986.

[65] Palindrome Inter-media Performance Group. Information: http://our-world.compuserve.com/homepages/PALINDROME/

[66] Paradiso, J. *American Innovations in Electronic Musical Instruments*, article in the "New Music Box" monthly online periodical of the American Music Center, October 1999. available: http://www.newmusicbox.org/third-person/index_oct99.html

[67] Paradiso, J., K. Hsiao and A. Benbasat. *Musical Trinkets: New Pieces to Play*. In Conference Abstracts and Applications, SIGGRAPH '00. ACM Press, page 90. 2000

[68] Perlin, K. and A. Goldberg. I*mprov: A System for Scripting Interactive Actors in Virtual Worlds*. Computer Graphics; Vol. 29 No. 3. 1996

[69] Pinhanez, C. S. *Representation and Recognition of Action in Interactive Spaces*. PhD diss. MIT. 1999.

[70] Premack, D and G. *Woodruff: Does the Chimpanzee Have a Theory of Mind?* Behavioral and Brain Sciences 1 no.4. 1978

[71] Pryor, K. *Clicker Training for Dogs*. Waltham, MA.: Sunshine Books, Inc. 1999

[72] Rabiner, L. and Juang, B. *Fundamentals of Speech Recognition*. New York, NY.: Prentice-Hall. 1993.

[73] Reeves, B. and C. Nass. *The Media Equation*. Cambridge: Cambridge University Press. 1996

[74] Rice, P. Stretchable Music: *A Graphically Rich, Interactive Composition System*. S.M. Thesis, MIT 1998

[75] Rose, C. Verbs and Adverbs, *Multidimensional motion interpolation using radial basis functions*. PhD dissertation. Princeton 1999.

[76] Rose, C., B. Guenter, B. Bodenheimer and M. F. Cohen. *Efficient Generation of Motion Transitions using Spacetime Constraints*. In: Proceedings of the 23rd annual conference on Computer graphics 1996.

[77] Rose, C., M. F. Cohen, and B. Bodenheimer. V*erbs and adverbs: Multidimensional motion interpolation*. IEEE Computer Graphics and Applications, 18(5):32–41, September/October 1998.

[78] Russell, J. *A circumplex model of affect*. Journal of Personality and Social Psychology, 29:1161-1178. 1980

[79] Scherl, R. and Levesque, H. *The frame problem and knowledge-producing actions*. In Proceedings of AAAI-93. Menlo Park, Calif.: AAAI Press. 1993.

[80] Schodl, A., R. Szeliski, D. H. Salesin, I. Essa. *Video Textures* . In: Proceedings of the 27th annual ACM conference on Computer graphics 2000.

[81] Schoner B., C. Cooper, C. Douglas, N. Gershenfeld. *Data-driven Modeling of Acoustical Instruments*. Journal for New Music Research Vol 28(2), 1999. available online: http://www.media.mit.edu/~schoner/papers/JNMR.ps

[82] Shoemake, K. *Quaternion calculus and fast animation*. SIGGRAPH Course Notes. 10:101-121. 1987. (Note that this paper no longer seems to be available)

[83] Side Effects Software. *Houdini 4.0*. January 2000. Information: http://www.sidefx.com/

[84] Sims, K. *Evolving Virtual Creatures*. In: Proceedings of the 22nd annual ACM conference on Computer graphics 1995.

[85] Spier, E.From *Reactive Behaviour to Adaptive Behaviour* DPhil thesis. Balliol College, Oxford. 1997. available at: http://www.cogs.susx.ac.uk/users/emmet/thesis_abs.html

[86] Suliteanu, G. *The tole of songs for children in the formation of musical perception*. In : The performing arts: music and dance. J. Blacking and J.W. Kealiinohomoku, Eds. Holland, Mouton. 1979.

[87] SuperCollider. *A software synthesis language*. information: http://www.audiosynth.com/

[88] Sutton, R. S. and A. G. Barto *Reinforcement Learning: An Introduction*. Cambridge MA, MIT Press, 1998

[89] Thomas, F. and O. Johnson. *The Illusion of Life: Disney Animation.* New York: Hyperion. 1981

[90] Todd, N. P. M. V*estibular Feedback in Musical Performance: Response to Somatosensory Feedback in Musical Performance*. Music Perception. 10(3), 379-382. 1993

[91] Todd, P. M. and D. G. Loy eds. *Music and Connectionism*. Cambridge, MA. MIT Press. 1994

[92] Todd, P. *Simulating the Evolution of Musical Behavior*. In [99].

[93] Tomlinson, B. *Interactivity and Emotion through Cinematography*. M.S. Thesis. MIT 1999

[94] Tomlinson, B., B. Blumberg, and D. Nain. *Expressive Autonomous Cinematography for Interactive Virtual Environments*. Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Catalonia, Spain. 2000

[95] Tomlinson, B., M. Downie, A. Benbasat, J. Wahl, W. Stiehl. 1999. "*sand:stone - Artist Statement.*" Leonardo Vol. 32, No. 5, p. 462-463

[96] Tu, X. and D. Terzopolous. Articial fishes: *Physics, locomotion, perception, behavior*. In: Proceedings of the 21st annual ACM conference on Computer graphics SIGGRAPH 94, 1994.

[97] Unuma, M., K. Anjyo and R. Takeuchi. *Fourier Principles for Emotion-based Human Figure Animation*. In: Proceedings of the 22nd annual ACM conference on Computer graphics 1995.

[98] Velasquez, J. *A computational framework for emotion-based control*. In Proceedings of the Grouding Emotions in Adaptive Systems Workshop, SAB '98, Zurich Switzerland, 1998

[99] Wallin, N. L., M. Björn, and S. Brown. *The Origins of Music*. Cambridge MA. The MIT Press. 1999

[100] Watt, A. and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. ACM Press, 1992.

[101] Wilkes, G. *Click and Treat Training Kit Version 1.2*. Mesa, AZ. 1995. more information : www.clickandtreat.com.

[102] Wilson, A.D. and A.F. Bobick. *Nonlinear PHMMs for the Interpretation of Parameterized Gesture*. Computer Vision and Pattern Recognition, 1998.

[103] Yoon, S. -Y. *Affective Synthetic Characters*. Ph D. Dissertation Department of Brain and Cognitive Sciences, MIT. 2000

[104] Yoon, S.Y., B. Blumberg, G. Schneider. *Motivation Driven Learning for Interactive Synthetic Characters*. Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Catalonia, Spain. 2000

[105] Yoon, S.Y., R. C. Burke, B. M. Blumberg, G. E. Schneider. *Interactive Training for Synthetic Characters*, AAAI 2000.