Appendix

Appendix 5-1: EarBenders Stories

Title	Story
marimba keys	OK; Am I first? There seems to be something about short term memory that makes it easier to compare 2 sounds one after the other; than it is to compare 2 visual objects (distances) one after the other. your brain remembers the sound more accurately than the distance. e.g. If you show me a marimba key; then hide it; then show me another; I would have some difficulty deciding which was longer. But if you played it; then played another; i would have no trouble. Could be a problem if you are tone deaf! Chris
phone clues	police sometimes use the background sounds heard in a phone call to locate the phone box where the call was made - for example near a railway station or a restaurant may have unique background sounds ....
call to prayer	churches have a bell that is rung on sundays as a call to prayer mosques have loudspeakers through which the call to prayers is sung
school bell	in schools a bell or siren is often used to call the students to an assembly or to announce the end of lunchtime
squeaky boards	the bedroom of the japanese emperor was made with flor-boards that squeaked so that he could hear anyone who might be sneaking up on him in his sleep
tapping shoes	there was a medieval fashion of shoes with heels that tapped on the group to attract attention to the wearer
windscreen wipers	the windscreen wipers start squealing when there is not enough rain sprinkiling on the windscreen and this tells you to slow them down or turn them off
geese guards	roman armies placed geese on the camp periphery at night to warn against sneak attacks - the geese would reat to a disturbance by loud honking calls
irritating chewing	the sound of someone chewing gum and blowing bubbles in the seat behind me on the bus nearly drove me crazy - sounds can be very irritating - another one is someone eating an apple.
healthy car	I am always sensitive to the “sound” my car makes. When the car is “healthy” I don’t hear any noises; but if a new noise develops I will immediately hear it and think “oh no; what’s wrong now...”
mechanics with stethoscopes	I am sure many mechanics use sound to diagnose engine problems; in fact most mechanics have stethoscopes.
the car engine has a familiar ‘normal’ sound	Example in mechanics. The engine of my (rather old) car makes a complex but predictable ‘sound’. Just recently probably due to the cold weather there has been a new component added when the engine is cold. This is a bit of a worry; because if its not caused by the cold it means something might be getting ready to break ! Now that I think about it there are numerous examples of ‘normal’ sounds where a change from the ‘normal’ pattern triggers awareness.
Dropped item	Last night Matt and I were removing a perspex thingey-me-jig off the car window and we kept dropping bits and pieces (cause it was cold and dark); but we found every piece because we heard where it fell.
Baby talk	Also; I always know what Keely wants from the different cries she gives (ie. hungry; tired; uncomfortable; lonely; wet or bored). You can also tell different things from her happy sounds. Baby’s are very expressive with sounds.
Games - Myst Space Ship Tune Puzzle	have you noticed that the computer game Myst gives sounds as major part of the game. Also to get to other worlds you may need to use and manipulate the sounds you have discovered into a sequential pattern. Some can be tricky (cause the sounds sound similar; but are different ie. on the organ in the spaceship).

Title	Story
Games - Myst Elements KlangFarben Puzzle	have you noticed that the computer game Myst gives sounds as major part of the game. To get to other worlds you may need to use and manipulate the sounds you have discovered into a sequential pattern. Some can be tricky (cause the sounds sound similar; but are different ie. on the organ in the spaceship).
Games - Myst Train Navigation Puzzle	have you noticed that the computer game Myst gives sounds as major part of the game. They are used for direction guides (ie when moving around the maze etc).
Overloaded electric tools	Here is something maybe. When using an electric drill one would like to know when it is being overloaded. Presumably tradesmen learn by experience with particular tools what a cry of pain sounds like; but not the average citizen. Similar of course for the redline rpm in cars.
Getting to sleep	Dear Stephen; Just one point about noise. I think that noise in a modern society is something that you are more likely to want to blot out than listen to. I live in a flat; and when I first moved in it took ages before I could sleep without being woken up by noises around me. I wired my micro-moog (love that machine) into a really BIG speaker (12" woofer and all that) with white noise and left it on 24 hours a day in my bedroom. The result was a peacefull and blissful nights sleep (after I sto
Waking up	You are interested in presenting noise as something useful. It might be interesting to explore the other side - that noise is sometimes a penalty - indeed it is designed that way to wake us up before being mauled my sabre-tooth tigers and the like. I believe you have the same problem with Bettina listening to those story tapes about "and then the horse jumped the gate; and raced out of site over the hill .."? You have my symphony .. oops; I mean my sympathy.Good luck with the write-up. Scott.
striking matches	striking matches I can hear how dan is doing it - the pressure of the match head on the box - the fluidity of motion - the flare as it lights, even though concentrating on another activity with my other son timmy
Estimating Numbers of Frogs or Cicadas	estimating frog populations by listening to tape recordings made of ponds at night, estimating the numbers of cicadas in the vicinity in the middle of the day, the number of animals contributing to the sound changes the loudness and timbre of the overall sound
Species identification	identifying frog species by tape recordings made by ponds at night, identifying cicada species by their distinctive chirpings,identifying bird species from their calls
Dishwasher	gone to bed and have turned on the dishwasher - can hear if something is going wrong. like bad stacking of the plates, or the water tap is off.
shaving	shaving with electric razor - starts off noisy but is finished when it goes quiet as hairs get chopped
Dropped object	finding dropped screws; coins; tools
Car diagnosis	renault manual - gear; brakes; transmission; engine diagnosis
Climbing Vines	life on a vine - sbs tv - thais climbing cliffs for birds eggs pull on a vine and listen to hear a 'bad' one
finding people	someone is there - physical presence,finding people in the rubble of collapsed buildings
finding people	finding people in the rubble of collapsed buildings or in the bush
outside tap	tap is on or off, tap is running, hose outside is running
footsteps	a persons walking style, recognise someone by their footsteps
visitors	a visitor has arrived - a car coming up the drive; people on the porch; doorbell - housemate has arrived has key in the lock instead of doorbell.

Title	Story
Syncronous revs of plane engines	difference between engine revs - pilots use beating between engines to adjust revs to be synchronous for comfort since the beating is annoying
Door key card	door key-card reader beeps on error
PIN keypresses	beep on successful keypress keypress recognised - eftpos & autobank input beep
pick up card	autobanks sometimes double beep to remind you to retrieve your card at the end of a session
Bikes in Traffic	When you are riding your bike you use your hearing all the time - Crossing blind intersections. - Pedestrian road crossings. - Trucks, cars traffic behind your bicycle.- Bicycle coming along footpath
bells and horns	bike bells and car horns indicate that someone is trying to get your attention - could be telling you to watch out, or saying hello as they drive by - depends on the situation...
bicycle maintenance	bike maintenance - chain lubrication; getting gears in; bell ringing; pinging spoke; pin in the tyre; puncture
cat talk	cat-talk - want food; get off my tail; let me out; pat me; purring with contentment
rafting rapids and waterfalls	rafting rapids ahead, waterfall is nearby
weather	wind gusts roaring toward tent thru trees at island bend (snowies). rain bursts on tin roof; absence of noise when its snowing.
sickness	coughing = disease (stay away) - a person is sick with the flu and they sniff or blow their nose
siren	air raid siren at 12 noon = first monday of the month in holland (in the 70s?)
timer alarm	alarm clock; oven timer; egg-timer
door muffle	when the lounge door is open the tv and conversation is loud; when it is closed it is soft
window or door is open	an open window in the car whistles, when the back screen-door is unlatched it flaps in the wind
Coffee percolator	the coffee percolator makes a burbling sound while it is pumping through the filter and then a louder brighter burble when it is finished, then eventually the burbling stops
boiling water	hot water kettle boiling starts off silent, then starts bubbling with a low dull sound and gradually gets higher and brighter until it reaches a stable boiling state. You can usually tell where it is up to in the sequence and how much further there is to go - e.g. 10 seconds or 1 minute etc.
operating bug in the burners	“Thumping” noise in the roof above the lobby corridor level 3 is from the plant room above. This noise is from incorrectly adjusted gas burners on the boilers. The burners have some operating bug which prevents correct adjustment and causes burner dropouts. Turning the boiler off fixes the noise but causes severe shivering to occupants. The suppliers have been working on the problem and have promised to have the bug fixed by Friday this week ?? Arch
Rubbish	I mentioned my shaving use-of-sound the other day. I was talking to my partner Robyn about your stuff; and she had a good one as a scavenger. She said the quickest; best way to see if a bag of garbage contains something interesting is to give it a bit of a kick. The combination of the weight of the bag and sound that it makes tells you what’s inside. Normal household rubbish sounds a bit plasticy; bottles clink; crockery clatters; disposable nappies squelch.... So a lot of the time her job involves wanderin
Voice identification	identifying a criminal by voice in a police line-up

Title	Story
car noises fit like a glove	car noises fit the car like a glove - most powerful diagnostic in mechanics toolkit. jon lilleyman
Helicopter Gearbox	Helicopter gearboxes are very difficult to remove and have stringent maintenance requirements. Maintentance diagnosis is done using a stethoscope to listen for worn gears, broken teeth etc.
Airspeed Indicator	aviation legend - after ww2 when cockpit instruments were hard to get glider pilots strapped a harmonica to the wing as a stall warning device - no music / no fly !
stall warning	contemporary cockpits use a buzzer to alert the pilot of an imminent stall
slow down warning	speed stripes on the road - rythmic tyre sound warns you to slow down
cereals	Dear Ear-Benders; I often rattle things (particularly in the kitchen) to see what's in 'em. This can give visual; muscular and auditory feedback. The other day I had two opaque film canisters; one with 200 speed film and the other with 400 - so I rattled them to find the one I wantedbut they both sounded the same
dog talk	Like Sharri's baby; my dog is quite expressive with noise. I have grown particularly alert to her (the dog's) "I'm throwing up on the carpet now" sound. Matthew.
footsteps	absence of noise - as in when the footsteps stop outside andrew lochley's door which makes him cringe because someone wants something
aural debug by disc access clicks	OK; I have two more 1/ Computer programmers used to be able to tell when there was in infinite loop in their program by listening to the disk accesses - a regular click ...click ...click ...click ... where there was none before was not considered a good sign.
internet break in alert	2/ My notebook is on the Internet with no fire-walling; but I know that I will hear the disk spin up if someone tries to break in (assuming that I am in my office of course).
filling bottles	Whenever I have to fill an opaque bottle with liquid I listen to the sound that the liquid makes and the pitch tells me how much liquid is in the bottle. This is particularly handy when filling hot water bottles for the kids. It means you can pour the hot water full blast out if the tap and not have too great a risk of scalding yourself. Cheers Don B.
jungle war	jungle warfare in a movie called "the last bullet" - a japanese soldier remembers "by the time you see your enemy it is too late - you must listen to the sounds"
faulty dishwasher	washing machine mechanic - first thing he does is listen - diagnosed faulty relay - matthew
pedestrain crossing beeper	when the traffic light starts beeping it means you can walk across the pedestrian crossing
phone	telephone you are calling is engaged or ringing or dead, modem monitoring - ringing; answer or engaged
depth of a hole	drop a rock down a hole to tell how deep it is - neale
lightning distance	Distance of lightning by counting til thunder arrives
fishing cast	fishing cast - a good cast makes a nice whizzing sound, and you can tell how far it has gone by the amount of time it whizzes before it "plonks" into the water. Also you can hear where it landed.
horse	how fast is a horse going - clippitty clop; galloping etc.
car speed	you can tell how fast a car is as it passes by by the rate of change of the location of the sound, the loudness and timbre of the engine; and for racing cars you hear a change in pitch as it passes by due to doppler shift as well.

Title	Story
hitchhikers guide	>But I still think the Earth got blown up before Ford's house got 'dozed! >Richard. In the book (and radio play I believe); the front of Arthur's house gets bulldozed while he is in the pub with Ford Prefect. The crashing noise prompts him to rush outside and back to the house (with Ford running after him). In the TV series - due to low budget no doubt - you just see the front fence being knocked over. Of course the Vogons then destroy the earth; which I suppose completes the job of knocking down Arthurs ho
CLC stuffed	The CLC still appears to be stuffed! i.e. It does not print anything and seems to be recognising only black. It sounded very rough as well.
air rescue beacon	in an air rescue an emergency radio search beacon transmits a tone at a certain radio frequency and searchers home in on it by loudness of the radio signal - neale
ice floe	the condition of a glacier ice floe is learnt by nearby residents who listen to the cracking noises it makes. neale
power station	maintenance men in power stations can hear the possibility of the transformer exploding and have been known to make a quick exit when things start sounding abnormal - neale
microwaving	microwave food pops and crackles when hot enough
gas leak	when the gas is on or leaking in the stove it makes a hissing sound.
gas cylinders	which gas cylinder has more in it ? you tap them and the fuller one has a low dull sound and the emptier one is higher and has a brighter hollower tone.
tracking marbles	18 month old boy tracking marbles by ear on the wooden floor when he cant see them because they are behind him - neale
Lorikeets	rainbow lorikeets calling at Batemans Bay golf-course reminded me of holidays 20 years ago when my family camped next to a bird sanctuary at Curumbin
Seagulls	seagulls squawking at Lake Gininderra makes it feel like being at the seaside
a good hit	a good golf hit or tennis hit sounds "sweet". You can improve your game with this feedback, or anticipate the speed of an opponents tennis return.
barcode scanner	when the librarian or supermarket checkout pass the barcode scanner over an item it beeps to signal "ok"
alarms	fire alarm; burglar alarm; car alarm
sirens	police; ambulance; fire truck sirens
dripping tap	a tap or shower is dripping - you hear this especially when trying to go to sleep at night, even when it is out in the laundry.
road surface	different road surfaces make different sounds as you drive over them - concrete roads make regular clacks due to the cracks; asphalt is even but noisy, dirt has loud thumps.
radio phone	The remote phone can be paged from the base so you can find it under a cushion in the lounge or amongst pots and pans on the kitchen bench.
doctors with stethoscopes	I am sure many mechanics use sound to diagnose engine problems; in fact most mechanics have stethoscopes. Now that leads on to the medical profession and how they use sound to diagnose a patient...
Engine Load	Another motoring example With familiarity, I can tell the load being applied to the engine of my car. The 'load level' can be sensed even at constant speed. It must be a change in harmonic content since the basic frequency is speed related. This principle could be used as an audio system load monitor to replace the visual perfmeter !

Title	Story
surfing	surfing - you know that you have caught a good wave and are going fast because there is a harmonic vibration in the fins which you can hear when you exceed a certain speed
Getting up to attend a baby	Sometimes Keely wakes at night and I can tell from bed whether I should check on her or leave her to cry herself back to sleep.
applause count	in debates the winning team is determined by the amount of applause given each team by the audience
internal or external phone call	On our phones at work any internal phone calls have a single ring, while external phone calls are signalled by a double ring. Sometimes you are too busy to deal with an internal call so you ignore it.
driving in fog	Driving up Bulli pass in thick fog we had to cross 3 lanes of the Sydney expressway at a T-junction. Visibility was down to 30 m but by rolling the windows down and listening it was possible to hear invisible cars approaching and so make a less risky crossing.
squeaky dog toys	Dogs love to play with squeaky or rattly toys - these noisy toys keep their interest.
eggs and bacon	bacon and eggs sizzle and pop when they are frying - this is how you tell the heat is high enough
popcorn	when you pop popcorn it starts off slowly popping then builds up in rate and loudness, then dies down again as most of the corn has popped. you can tell how many popcorns are going off all through the cooking process, and you can tell when its finished
Baby identification	You can recognise the cry of your own baby from those of others in a group of babies.
in tune	you can tell if a guitar is out of tune by strumming a familiar chord - if ok it sounds sweet, otherwise it sounds rough
in tune	you can also tell if a guitar is out of tune by plucking a familiar chord from low to high as an arpeggio - there is a rgular pattern if it is ok, otherwise the out of tune string sticks out like a sore thumb and the arpeggio sounds “off”
micro-tuning	You tune a guitar by listening for “beats” between adjacent strings when the same note on each is plucked together. When the beats disappear the strings are in tune.
caving	when you are in the dark in a small tunnel there is no reverb and the sound is very “dry”, when you enter a larger room you automatically register the increased size of the room by the increase in the “wetness” or reverb of voice and dripping and scraping sounds
mowing the lawn	When you are mowing you can monitor the thickness of the grass by the loudness of the motor soundm plus you can tell that you are mowing over sticks, stones or other obstacles from the clackety sounds.
customer buzzer	Shops sometimes have a bell on the door or an electronic buzzer which goes off as a customer passes through the doorway of the shop. Even if the shop keeper is serving someone or is stocking the shelves elsewhere in the shop they are able to monitor comings and goings. Actually the sound will be the same for both coming or going, but the context is different - so you might discern a going by a connection with preliminary sounds such as footsteps or talking ... maybe coming and going are actually different c
wind chimes	Wind chimes make a pleasant realxing sound when the wind blows. However they are not usually used to inform that the wind is blowing but as a form of aural decoration. Other garden decorations that come to mind are fountains and bubbling brooks and japanese water knockers.

Title	Story
vacuum cleaner	What Ducted vacuum cleaner with 9 metre hose -Action suck suck suck, then something gets blocked (in fact the hose twists and kinks blocking off the air flow.-Sound pitch of whine of electric motor rises alarmingly as the load becomes excessive -Interpretation hose is blocked
hammering a nail	Sound solid sounding knock, then a softer slightly clangy sound -Interpretation Nail is bent -Sound solid sounding knock, then overly solid (higher pitch) -Interpretation Nail has hit something (pipe ? concrete ?)
meditation mantra	In meditation you mentally repeat a mantra sound (a word without meaning) to help focus your mind and distance your connection with the external world.
music to study by	It is common for students to listen to music turned up very loud when studying. Rather than being distracting this actually helps some people focus on their work (well it did for me !). I think it is helpful because it blocks out other distracting sounds (noises) such as the tv in the next room, or people talking etc which would likely lure you away from the textbooks...also it can help you settle into your study habitat, acting as a psychological conditioner like a bell causing a dog to salivate.
pavlovs dogs	Pavlov trained dogs to assosciate the sound of a bell with food by ringing it when they were given a meal. Once this conditioning had occurred the dogs would salivate at the sound of a bell, even though food was not present. A sound which does not naturally occur in the presence of food could trigger a reflex which is only useful for eating. This is strong evidence against the ecological premise that information is structured by the environment, and undermines some of the claims made for the Auditory Icon m
Computer	What Waiting for something to happen as per instruction -Sound nothing then a busy clicking of the hard disc heads -Interpretation Something is happening
laser printer maintenance	printing a page on the laser printer -Sound graunch, grind -Interpretation something needs maintenance, maybe oil
electric kettle	boiling water in the electric kettle -Sound rumble gets louder, bubbling, then ‘click’ -Intepretation auto switch-off when boiled
Raining Outside	in a house with a tin roof you can hear how bad the rain is outside - sprinkling is a soft pitter-pat, but when its raining hard it can be so loud it drowns out your voice
cocktail party	at a cocktail party you can choose to listen to the person you are conversing with, whilst monitoring another conversation elsewhere in the room, or follow someones voice or identify people and groups of people, all against the general hubbub of voices, music and other background noise which indicate the level of animation of the proceedings...
Diabetes advert	TV advertisements for diabetes campaign Do you remember this song ? “chewie, chewie etc.” then you are over 40 years old and need a diabetes test.
wind chime	my family in Queensland have tuned windchimes on their porch and when the wind blows in a northerly it plays a distinctly different chord to when it blows east.
tennis	when you play tennis in a pressurised dome you can’t hear the ball when you hit it and it makes playing a very strange becuase you can’t tell when the ball has been hit by your opponents.
macro-tuning	to tune strings play the same note on adjacent strings, one close after the other. If the pitch of the tuned string is too high then release the string tension a little and test again. If too low then increase the string tension. Keep doing this until there is no gross pitch difference. Now use the beats as in the micro-tuning story.

Title	Story
Irrigation	ACT Parks and Gardens is responsible for maintaining the green spaces in and around Canberra, which is a national capital with a strong tourist industry. Irrigation is a major investment and it is important to ensure that the system is in good working order and is performing efficiently. Satellite images can be used to assess vegetation growth and soil wetness. An image of greeness difference is overlaid over cultural features but it is difficult to show soil wetness on the same display. Can sounds be used
walking at night	walking home in the dark you can tell when you are walking over leaves, grass, concrete gravel etc. by the sound of your footsteps
walking speed	when walking home you can hear someone behind you apporaching as their footsteps get louder and also by the rate of their steps which is faster than your own
breathing	you can tell if someone is exhausted by their breathing - heavy, fast or shallow wheezy, - sharp short means sick, - you can't normally hear someone breathing if they aren't somehow distressed
flag in the wind	a flag flapping in the wind makes a louder faster flappier sound as the wind speed increases
ducted central heating	action - turn-up heat on thermostat, sound - slight delay then satisfying click, interpretation - relay has clicked and it will start soon
ducted heating 2	action - turn-up heat on thermostat, sound - distant soft 'explosion', interpretation - gas burner has switched on and heat will start soon
cold start	action - cold start of car, sound - engine clatter gradually settles down to a hum after 5 mins, interpretation - engine was cold
power saw	as the saw cuts into the saw the whine gets deeper in pitch ad duller, as you pull it out it raises in pitch to its normal whine again - I guess its something to do with the speed of the saw.
sprinkler system	small drip sprinklers make a hissing sound when they are operating properly and though you cant usually see them you can walk along the row of bushes and listen to make sure the waterinng is going ok at each bush
wolf whistle	a wold whistle is someone trying to get your attention at a distance in an outdoor environment
counting	a person can communicate a quantity using spoken numbers
objects = nouns	nouns are usually perceptually distinctive sounds, especially if commonly used in similar contexts - e.g. car, bike, van, bus, train, ute
japanese ladies	japanese women continually flush the toilet to cover other noises
relaxation tapes	Rain is natures steady, cleansing symphony - Rainfall in the Mist relaxation tape helps you relax and drift away to a place of beauty and serenity. Other tapes have ocean or waterfall scenes.
dog collar bell	walking my dog ziggy on Mt. Ainslie I stopped for a breather and Ziggy went off into the bush sniffing around. I couldn't see her but couyld tell she was moving away from me by the sound of the bell on her collar. In the end when she wouldne come I had to go and find her - I think she was lost !
cicadas in heat	cicadas only start to sing when the temperature rises above 18 degress celsius, though different species may vary several degrees
bumble bee	bumblre bees use a special buzzing key to open a particular type of flower
egg collectors	a tv documentary showed thai egg collectors who sail to remote islands and climb hundreds of metres up cliff-faces to reach bird nests using vines which grow there. The collectors pull on a vine and listen to the sound it makes before trusting it with their lives, apparently they can hear a good from a bad vine this way...

Title	Story
dropped keys	you can usually find tour keys straight away if you drop them in the dark by the sound of where they fell
camera shutter	you know you have taken the photo when the shutter clicks
camera timer	a camera timer makes a buzzing noise until it clicks off the shot
cassette rewind	as a cassette rewinds the sound gets faster and faster unti it clunks to a stop
phone dial tunes	when you dial a push button phone consecutive numbers have increasing pitch - with familiar numbers you know that you have mis-dialled because of the wrong tune
phone dial press	when you press a phone dialling button it makes a characteristic pitch that lets you know that you pressed it hard enough
broken cicada	a damaged male cicada cannot sing well and will not be able to attract mates
insects attract mates	crickets, cicadas and many other insects chirp, rub their legs together or hammer their abdomens to call mates. Each cicada species has a distinctive call and the females are very selective in their response so overlapping territories do not cause disruptions
insects frighten predators	beetles make explosions using gas to scare off predators, masses of cicadas chirp loudly (>100dB) to scare off birds
insects stake territory	as a grasshopper flies it clicks its legs to let others know that this is its territory
pair identification	a telecom technician in civic was using a beeping device to identify wire pairs between his location and a site across the road
continuity checker	my multimeter has a continuity option which makes a beep when there is no resistance between the leads
mice,rats or possums in roof	whilst staying at the coast house I heard something running around in the roof, maybe a rat or possum because it sounded bigger than a mouse
tuning a radio	when searching for a radio station you get a noise or whine which diminishes around the a station frequency which is a kind of notch in the noise band
microwave or oven timer	the microwave timer goes bing when the time has run out
collision	We may predict the potential collision of two objects by observing their paths with our eyes, but it is the sound of the collision that best reveals how the structure of the objects has been affected by the collision.
image projection	teenagers cruise around civic with their music blaring from their mobiles to project their particular image - hip-hop, gothic, thrash whatever...same with partioes and personal collections - play it loud to let people know who you are !
earcons	earcons are short musical motifs consisting of 2 or 3 changes in timbre, pitch or rhythm which are used to signal interface events such as opening a file or scrolling a menu.
seismic audification	seismic data consists of ? channels of continuous time series ratio pressure measurements from an array of pressure transducers buried in the ground. The analyst looks at visual traces to dentify event signatures which can be used to identify the time of unusual events and discriminate between nuclear tests and earthquakes at remote locations.
ripe fruit	you can tell whether a water melon is ripe by tapping it and listening, *this also works for apples, it must have something to do with the the damping of the sound when the fruit isn't ripe, because it sort of rings when its ripe*

Title	Story
songlines	youll hear the expression aquiring ritual knowledge. All this meant was that the man was extending his son-map...*The next point, he said, was to understand that every song cycle went leap-frogging through language barriers, regardless of tribe or frontier. A Dreaming track might start in the north-west, near Broome; thread its way through twenty languages or more; and go on to hit the sea near Adelaide. And yet, I said, its still the same song. ...Does that mean, I asked, that a young man on Walkabout c*
geese navigation	Certain ducks and geese can 'record' the choruses of frogs beneath them, and 'know' that they are flying over marsh.
birds migrations	songlines - *Other night fliers bounce their calls on to the ground below, and, catching the echo, fix their altitude and the nature of the terrain.*
dolphin triangulation	Dolphins flash echo-locating clicks on to submarine reefs, in order to steer a safe passage through..It has even occurred to me that, when a dolphin 'triangulates' to determine its position, its behaviour is analogous to our own, as we name and compare the 'things' encountered in our daily lives, and so establish our place in the world.
hula hoops	from the movie "the hudsucker proxy" *then we put a little bit of sand inside the hula hoop that shoooooshes to make it - well to make it more fun*
a clue	in a tv cop show *the hunted man brushes past a hanging mobile that tinkles as he passes throgh the doorway. He hides in an adjoining room and waits. Subtly we hear the tinkling of the mobile again and the man opens fire through the fibro wall into the next room, killin gthe assassin in the dioorway.*
oil drilling	sometimes there is a risk of accidentlaly intersecting another oil well when drilling for oil. One way of monitoruing the progress is to lower a microphone down the oil well at risk and listen ffor warning sounds whilst drilling
mr whippy	the mr whippy ice-cream van plays greensleeves and you can hear it coming several blocks away, and tell when its getting close to your house in time to get som echanges and track it down
troubadors	before the printing press the troubadors were paid to sing the news as they travelled the countryside - they were the source of news. they would trade songs with each other, and were able to remember songs that lasted hours after only one hearing
gold	Can you find the gold? It is hypothesised that six different aspects of the land in which gold may be found are determinative of whether or not gold is there. The first 20 data variables (each 6-d) are from sites known to have gold; the second 20 data variables are from sites known not to have gold. For each of the remaining 10 data variables, decide fro each whether or not it is from a site with gold.
entomology	when working at entomology arch was asked to build a listening device to detect weevils on the conveyor belt on their way into grain trucks. you could hear these insects rustling in the grain even though you couldn't see them - especially in a big silo.
finding studs	you can find where the studs that the plasterboard walls of the house are fixed to by knocking on the walls. the walls sound hollow *but where the studs are it is more solid and duller*
male spacing behaviour	As in the bladder cicada, most insects use the distant cue of sound to indicate their presence to would be competitors, and maintain a safe distance between agressors. If the males adopt this tactic, and the vegetation is homogenous, the males will space themselves quite evenly, and spacing will be a consequence of acoustic rivalry. But the distance at which the male calls may be under selection, depending on the number of males present, the intensity of the call, the hearing sensitivity of neighbouring ma

Title	Story
size and reproductive success	In those insects using sound as the primary long-distance cue for mate attraction, females do prefer larger, more loudly calling males e.g. mole crickets, bush crickets, fruit flies. Even under water intensity is an overriding factor infleuncing the behaviour of female corixids. Deep croaking frogs repeatedly get more mates than higher pitched rivals, and the louder calls of larger natterjack toads attrack more mates. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and
species pattern	Males call with a species pattern and the call must have a quite resticted frequency range. The female must recognise this signal as its own species, and its hearing system would do well to be tuned to the males call. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, London
aggregations of calling insects	Calling by aggregated insects became central to the group versus individual slection controversyof the mid 1960s where the group selection theory was that chorusing groups evolved as displays that allowed individuals to gain information as to the size of the population, and adjust their behaviour accordingly. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, London
oversinging and masking	Perhaps males use their song to mask the call of another to minimise the influence of an intruder on a searching female. Resident male O.nigripes over-sing their neighbours in this way. The song consists of a long series of buzzes, and with this style of song masking could be quite effective. The interval between buzzes is adjusted to fit over the intruder's song, essentially synchronisign the sound bursts. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, Lon
war propaganda	Amphiacusta maya is a phalangopsid cricket from Central America which occurs in groups in tree hollows. It is unusual because the males song has no calling function: females are not attracted to it. The primary role of the song appears to be for keeping other males away during copulation - war propaganda rathaer than courtship. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, London
insect thermometer	The snowy tree cricket produces its call as a nearly continuous sequence of chirps, with each containing 2-11 pulses, although 5-8 is more common. The chirp rate and structure of the chirp are highly temperature dependent, varying between 50 and 200 chirps per minute. To maintain synchrony in a chorus some members of the population in warm spots must slow down, whilst those at lower temperature must increase their singing rate. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, C
distance information	Insects are frequency sensitive, with many showing specialised hearing structures capable of differentially filtering certain frequency bands. One of the features of sound as it travels through the medium is that certain frrequencies are lost faster than others and this loss may give information on the distance of the caller. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, London

Title	Story
defence	Many ants produce sounds with high frequencies clsoe to 10 kHz that function in defence. The production of sound in ants is throught the movement of a plectrum on the posterior part of the petiole, where it engages a file on the first segment of the gaster. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, London
call for help	High frequencies are severely attenuated by soil, favouring the transmission of low frequency elements. Thus the non-sexual signals by ants, such as the leaf cutter ant Atta sexdens are low frequency calls, and although the function of these may normally be between nest mates, the signal also acts as an alarm to members of the colony when one of their number is buried in an earth fall. Bailey W.J. (1991) Acoustic Behaviour of Insects: an evolutionary perspective, Chapman and Hall, London
popcorn done	when you pop popcorn it starts off slowly popping then builds up in rate and loudness, then dies down again as most of the corn has popped. you can tell how many popcorns are going off all through the cooking process, and you can tell when its finished

Appendix 5-2: Lookup.prl

```
#!/local/bin/perl5 -w
#
##############################################################################
#COPYRIGHT (C) CSIRO DIT CSIS 1994
#
#SOURCE FILElookup.prl
#
#MODULE       Sonify
#
#SYSTEM       Thesis
#
#ENVIRONMENTSun SPARC and SUNOS4.1.2
#
#AUTHORStephen Barrass, CSIS
#
#HISTORY
#               : First written October 1995
##############################################################################


#-----------------------------------------------------------------------------
# help message
#
sub help {
print STDERR “Synopsis : $0 < database > examples\n”;
print STDERR “EarBenders Lookup - ranks records by similarity to a lookup record\n”;
print STDERR “newline is the record separator\n”;
print STDERR “+ is the within field AND operator\n”;
print STDERR “; is the within field OR operator\n”;

print STDERR “[-h]\t\t\t\tthis help message\n”;
}

#-----------------------------------------------------------------------------
# main
#
print STDERR “$0\n”;

#-----------------------------------------------------------------------------
# parse command line
#

while ($_ = $ARGV[0], /^-/) {
  shift;
if (/^-h/)
    {
            help;
           exit 1;
    }
  else
    {
            help;
           exit 1;
    }
}

$i = 0;
while (<@ARGV>) {
print “$_\n”;

  shift;
  $i++;
```

```
}

# score the entries
$i = 0;
$max = 0;
$maxcnt = 0;
$TASK_OFFSET=6;

while (<>) {
	chomp;
# split the line into fields
	@fields = split(/\|/);
	if ($i == 0) {
		@query = @fields;
	print STDERR "#QUERY\n#FIELD#=$#fields\n$_\n\n";
		$i++;
		next;
	}
	print STDERR "#story $i, fields=#=$#fields\n@fields\n";

# for each field
	$score = 0;
	for ($j=$TASK_OFFSET; $j < $#fields; $j++) {
		@subfields = split(/[;+]/, "$fields[$j]");
		for ($k=0; $k <= $#subfields; $k++) {
			print STDERR "\nQuery=$query[$j],Field=$subfields[$k]";
		  if ($query[$j] eq "???") {
					next;
				}
		  elsif ($query[$j] eq "") {
					next;
				}
		  elsif ($query[$j] eq "all") {
					print STDERR "=MATCH";
					$score++;
				}
		  elsif ($subfields[$k] eq "all") {
					print STDERR "=MATCH";
					$score++;
				}
		    elsif ((index $query[$j],$subfields[$k]) >= 0) {
					print STDERR "=MATCH";
					$score++;
		    }
		    elsif ((index $subfields[$k],$query[$j]) >= 0) {
					print STDERR "=MATCH";
					$score++;
		    }
		}
	}
print STDERR "SCORE=$score, MAX=$max\n";
	if ($score >= $max) {
		print "#story $i=$score, FIELDS=$#fields\n$_\n\n";
	 $maxcnt++;
		if (($i > 0) && ($maxcnt > 1)) {
			$max = $score;
		    $maxcnt = 0;
		}
	}
	$i++;
}

print "#QUERY -------\n";
print "@query\n";
print "#max=$max\n";
```

Appendix 5-3: Synthesis.prl

```
#!/local/bin/perl5
#
##############################################################################
#       COPYRIGHT (C) CSIRO DIT CSIS 1996
#
#       SOURCE FILE     synthesis.prl
#
#       MODULE          Sonify
#
#       SYSTEM          Thesis
#
#       ENVIRONMENT     Sun SPARC and SUNOS4.1.2
#
#       AUTHOR          Stephen Barrass, CSIS
#
#       HISTORY
#                       : First written December 1996
##############################################################################


#-----------------------------------------------------------------------------
# help message
#
sub help {
print STDERR “Synopsis : $0 < database > examples\n”;
print STDERR “EarBenders Design\n”;
print STDERR “synthesise design from Sound Analyses\n”;
print STDERR “[-i (3)]\t\t\tnumber of analyses to use\n”;

print STDERR “[-h]\t\t\t\tthis help message\n”;
}

#-----------------------------------------------------------------------------
# Sound Design
#

format DESIGN =
-----------------------------------------------
Sound Design
Nature          @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<|
@SoundNature
Level           @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<|
@SoundLevel
Streams         @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<|
@SoundStream
Range           @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<|
@SoundRange
Compound        @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
@SoundCompound
Occurence       @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
@SoundOccurence
Pattern         @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
@SoundPattern
Movement        @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
@SoundMovement
Relation        @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
@SoundRelation
Organisation    @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
@SoundOrganisation
.

sub SoundDesign {
```

```
# descriptions of the actual sounds to be used
@Sounds=();
# descriptions of principal variation
@Descriptors = ();

@SoundNatures = (‘everyday’, ‘synthetic’, ‘vocal’, ‘verbal’);
@SoundNature = ();

@SoundLevels = (‘local’, ‘global’);
@SoundLevel = ();

@SoundStreams = (‘single’,’few’,’many’);
@SoundStream = ();

@SoundOccurences = (‘isolated’);
@SoundOccurence = ();

@SoundPatterns = (‘discrete’);
@SoundPattern = ();

@SoundMovements = (‘stationary’,’jumping’,’smooth’,’blanket’);
@SoundMovement = ();

@SoundRelations = (‘category’, ‘continuum’, ‘order’, ‘metric’, ‘zero’);
@SoundRelation=();

@SoundOrganisations = (‘nominal’,’ordinal’,’interval’,’ratio’);
@SoundOrganisation = ();

@SoundCompounds = (‘integral’,’seperable’);
@SoundCompound = ();

#number of different sounds
@SoundRange = ();
}

#--------------------------------------------------------------------------------
# Majority
# return the majority trend in a list of fields
#
sub Majority {
(@fields) = @_;
local($i,$j);
$majority = 0;
$matchesMAX = 0;
# for each but the last field
for ($i=0; $i<$#fields; $i++) {
# count matches along the fields
      $matches = 0;
    for ($j=$i+1; $j <= $#fields; $j++) {
            if ($fields[$i] eq $fields[$j]) {
                  $matches++;
                }
            }
    if ($matches > $matchesMAX) {
            $matchesMAX = $matches;
             $majority = $i;
            }
   }
# if no majority then return the fields
if ($matchesMAX == 0) {
    return “no majority in (@fields)”;
    }
# return the majority trend
$m = ($matchesMAX+1);
$f = ($#fields+1);
```

```
$mTotal += $m;
$fTotal += $f;
return “$m\/$f ($fields[$majority]) in (@fields)”;
}

#------------------------------------------------------------------------------
# main
#
print STDERR “$0\n”;

#------------------------------------------------------------------------------
# parse command line
#
$iMAX = 3;                    # number of analyses to synthesise from
while ($_ = $ARGV[0], /^-/) {
  shift;
if (/^-h/)
    {
            help;
           exit 1;
    }
elsif (/^-i/)
    {
     if ($i=shift > 0) {
                $iMAX = $i;
              }
    }
  else
    {
            help;
           exit 1;
    }
}


$SOUND_OFFSET=15;             # offset of the sound analysis in EarBenders
SoundDesign;                  # initialise the Sound Design

$i = 0;

print “#--------- query\n<“;
$q = <STDIN>;
$q = <STDIN>;
$q = <STDIN>;
$q = <STDIN>;
$q = <STDIN>;
print “$q\n”;

while (<STDIN>) {
if (/^ *#/) { next; }# skip comment lines
if (/^ *\n/) { next; }# skip blank lines
if ($i >= $iMAX) { last; }
 chomp;
# split the line into fields
@fields = split(/\|/);
# get each field from each of the top entries
print STDOUT “#--------- case $i\n@fields\n”;
$j = $SOUND_OFFSET;
@SoundLevel = ($fields[$j++],@SoundLevel);
@SoundCompound = ($fields[$j++],@SoundCompound);
@SoundDescriptors = ($fields[$j++],@SoundDescriptors);
@SoundRelation = ($fields[$j++],@SoundRelation);
@SoundNature = ($fields[$j++],@SoundNature);
@SoundOccurence = ($fields[$j++],@SoundOccurence);
@SoundPattern = ($fields[$j++],@SoundPattern);
@SoundStream = ($fields[$j++],@SoundStream);
```

```
@SoundMovement = ($fields[$j++],@SoundMovement);

  $i++;
}

# design synthesis
# use majority trend for each field
# if no majority then leave it up to the designer
@SoundLevel = Majority(@SoundLevel);
@SoundCompound = Majority(@SoundCompound);
@SoundDescriptors = Majority(@SoundDescriptors);
@SoundRelation = Majority(@SoundRelation);
@SoundNature = Majority(@SoundNature);
@SoundOccurence = Majority(@SoundOccurence);
@SoundPattern = Majority(@SoundPattern);
@SoundStream = Majority(@SoundStream);
@SoundMovement = Majority(@SoundMovement);

open(DESIGN, ">-") || die "can't create";
write DESIGN;                              # output

#close INFILE;

$percentScore = ($mTotal*100)/$fTotal;
print "Total Match Score = $mTotal\/$fTotal = $percentScore\n";
```

Appendix 5-4: Casedesign.csh

```
#!/bin/csh -f
##############################################################################################
#COPYRIGHT (C) CSIRO DIT CSIS 1994
#
#SOURCE FILEbatch.csh
#
#MODULE       Sonify
#
#SYSTEM       Thesis
#
#ENVIRONMENTSun SPARC and SUNOS4.1.2
#
#AUTHORStephen Barrass, CSIS
#
#HISTORY
#               : First written September 1994
##############################################################################################
# Synopsis
#---------------------------------------------------------------------------
# command line
#
set files =
while ($#argv > 0)

      switch ($1)
            case -h:
                  goto synopsis
                  breaksw

          default:
               set files = ($files $1)
                    breaksw

      endsw
      shift
end

#---------------------------------------------------------------------------
# main
# assume query is the last item in the earbend.txt
# need to move it to the top position
tail -1r < earbend.txt > x
cat x earbend.txt | lookup.prl > ranked.txt
# best stories are at the end
tail -r < ranked.txt | synthesis.prl > design.txt
rm x
goto end
#---------------------------------------------------------------------------
# help message
#
synopsis:
echo “synopsis: $0”
echo “EarBenders case-based design synthesis”
echo “the earbenders case base is assumed to be in earbend.txt”
echo “the synthesised design goes to design.txt”
echo “lots of messages go to stderr”
end:
```

Appendix 6-1: GoldMaker.prl

```
#!/local/bin/perl5
#
################################################################################
#COPYRIGHT (C) CSIRO DIT CSIS 1994
#
#SOURCE FILEgoldbug.prl
#
#MODULE       Sonify
#
#SYSTEM       Thesis
#
#ENVIRONMENTSun SPARC and SUNOS4.1.2
#
#AUTHORStephen Barrass, CSIS
#
#HISTORY
#               : First written March 1997
################################################################################

# generate a mixture of dirt and gold
# there are a number of elements per handful and a number of handfuls that are thrown
$handful = 20;
$throws = 5;
$tempo = 600;
$goldlevel = 0.5;

#-------------------------------------------------------------------------
# help message
#
sub help {
print STDERR “Synopsis : $0 \n”;
print STDERR “generate handfuls of Bly’s dirt and gold data\n”;
print STDERR “[-h]\t\t\t\tthis help message\n”;
print STDERR “[-e <$handful>]\t\t\telements in a handful\n”;
print STDERR “[-g <$goldlevel>]\t\t\tratio of gold 0.0-1.0\n”;
print STDERR “[-l <$tempo>]\t\t\trate at which elements occur\n”;
print STDERR “[-t <$throws>]\t\t\tnumber of handfuls that are thrown\n”;
}

#-------------------------------------------------------------------------
# main
#
print STDERR “$0\n”;

#-------------------------------------------------------------------------
# parse command line
#

while ($_ = $ARGV[0], /^-/) {
  shift;
if (/^-h/)
     {
            help;
           exit 1;
     }
elsif (/^-e/)
     {
      $handful = shift;
     }
elsif (/^-l/)
     {
      $tempo = shift;
```

```
    }
elsif (/^-t/)
    {
     $throws = shift;
    }
elsif (/^-g/)
    {
     $goldlevel = shift;
    }
  else
    {
           help;
          exit 1;
    }
}

# normal deviate by the Polar method

sub norm {
my $s = 1;
while ($s >= 1) {
     $u1 = (rand) * 2-1;
     $u2 = (rand) * 2-1;
     $s = $u1*$u1 + $u2*$u2;
    }
$r = abs($u1*sqrt(-2*log($s)/$s));
return($r);
}

sub norm1 {
my $r = 4;
while ($r > 3.5) {
        $r = norm;
    }
return $r;
}

sub score {
      printf “i1 %.3f 1 %.3f %.3f %.3f %.3f %.3f %.3f “,$t+$x1,$x1,$x2,$x3,$x4,$x5,$x6;
#       printf “i1 + . %.3f %.3f %.3f %.3f %.3f %.3f “,$x1,$x2,$x3,$x4,$x5,$x6;

}

sub handful {
my($goldlevel) = @_;
my $count = 0;
my $goldcount = 0;
while ($count < $handful) {
# first get a random number for each dimension
      $x1 = norm1; $x2 =  norm1; $x3 = norm1;$x4 = norm1;$x5 = norm1;$x6 = norm1;
      $type = gold;
# default is gold
# set 2 = DIRT must fit these criteria
      if (
            ($x2*$x2+$x3*$x3+$x4*$x4+$x5*$x5+$x6*$x6 <= 2.25) ||
            ($x1*$x1+$x3*$x3+$x4*$x4+$x5*$x5+$x6*$x6 <= 2.25) ||
            ($x1*$x1+$x2*$x2+$x4*$x4+$x5*$x5+$x6*$x6 <= 2.25)
      ) {
            $q = 0; $r=0;
            if ($x1 < 1.5) {$q++};
            if ($x2 < 1.5) {$q++};
           if ($x3 < 1.5) {$q++};
            if ($x4 < 1.5) {$q++};
            if ($x5 < 1.5) {$q++};
            if ($x6 < 1.5) {$q++};
            if ($q < 5) {next};
```

```
            if ($x1 > 1.5) {$r++};
            if ($x2 > 1.5) {$r++};
            if ($x3 > 1.5) {$r++};
            if ($r > 1) {next};
            if ($x4 > 1.5 || $x5 > 1.5 || $x6 > 1.5) {next};
                $type = dirt;
      }

# if its gold and theres not enough in the mix then add it
#print "G=$type,$goldlevel,$goldcount,$handful\n";
     if ($type eq gold) {
               if ($goldcount++/$handful < $goldlevel) {
                              score();
            print "; gold\n";
                              $count++;
                        }
                     next;
              }
          score();
             print "; dirt\n";
         $count++;
     }
# end of handful
#print "s\n";
}

#
# MAIN
#
# print the header
print "f1 0 8193 10 1; sin\n";
print "f2 0 129 7 0 128 1; ramp 0 to 1\n";
print "t 0 $tempo; tempo\n";
printf "i1 %.3f 1 %.3f %.3f %.3f %.3f %.3f %.3f\n",0,0,0,0,0,0,0;

srand;
$t = 0;
for (; $t <= $throws; $t++) {
handful($goldlevel);
}
```

Appendix 7-1: PitchCircle.orc

```
;###############################################################################
;#     COPYRIGHT (C) CSIRO DIT CSIS 1993
;#
;#     SOURCE FILE     Shephard.orc
;#
;#     MODULE          Sonify
;#
;#     SYSTEM          PostGrad
;#
;#     ENVIRONMENT     Sun SPARC and SUNOS4.1.2
;#
;#     AUTHOR          Stephen Barrass, CSIS
;#
;#     HISTORY
;#                     : First written March 1994
;###############################################################################
;
;
; sparc10
;       sr = 44100
;       kr = 441
;       ksmps = 100

; sparc2
sr = 16000
kr = 1600
ksmps = 10

nchnls = 1

gifnyq = sr/2    ; nyquist freq
giattack  = 0.01  ; prevent onset clicks
gidecay   = 0.005  ; time buffer (in seconds) to prevent clicks
giampmax = 70     ; max amplitude in dB

; timbre  - consists of timbre angle and timbre radius
gatangle init 0
gatradius init 0
gatimbre init 0

;
;// instrument                 ////////////////////////////////////////////////
; pitch cycle - after Shephard
; p1p2       p3        p4        p5        p6        p7
; instrstartdur     comb    teeth     loud   ncycles
;                   oct.cl    oct.cl0.0..1.0    0-n

instr   1
print p1, p2, p3,p4,p5,      p6,        p7
idur=        p3
ifcomb=cpspch(p4)
ifteeth=cpspch(p5)
iamp  =ampdb(p6*giampmax)
incycles = p7
print idur, ifcomb, ifteeth, iamp, incycles

;-- loudness                 ------------------------------------------------
kamp    linen  iamp, giattack, idur, gidecay  ; de-click envelope
print iamp

; build the cycle-----------------------------------------------
; creat the src spectrum
```

```
;
; cycle up a ramp if continuous version is called by ncycles > 0
kcycleramp = 0
if (incycles < 1) kgoto discrete
kcycleramp line 0,idur,incycles    ; ramp
discrete:

; comb of 20 teeth----------------------------------------

; slight vibrato to bind the teeth together
ktrem oscil ifteeth*0.01, 15, 4

; controlling edge is sub-fundamental
kfh1 = ifcomb-ifteeth + ktrem + kcycleramp*ifteeth
axh1    oscil 1, kfh1, 3
; tooth
kfh2 = kfh1 + ifteeth
axh2    oscil 1, kfh2, 3
; tooth
kfh3 = kfh2 + ifteeth
axh3    oscil 1, kfh3, 3
; tooth
kfh4 = kfh3 + ifteeth
axh4    oscil 1, kfh4, 3
; tooth
kfh5 = kfh4 + ifteeth
axh5    oscil 1, kfh5, 3
; tooth
kfh6 = kfh5 + ifteeth
axh6    oscil 1, kfh6, 3
; tooth
kfh7 = kfh6 + ifteeth
axh7    oscil 1, kfh7, 3
; tooth
kfh8 = kfh7 + ifteeth
axh8    oscil 1, kfh8, 3
; tooth
kfh9 = kfh8 + ifteeth
axh9    oscil 1, kfh9, 3
; tooth
kfh10 = kfh9 + ifteeth
axh10    oscil 1, kfh10, 3
; tooth
kfh11 = kfh10 + ifteeth
axh11    oscil 1, kfh11, 3
; tooth
kfh12 = kfh11 + ifteeth
axh12    oscil 1, kfh12, 3
; tooth
kfh13 = kfh12 + ifteeth
axh13    oscil 1, kfh13, 3
; tooth
kfh14 = kfh13 + ifteeth
axh14    oscil 1, kfh14, 3
; tooth
kfh15 = kfh14 + ifteeth
axh15    oscil 1, kfh15, 3
; tooth
kfh16 = kfh15 + ifteeth
axh16    oscil 1, kfh16, 3
; tooth
kfh17 = kfh16 + ifteeth
axh17    oscil 1, kfh17, 3
; tooth
kfh18 = kfh17 + ifteeth
axh18    oscil 1, kfh18, 3
```

```
; tooth
kfh19 = kfh18 + ifteeth
axh19    oscil 1, kfh19, 3
; tooth
kfh20 = kfh19 + ifteeth
axh20    oscil 1, kfh20, 3

ax                                                                                                    =
axh1+axh2+axh3+axh4+axh5+axh6+axh7+axh8+axh9+axh10+axh11+axh12+axh13+axh14+axh15+axh16+axh17+a
xh18+axh19+axh20

; smooth the edges with a formant-------------------------------
dispfft ax, 1, 1024, 0, 0, 0

iformc= 2000
iformw= 1000
ay1reson  ax, iformc, iformw
ay2reson  ay1, iformc, iformw
ayreson  ay2, iformc, iformw

dispfft ay, 1, 1024, 0 ,0 ,0

;-- output                -------------------------------
gatangle    gain ay, kamp
dispfft gatangle, 1, 1024, 0, 0, 1
out     gatangle
endin


;
;################################################################################;#
;#      COPYRIGHT (C) CSIRO DIT CSIS 1993
;#
;#      SOURCE FILE     TWheel.sco
;#
;#      MODULE          Sonify
;#
;#      SYSTEM          PostGrad
;#
;#      ENVIRONMENT     Sun SPARC and SUNOS4.1.2
;#
;#      AUTHOR          Stephen Barrass, CSIS
;#
;#      HISTORY
;#                      : First written March 1994
;################################################################################
;
;-- tables    --------------------------------------------
;
; exponential curve
f2 0 128 5 0.001 128 1

; cosine wave
f3 0 8193 9 1 1 90

; sine wave
f4 0 8193 10 1

;-- rate      --------------------------------------------
;t 0 60

; streaming
; onsets 100 ms (range 50 -> 150 ms) = 60000/onset_in_ms
;t 0 1200; 5 = 50 ms onsets
;t 0 857; 4 = 70 ms onsets
;t 0 750; 80 ms onsets
```

```
;t 0 666; 3 = 90 ms onsets
;t 0 545; 2 = 110 ms onsets
;t 0 461; 1 = 130 ms onsets
;t 0 400; 0 = 150 ms onsets

;// instrument                ////////////////////////////////////////////
; pitch cycle - after Shephard
; p1p2      p3      p4       p5       p6       p6
; instrstartdur    comb    teeth    loud   ncycles
;                  oct.cl   oct.cl0.0..1.0   0-n

;-- galloping-----------------------------------------------
; adjacent
; 0 = 0,5,0
i1  0       1.0     8.00     8.00     1.0       0
i.  +         .     8.05       .        .        .
i.  +         .     8.00       .        .        .
i.  +         .       .        .        0        .
e

; complementary
; 0 = 0,6,0
i1  0       1.0     8.00     8.00     1.0       0
i.  +         .     8.06       .        .        .
i.  +         .     8.00       .        .        .
i.  +         .       .        .        0        .
e
;-- triplets   ---------------------------------------------
; split complementaries
; 0 = 0,8,4
i1  0       1.0     8.00     8.00     1.0       0
i.  +         .     8.08       .        .        .
i.  +         .     8.04       .        .        .
e
; 1 = 3,7,11
i1  0       1.0     8.03     8.00     1.0       0
i.  +         .     8.07       .        .        .
i.  +         .     8.11       .        .        .
e
; 2 = 6,10,2
i1  0       1.0     8.06     8.00     1.0       0
i.  +         .     8.10       .        .        .
i.  +         .     8.02       .        .        .
e
; 3 = 9,5,1
i1  0       1.0     8.09     8.00     1.0       0
i.  +         .     8.05       .        .        .
i.  +         .     8.01       .        .        .
e

;-- triplets   ---------------------------------------------
; adjacent
; 0 =  0,5,10
i1  0       1.0     8.00     8.00     1.0       0
i.  +         .     8.05       .        .        .
i.  +         .     8.10       .        .        .
e
; 3 = 9,2,7
i1  0       1.0     8.09     8.00     1.0       0
i.  +         .     8.02       .        .        .
i.  +         .     8.07       .        .        .
e
; 2 = 6,11,4
i1  0       1.0     8.06     8.00     1.0       0
i.  +         .     8.11       .        .        .
i.  +         .     8.04       .        .        .
```

```
e
; 1 = 3,8,1
i1  0       1.0     8.03    8.00     1.0       0
i.  +         .     8.08      .        .        .
i.  +         .     8.01      .        .        .
e

;-- circle of categories-----------------------------------------------
; cycle of fifths complementary
i1  0       1.0     8.00    8.00     1.0       0
i.  +         .     8.06      .        .        .
i.  +         .     8.05      .        .        .
i.  +         .     8.11      .        .        .
i.  +         .     8.10      .        .        .
i.  +         .     8.04      .        .        .
i.  +         .     8.03      .        .        .
i.  +         .     8.09      .        .        .
i.  +         .     8.08      .        .        .
i.  +         .     8.02      .        .        .
i.  +         .     8.01      .        .        .
i.  +         .     8.07      .        .        .

e

;-- circle of categories-----------------------------------------------
; cycle of fifths adjacent
i1  0       1.0     8.00    8.00     1.0       0
i.  +         .     8.05      .        .        .
i.  +         .     8.10      .        .        .
i.  +         .     8.03      .        .        .
i.  +         .     8.08      .        .        .
i.  +         .     8.01      .        .        .
i.  +         .     8.06      .        .        .
i.  +         .     8.11      .        .        .
i.  +         .     8.04      .        .        .
i.  +         .     8.09      .        .        .
i.  +         .     8.02      .        .        .
i.  +         .     8.07      .        .        .
e

;-- continuous ---------------------------------------------
i1  0      10.0     8.00     8.0      1.0       2
e

;-- discrete  ----------------------------------------------
; pitch order
i1  0       1.0     8.00    8.00     1.0       0
i.  +         .     8.01      .        .        .
i.  +         .     8.02      .        .        .
i.  +         .     8.03      .        .        .
i.  +         .     8.04      .        .        .
i.  +         .     8.05      .        .        .
i.  +         .     8.06      .        .        .
i.  +         .     8.07      .        .        .
i.  +         .     8.08      .        .        .
i.  +         .     8.09      .        .        .
i.  +         .     8.10      .        .        .
i.  +         .     8.11      .        .        .
e

Static Timbre, Formant and Timbre Circle
;
;##############################################################################;#
;#COPYRIGHT (C) CSIRO DIT CSIS 1993
;#
;#SOURCE FILEGreySun.orc
```

```
;#
;#MODULE     Sonify
;#
;#SYSTEM     PostGrad
;#
;#ENVIRONMENTSun SPARC and SUNOS4.1.2
;#
;#AUTHORStephen Barrass, CSIS
;#
;#HISTORY
;#              : First written March 1994
;###############################################################################
;
;
;

sr = 44100
kr = 4410
ksmps = 10
nchnls = 1

gifnyq = sr/2.1    ; nyquist freq + some spare (as per Moore)
giattack  = 0.01; time buffer (in seconds) to prevent clicks
gidecay   = 0.01; time buffer (in seconds) to prevent clicks
giampmax = 90; max amplitude in dB

; gamut stuff----------------------------------------------------
; notenum gamut ranges are in the score tables
giminp = 1; patch min
gimaxp = 8; patch max
giminf = 0; filter min
gimaxf = 127; filter max

gioutofgamut = 600; out-of-gamut sample
gioogamutp = 601; out-of-patch gamut sample
gioogamutn = 602; out-of-notenumber gamut sample
gioogamutf = 603; out-of-filter gamut sample
gisinfunc = 604; sine function
gibuzzfunc = 605; buzz function

gicountbase = 610; words for counting


;# Device              ####################################
; CSOUND SAMPLE PLAYER
; this instrument plays samples
; patch selects the sample bank
; notenum selects the closest sample from the bank
;
;// Instrument params//////////////////////////////////////
;   p1// in : instrument 1
;   p2// in : start time
;
instr1
print p1, p2, p3, p4, p5, p6, p7
idur    =p3   ;// in : duration
ipatch =p4+1;// in : patch number0..127
inotenum =p5;// in : notenum     0..127
ifilter =p6      ;// in : filter          0..127
ivelocity = p7;// in : velocity0..127

;-- velocity ------------------------------------------
iamp    = ampdb(ivelocity*giampmax/127); loudness max
;kamp linenr iamp, giattack, 0, .1 ;at noteoff, extend by 100 millisecs
kamplinen iamp, giattack, idur, gidecay
print iamp
```

```
;-- notenum----------------------------------------------
; calculate froot
; notenum 60 = middle C = csound oct 8.0
; range is from midi C0 to midi G10

ioct=3.0 + inotenum/12.0
ifroot=cpsoct(ioct)
print ioct, ifroot

;-- filter      ----------------------------------------------
if (ifilter > gimaxf) goto oogamutf
if (ifilter < giminf) goto oogamutf
; number of harmonics
; range is 0-32
inharm= ifilter/4.0
print inharm

; low pass filter with cutoff moving up the harmonics
ifcut =ifroot+ifroot*inharm

;check against nyquist - cut at nyquist if necessary
ifcut = (ifcut < gifnyq ? ifcut : gifnyq)
inharmnyq = gifnyq/ifroot
print gifnyq, ifcut, inharmnyq

;-- patch     ----------------------------------------------
print ipatch
if (ipatch == 500) goto refharmonic
if (ipatch == 501) goto refnoise
if (ipatch == 502) goto refacum
if (ipatch == 503) goto reftone
if (ipatch == gioogamutp) goto oogamutp
if (ipatch > gicountbase) goto count
; patch gamut range check for normal instruments
if (ipatch < giminp) goto oogamutp
if (ipatch > gimaxp) goto oogamutp

;---------
sample:
; get the sample bank index
;ibankindextableipatch, 1
; get the sample for this pitch
;isample  tableinotenum, ibankindex
isample   tableinotenum, ipatch
print isample, ipatch, inotenum
if (isample == gioogamutn) goto oogamutn
asrcloscilkamp, ifroot, isample
goto filter

;----------
reftone:
; sine tone
afinoscilkamp, ifroot, gisinfunc
goto output

;----------
refharmonic:
; Harmonic reference instrument
asrcoscilkamp, ifroot, gibuzzfunc
goto filter

;----------
refnoise:
; Noise reference instrument
asrcrandhkamp, gifnyq
```

```
goto filter

;----------
refacum:
; Acum reference instrument
; get the filter position
ifcut = p6 * gifnyq/127
; critical bandwidth ~ 0.2*f for f > 500Hz
;                 100Hz for f < 500Hz
;anoiserandifcut*0.2
asrcrandhkamp, gifnyq
; centre the critical-bandlimited noise around the cutoff
;afinosciliamp,  ifcut+anoise
iwidth = (ifcut < 500 ? 100 : ifcut*0.2)
afinresonasrc, ifcut, iwidth
goto output

;----------
oogamutp:
; out of patch gamut
afinlosciliamp, 200, gioogamutp
goto output

;----------
oogamutn:
; out of notenumber gamut
afinlosciliamp, 200, gioogamutn
goto output

;----------
oogamutf:
; out of filter gamut
afinlosciliamp, 200, gioogamutf
goto output

;----------
count:
; count instrument
asrclosciliamp, 200, ipatch
; don’t apply the filter to words
afin = asrc
goto output

; apply the brightness filter-----------------------------------------------
filter:
dispfftasrc, 0.1, 1024; DEBUG - look at signal

; use lowpass
afilttoneasrc, ifcut
afilttoneafilt, ifcut
afilttoneafilt, ifcut
afintoneafilt, ifcut
afinbalanceafilt, asrc
;
;-- output the sound---------------------------------------------
output:
dispfftafin, 0.1, 1024; DEBUG - look at signal
outafin
endin



;--------- flu
f1 0 128 -7 602 48 602 0 20 36 56 0 602 44 602
f20 0 0 1 “flu_48.aif” 0 4 0
f21 0 0 1 “flu_49.aif” 0 4 0
```

```
f22 0 0 1 “flu_50.aif” 0 4 0
f23 0 0 1 “flu_51.aif” 0 4 0
f24 0 0 1 “flu_52.aif” 0 4 0
f25 0 0 1 “flu_53.aif” 0 4 0
f26 0 0 1 “flu_54.aif” 0 4 0
f27 0 0 1 “flu_55.aif” 0 4 0
f28 0 0 1 “flu_56.aif” 0 4 0
f29 0 0 1 “flu_57.aif” 0 4 0
f30 0 0 1 “flu_58.aif” 0 4 0
f31 0 0 1 “flu_59.aif” 0 4 0
f32 0 0 1 “flu_60.aif” 0 4 0
f33 0 0 1 “flu_61.aif” 0 4 0
f34 0 0 1 “flu_62.aif” 0 4 0
f35 0 0 1 “flu_63.aif” 0 4 0
f36 0 0 1 “flu_64.aif” 0 4 0
f37 0 0 1 “flu_65.aif” 0 4 0
f38 0 0 1 “flu_66.aif” 0 4 0
f39 0 0 1 “flu_67.aif” 0 4 0
f40 0 0 1 “flu_68.aif” 0 4 0
f41 0 0 1 “flu_69.aif” 0 4 0
f42 0 0 1 “flu_70.aif” 0 4 0
f43 0 0 1 “flu_71.aif” 0 4 0
f44 0 0 1 “flu_72.aif” 0 4 0
f45 0 0 1 “flu_73.aif” 0 4 0
f46 0 0 1 “flu_74.aif” 0 4 0
f47 0 0 1 “flu_75.aif” 0 4 0
f48 0 0 1 “flu_76.aif” 0 4 0
f49 0 0 1 “flu_77.aif” 0 4 0
f50 0 0 1 “flu_78.aif” 0 4 0
f51 0 0 1 “flu_79.aif” 0 4 0
f52 0 0 1 “flu_80.aif” 0 4 0
f53 0 0 1 “flu_81.aif” 0 4 0
f54 0 0 1 “flu_82.aif” 0 4 0
f55 0 0 1 “flu_83.aif” 0 4 0
f56 0 0 1 “flu_84.aif” 0 4 0


;--------- cel
f2 0 128 -7 602 24 602 0 67 34 101 0 602 70 602
f67 0 0 1 “cel_24.aif” 0 4 0
f68 0 0 1 “cel_25.aif” 0 4 0
f69 0 0 1 “cel_26.aif” 0 4 0
f70 0 0 1 “cel_27.aif” 0 4 0
f71 0 0 1 “cel_28.aif” 0 4 0
f72 0 0 1 “cel_29.aif” 0 4 0
f73 0 0 1 “cel_30.aif” 0 4 0
f74 0 0 1 “cel_31.aif” 0 4 0
f75 0 0 1 “cel_32.aif” 0 4 0
f76 0 0 1 “cel_33.aif” 0 4 0
f77 0 0 1 “cel_34.aif” 0 4 0
f78 0 0 1 “cel_35.aif” 0 4 0
f79 0 0 1 “cel_36.aif” 0 4 0
f80 0 0 1 “cel_37.aif” 0 4 0
f81 0 0 1 “cel_38.aif” 0 4 0
f82 0 0 1 “cel_39.aif” 0 4 0
f83 0 0 1 “cel_40.aif” 0 4 0
f84 0 0 1 “cel_41.aif” 0 4 0
f85 0 0 1 “cel_42.aif” 0 4 0
f86 0 0 1 “cel_43.aif” 0 4 0
f87 0 0 1 “cel_44.aif” 0 4 0
f88 0 0 1 “cel_45.aif” 0 4 0
f89 0 0 1 “cel_46.aif” 0 4 0
f90 0 0 1 “cel_47.aif” 0 4 0
f91 0 0 1 “cel_48.aif” 0 4 0
f92 0 0 1 “cel_49.aif” 0 4 0
f93 0 0 1 “cel_50.aif” 0 4 0
```

```
f94 0 0 1 “cel_51.aif” 0 4 0
f95 0 0 1 “cel_52.aif” 0 4 0
f96 0 0 1 “cel_53.aif” 0 4 0
f97 0 0 1 “cel_54.aif” 0 4 0
f98 0 0 1 “cel_55.aif” 0 4 0
f99 0 0 1 “cel_56.aif” 0 4 0
f100 0 0 1 “cel_57.aif” 0 4 0
f101 0 0 1 “cel_58.aif” 0 4 0


;--------- cla
f3 0 128 -7 602 25 602 0 112 24 136 0 602 79 602
f112 0 0 1 “cla_25.aif” 0 4 0
f113 0 0 1 “cla_26.aif” 0 4 0
f114 0 0 1 “cla_27.aif” 0 4 0
f115 0 0 1 “cla_28.aif” 0 4 0
f116 0 0 1 “cla_29.aif” 0 4 0
f117 0 0 1 “cla_30.aif” 0 4 0
f118 0 0 1 “cla_31.aif” 0 4 0
f119 0 0 1 “cla_32.aif” 0 4 0
f120 0 0 1 “cla_33.aif” 0 4 0
f121 0 0 1 “cla_34.aif” 0 4 0
f122 0 0 1 “cla_35.aif” 0 4 0
f123 0 0 1 “cla_36.aif” 0 4 0
f124 0 0 1 “cla_37.aif” 0 4 0
f125 0 0 1 “cla_38.aif” 0 4 0
f126 0 0 1 “cla_39.aif” 0 4 0
f127 0 0 1 “cla_40.aif” 0 4 0
f128 0 0 1 “cla_41.aif” 0 4 0
f129 0 0 1 “cla_42.aif” 0 4 0
f130 0 0 1 “cla_43.aif” 0 4 0
f131 0 0 1 “cla_44.aif” 0 4 0
f132 0 0 1 “cla_45.aif” 0 4 0
f133 0 0 1 “cla_46.aif” 0 4 0
f134 0 0 1 “cla_47.aif” 0 4 0
f135 0 0 1 “cla_48.aif” 0 4 0
f136 0 0 1 “cla_49.aif” 0 4 0


;--------- tsx
f4 0 128 -7 602 36 602 0 147 13 160 0 602 79 602
f147 0 0 1 “tsx_36.aif” 0 4 0
f148 0 0 1 “tsx_37.aif” 0 4 0
f149 0 0 1 “tsx_38.aif” 0 4 0
f150 0 0 1 “tsx_39.aif” 0 4 0
f151 0 0 1 “tsx_40.aif” 0 4 0
f152 0 0 1 “tsx_41.aif” 0 4 0
f153 0 0 1 “tsx_42.aif” 0 4 0
f154 0 0 1 “tsx_43.aif” 0 4 0
f155 0 0 1 “tsx_44.aif” 0 4 0
f156 0 0 1 “tsx_45.aif” 0 4 0
f157 0 0 1 “tsx_46.aif” 0 4 0
f158 0 0 1 “tsx_47.aif” 0 4 0
f159 0 0 1 “tsx_48.aif” 0 4 0
f160 0 0 1 “tsx_49.aif” 0 4 0


;--------- ssx
f5 0 128 -7 602 61 602 0 171 14 185 0 602 53 602
f171 0 0 1 “ssx_61.aif” 0 4 0
f172 0 0 1 “ssx_62.aif” 0 4 0
f173 0 0 1 “ssx_63.aif” 0 4 0
f174 0 0 1 “ssx_64.aif” 0 4 0
f175 0 0 1 “ssx_65.aif” 0 4 0
f176 0 0 1 “ssx_66.aif” 0 4 0
f177 0 0 1 “ssx_67.aif” 0 4 0
```

f178 0 0 1 “ssx_68.aif” 0 4 0
f179 0 0 1 “ssx_69.aif” 0 4 0
f180 0 0 1 “ssx_70.aif” 0 4 0
f181 0 0 1 “ssx_71.aif” 0 4 0
f182 0 0 1 “ssx_72.aif” 0 4 0
f183 0 0 1 “ssx_73.aif” 0 4 0
f184 0 0 1 “ssx_74.aif” 0 4 0
f185 0 0 1 “ssx_75.aif” 0 4 0

;--------- ehn
f6 0 128 -7 602 40 602 0 196 30 226 0 602 58 602
f196 0 0 1 “ehn_40.aif” 0 4 0
f197 0 0 1 “ehn_41.aif” 0 4 0
f198 0 0 1 “ehn_42.aif” 0 4 0
f199 0 0 1 “ehn_43.aif” 0 4 0
f200 0 0 1 “ehn_44.aif” 0 4 0
f201 0 0 1 “ehn_45.aif” 0 4 0
f202 0 0 1 “ehn_46.aif” 0 4 0
f203 0 0 1 “ehn_47.aif” 0 4 0
f204 0 0 1 “ehn_48.aif” 0 4 0
f205 0 0 1 “ehn_49.aif” 0 4 0
f206 0 0 1 “ehn_50.aif” 0 4 0
f207 0 0 1 “ehn_51.aif” 0 4 0
f208 0 0 1 “ehn_52.aif” 0 4 0
f209 0 0 1 “ehn_53.aif” 0 4 0
f210 0 0 1 “ehn_54.aif” 0 4 0
f211 0 0 1 “ehn_55.aif” 0 4 0
f212 0 0 1 “ehn_56.aif” 0 4 0
f213 0 0 1 “ehn_57.aif” 0 4 0
f214 0 0 1 “ehn_58.aif” 0 4 0
f215 0 0 1 “ehn_59.aif” 0 4 0
f216 0 0 1 “ehn_60.aif” 0 4 0
f217 0 0 1 “ehn_61.aif” 0 4 0
f218 0 0 1 “ehn_62.aif” 0 4 0
f219 0 0 1 “ehn_63.aif” 0 4 0
f220 0 0 1 “ehn_64.aif” 0 4 0
f221 0 0 1 “ehn_65.aif” 0 4 0
f222 0 0 1 “ehn_66.aif” 0 4 0
f223 0 0 1 “ehn_67.aif” 0 4 0
f224 0 0 1 “ehn_68.aif” 0 4 0
f225 0 0 1 “ehn_69.aif” 0 4 0
f226 0 0 1 “ehn_70.aif” 0 4 0

;--------- bsn
f7 0 128 -7 602 22 602 0 237 31 268 0 602 75 602
f237 0 0 1 “bsn_22.aif” 0 4 0
f238 0 0 1 “bsn_23.aif” 0 4 0
f239 0 0 1 “bsn_24.aif” 0 4 0
f240 0 0 1 “bsn_25.aif” 0 4 0
f241 0 0 1 “bsn_26.aif” 0 4 0
f242 0 0 1 “bsn_27.aif” 0 4 0
f243 0 0 1 “bsn_28.aif” 0 4 0
f244 0 0 1 “bsn_29.aif” 0 4 0
f245 0 0 1 “bsn_30.aif” 0 4 0
f246 0 0 1 “bsn_31.aif” 0 4 0
f247 0 0 1 “bsn_32.aif” 0 4 0
f248 0 0 1 “bsn_33.aif” 0 4 0
f249 0 0 1 “bsn_34.aif” 0 4 0
f250 0 0 1 “bsn_35.aif” 0 4 0
f251 0 0 1 “bsn_36.aif” 0 4 0
f252 0 0 1 “bsn_37.aif” 0 4 0
f253 0 0 1 “bsn_38.aif” 0 4 0
f254 0 0 1 “bsn_39.aif” 0 4 0
f255 0 0 1 “bsn_40.aif” 0 4 0

```
f256 0 0 1 “bsn_41.aif” 0 4 0
f257 0 0 1 “bsn_42.aif” 0 4 0
f258 0 0 1 “bsn_43.aif” 0 4 0
f259 0 0 1 “bsn_44.aif” 0 4 0
f260 0 0 1 “bsn_45.aif” 0 4 0
f261 0 0 1 “bsn_46.aif” 0 4 0
f262 0 0 1 “bsn_47.aif” 0 4 0
f263 0 0 1 “bsn_48.aif” 0 4 0
f264 0 0 1 “bsn_49.aif” 0 4 0
f265 0 0 1 “bsn_50.aif” 0 4 0
f266 0 0 1 “bsn_51.aif” 0 4 0
f267 0 0 1 “bsn_52.aif” 0 4 0
f268 0 0 1 “bsn_53.aif” 0 4 0


;--------- tbn
f8 0 128 -7 602 28 602 0 279 32 311 0 602 68 602
f279 0 0 1 “tbn_28.aif” 0 4 0
f280 0 0 1 “tbn_29.aif” 0 4 0
f281 0 0 1 “tbn_30.aif” 0 4 0
f282 0 0 1 “tbn_31.aif” 0 4 0
f283 0 0 1 “tbn_32.aif” 0 4 0
f284 0 0 1 “tbn_33.aif” 0 4 0
f285 0 0 1 “tbn_34.aif” 0 4 0
f286 0 0 1 “tbn_35.aif” 0 4 0
f287 0 0 1 “tbn_36.aif” 0 4 0
f288 0 0 1 “tbn_37.aif” 0 4 0
f289 0 0 1 “tbn_38.aif” 0 4 0
f290 0 0 1 “tbn_39.aif” 0 4 0
f291 0 0 1 “tbn_40.aif” 0 4 0
f292 0 0 1 “tbn_41.aif” 0 4 0
f293 0 0 1 “tbn_42.aif” 0 4 0
f294 0 0 1 “tbn_43.aif” 0 4 0
f295 0 0 1 “tbn_44.aif” 0 4 0
f296 0 0 1 “tbn_45.aif” 0 4 0
f297 0 0 1 “tbn_46.aif” 0 4 0
f298 0 0 1 “tbn_47.aif” 0 4 0
f299 0 0 1 “tbn_48.aif” 0 4 0
f300 0 0 1 “tbn_49.aif” 0 4 0
f301 0 0 1 “tbn_50.aif” 0 4 0
f302 0 0 1 “tbn_51.aif” 0 4 0
f303 0 0 1 “tbn_52.aif” 0 4 0
f304 0 0 1 “tbn_53.aif” 0 4 0
f305 0 0 1 “tbn_54.aif” 0 4 0
f306 0 0 1 “tbn_55.aif” 0 4 0
f307 0 0 1 “tbn_56.aif” 0 4 0
f308 0 0 1 “tbn_57.aif” 0 4 0
f309 0 0 1 “tbn_58.aif” 0 4 0
f310 0 0 1 “tbn_59.aif” 0 4 0
f311 0 0 1 “tbn_60.aif” 0 4 0



;-- OUT-of-gamut------------------------------------------------------------
f600 0 0 1 “out.aif” 0 4 0; out of gamut
f601 0 0 1 “outp.aif” 0 4 0; out of patch gamut
f602 0 0 1 “outn.aif” 0 4 0; out of notenum gamut
f603 0 0 1 “outf.aif” 0 4 0; out of filter gamut
f604 0 2049 11 1; sinusoid
f605 0 2049 11 32; harmonics



;-- Count------------------------------------------------------
f610 0 0 1 “zero.aif” 0 4 0
f611 0 0 1 “one.aif” 0 4 0
```

```
f612 0 0 1 “two.aif” 0 4 0
f613 0 0 1 “three.aif” 0 4 0
f614 0 0 1 “four.aif” 0 4 0
f615 0 0 1 “five.aif” 0 4 0
f616 0 0 1 “six.aif” 0 4 0
f617 0 0 1 “seven.aif” 0 4 0
f618 0 0 1 “eight.aif” 0 4 0

;-- rate      ---------------------------------------------------------------
;t 0 60        ; 1000 ms = 1s onsets

; streaming
; typical range is 50 to 150 ms repeat (from Bregman)
; t (beats/minute) =  60000 (ms/minute)/(repeat time) ms
;t 0 1200; 50 ms onsets
;t 0 1000; 60 ms onsets
;t 0 857; 70 ms onsets
;t 0 750; 80 ms onsets
;t 0 666; 90 ms onsets
t 0 600; 100 ms onsets
;t 0 545; 110 ms onsets
;t 0 461; 130 ms onsets
;t 0 400; 150 ms onsets
;t 0 333; 180 ms onsets
;t 0 300; 200 ms onsets
;t 0 240; 250 ms onsets
;t 0 218; 275 ms onsets
;t 0 200; 300 ms onsets
;t 0 150; 400 ms onsets
;t 0 120; 500 ms onsets

;------- add score events below this line ------------------------------------
; parameters
;device startdurpatchnotenumfiltervelocity
;1beats   beats   0..127  0..127  0..127  0..127

; max duration = 1s
; duration range
;0.2 - 2s = duration
; 0.2 - 0.05 s = streaming
; 0.01-0.05 = clicks

; vertical pitch  gallop ----------------------------------------------

; far
i1  0       1       6       32      127     127
i.  +       .       .       52      .       .
i.  +       .       .       32      .       .
i.  +       .       .        .      .       0
e

; near
i1  0       1       6       47      127     127
i.  +       .       .       52      .       .
i.  +       .       .       47      .       .
i.  +       .       .        .      .       0
e

; mid
i1  0       1       6       42      127     127
i.  +       .       .       52      .       .
i.  +       .       .       42      .       .
i.  +       .       .        .      .       0
e

; vertical pitch triplet ----------------------------------------------
```

```
; 5%
i1  0       1       3       38      127     127
i.  +       .       .       >       .       .
i.  +       .       .       48      .       .
i.  +       .       .       .       .       0
e

;-- brightness zeros+outlier-----------------------------------------------
i1  0       1.0     0       48      0       127
i.  +       .       1       .       .       .
i.  +       .       2       .       .       .
i.  +       .       3       .       .       .
i.  +       .       5       .       .       .
i.  +       .       2       .       127     .
i.  +       .       6       .       0       .
i.  +       .       7       .       .       .
e

;-- brightness spiral------------------------------------------------
i1  0       1.0     0       48      127     127
i.  +       .       1       .       .       .
i.  +       .       2       .       .       .
i.  +       .       3       .       0       .
i.  +       .       5       .       .       .
i.  +       .       6       .       .       .
i.  +       .       7       .       127     .
e

;-- brightness spiral triplet ----------------------------------------------
i1  0       1.0     1       48      0       127
i.  +       .       2       .       >       .
i.  +       .       3       .       80      .
e

;-- brightness bilateral ---------------------------------------------
i1  0       1.0     2       48      0       127
i.  +       .       2       .       127     .
i.  +       .       6       .       0       .
i.  +       .       6       .       127     .
e

; duration triplet-----------------------------------------------

; 25%
i1  0       0.25    3       48      127     127
i.  1       0.5     .       .       .       .
i.  2       0.75    .       .       .       .
i. 2.9      0.1     .       .       .       0
e

; 5%
i1  0       0.5     3       48      127     127
i.  1       0.55    .       .       .       .
i.  2       0.6     .       .       .       .
i. 2.9      0.1     .       .       .       0
e

; duration sequence-----------------------------------------------

; 8 steps
i1  0       0.1     3       48      127     0
i.  1       0.1     .       .       .       127
i.  2       0.2     .       .       .       .
i.  3       0.3     .       .       .       .
i.  4       0.4     .       .       .       .
```

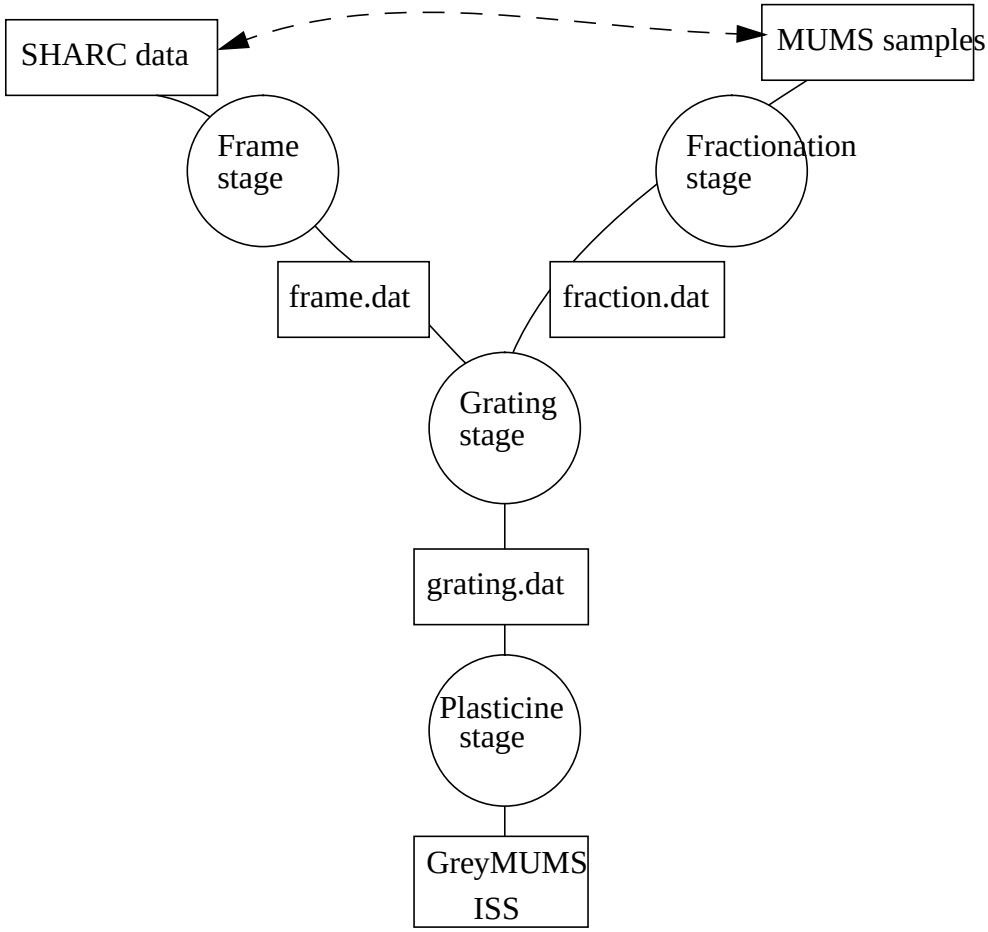
```
i.  5     0.5     .     .     .     .
i.  6     0.6     .     .     .     .
i.  7     0.7     .     .     .     .
i. 7.9    0.1     .     .     .     0
e

; duration gallop----------------------------------------------

; close
i1  0     0.5     3     48    127   127
i.  1     0.6     .     .     .     .
i.  2     0.5     .     .     .     .
i.  3      .      .     .     .     0
e

; far
i1  0     0.1     3     48    127   127
i.  1     1.0     .     .     .     .
i.  2     0.1     .     .     .     .
i.  3      .      .     .     .     0
e

; mid
i1  0     0.5     3     48    127   127
i.  1     1.0     .     .     .     .
i.  2     0.5     .     .     .     .
i.  3     0.5     .     .     .     0
e


;-- loudness gallop-----------------------------------------------
; far
i1  0     1.0     3     48    127   50
i.  +      .      .     .     .     100
i.  +      .      .     .     .     50
i.  +      .      .     .     .     0
e

; mid
i1  0     1.0     3     48    127   60
i.  +      .      .     .     .     80
i.  +      .      .     .     .     60
i.  +      .      .     .     .     0
e
; close
i1  0     1.0     3     48    127   70
i.  +      .      .     .     .     76
i.  +      .      .     .     .     70
i.  +      .      .     .     .     0
e


; loudness triplet 5%-------------------------------------------
i1  0     1.0     4     65    127   80
i.  +      .      .     .     .     86
i.  +      .      .     .     .     92
e

;-- brightness triplet--------------------------------------------

; 25%,50%,75%
i1  0     1.0     7     48    32    127
i.  +      .      .     .     64    .
i.  +      .      .     .     96    .
e
```

```
;-- brightness gallop----------------------------------------------

; mid
i1  0       1.0       4       65       70       127
i.  +         .         .         .       10         .
i.  +         .         .         .       70         .
i.  +         .         .         .         .         0
e

; close
i1  0       1.0       4       65       40       127
i.  +         .         .         .       45         .
i.  +         .         .         .       40         .
i.  +         .         .         .         .         0
e

; far
i1  0       1.0       4       65       10       127
i.  +         .         .         .      127         .
i.  +         .         .         .       10         .
i.  +         .         .         .         .         0
e


;-- brightness sequence-------------------------------------------
; filter = 0..127
i1  0       1.0       3       48        0       127
i.  +         .         .         .         >         .
i.  +         .         .         .         >         .
i.  +         .         .         .         >         .
i.  +         .         .         .         >         .
i.  +         .         .         .         >         .
i.  +         .         .         .         >         .
i.  +         .         .         .         >         .
i.  +         .         .         .       127         .
e

;-- timbre   -----------------------------------------------
; galloping
; adjacent
; 0 = 2,3,2_
i1  0       1.0       2       48      100       127
i.  1         .         3         .         .         .
i.  2         .         2         .         .         .
i.  3         .         .         .         .       0.0
e
; complementary
; 0 = 1,5,1_
i1  0       1.0       1       48      100       127
i.  1         .         5         .         .         .
i.  2         .         1         .         .         .
i.  3         .         .         .         .       0.0
e


;-- triplets   ---------------------------------------------
; complementary
; 1 = 2,5,7
i1  0       1.0       2       48      100       127
i.  +         .         5         .         .         .
i.  +         .         7         .         .         .
e
; 2 = 4,7,1
;i1 0       1.0       4       48      100       127
;i.  +         .         7         .         .         .
```

```
;i. +       .       1       .       .       .
;e
; 3 = 6,1,3
i1  0       1.0     6       48      100     127
i.  +       .       1       .       .       .
i.  +       .       3       .       .       .
e
; 0 = 0,3,5
i1  0       1.0     0       48      100     127
i.  +       .       3       .       .       .
i.  +       .       5       .       .       .
e


;-- triplets  ----------------------------------------------
; adjacent
; 3 = 5,6,7
i1  0       1.0     5       48      100     127
i.  +       .       6       .       .       .
i.  +       .       7       .       .       .
e
; 0 = 7,0,1
i1  0       1.0     7       48      100     127
i.  +       .       0       .       .       .
i.  +       .       1       .       .       .
e
; 1 = 1,2,3
i1  0       1.0     1       48      100     127
i.  +       .       2       .       .       .
i.  +       .       3       .       .       .
e
; 2 = 3,4,5
;i1 0       1.0     3       48      100     127
;i.  +      .       4       .       .       .
;i.  +      .       5       .       .       .
;e

;-- circle    ----------------------------------------------
i1  0       1.0     0       48      100     127
i.  +       .       1       .       .       .
i.  +       .       2       .       .       .
i.  +       .       3       .       .       .
;i. +       .       4       .       .       .
i.  +       .       5       .       .       .
i.  +       .       6       .       .       .
i.  +       .       7       .       .       .
e
```

Appendix 8-1: Building an ISS

This appendix describes the process and tools used to build the GreyMUMS perceptually linearised information sound space. The process has the following stages, which are described in the following subsections.



Appendix 8-2: SHARC and MUMS

Written and maintained by Gregory J. Sandell, Sussex University
Release 0.90("beta"), November 1994

Contents of this README file

WHAT IS SHARC?

SHARC is a database of musical timbre information by Gregory Sandell. It stands for "Sandell Harmonic Archive." People for whom this dataset may be useful are Acousticians, Psychoacousticians, researchers in Music Perception and Cognition, researchers in Digital Signal Processing, Music Theorists, and Musicologists.
Over 1300 different notes have been analysed. Complete chromatic runs from the standard playing range of essentially all the non-percussive instruments of the modern orchestra have been included; for example, individual analyses of 32 different oboe notes (the chromatic scale from the pitches a#3 to f6) are available.
For each note, a short portion corresponding to the sustain or "steady state" portion of the tone was selected and analysed with a Fourier analysis. Each analysis consists of a list of amplitudes and phases for all the note's harmonics in the range 0-10,000 Hz.
The source of the musical notes were the orchestral tones from the McGill University Master Samples (MUMS) Compact Discs. These are digital recordings of live musical performers.

RELEASE VERSION, AVAILABILITY, AUTHOR INFORMATION

Version 0.90 ("beta", November 1994) is the first-ever release. The reason I am calling it a "beta" is that there are a number of changes which may occur in the short term: the name of the archive, the internet location or URL, availability of the database from a US ftp site, the availability of online graphics to view the database, and adding more instruments. Any of these changes will be announced in all the relevant internet bulletin boards and distributed email lists. Your comments are most welcome, especially now, as they will have the most impact on the nature of the database. Send email to sandell@epunix.sussex.ac.uk.
Here are instructions for installing the archive on a UNIX platform.

- Go to the directory where you want to install SHARC. You will need about 10 free megabytes to perform the installation. Once finished, the archive will occupy 4.5 megabytes space.
- ftp to ftp.ep.susx.ac.uk

- Enter user name "anonymous" and give your full email address as a password
- Type "binary" to set the transfer to binary mode
- Type "get sharc.tar.Z"
- When the file is finished transferring (it is about 0.8 megabytes large), leave ftp by typing "quit"
- Type "uncompress sharc.tar.Z". The result of this is that sharc.tar.Z will be replaced by sharc.tar
- Type "tar xf sharc.tar". This will put the archive in a directory called "sharc".
- Dispose of the sharc.tar file (type "rm sharc.tar") to reclaim about 4.5 megabytes space.

Gregory J. Sandell is a research fellow in the Hearing Research Group at the University of Sussex Experimental Psychology department in the U.K. Starting April 1995 I will be at Parmly Hearing Institute at Loyola University in Chicago, IL (USA), but my Sussex email address (sandell@epunix.sussex.ac.uk) will remain active for a while.

PERMISSIONS

The data contained in this directory are available for free with the following restrictions:

- It may be used and shared on a non-profit basis only. You may not sell it for money or use it for trade; you may not bundle it with a commercial product or use it to attract customers to buy a product.
- It must be identified as "SHARC" with Gregory Sandell identified as its author. The author appreciates proper acknowledgement when this data is referred to in published papers.

CONTENTS

The database consists of 39 directories, each corresponding to a particular instrument. Each directory consists of separate files for each note analysed for that instrument. The instruments are:

Bach_trumpet	bass_clarinet	altoflute_vibrato
Bb_clarinet	bass_trombone	piccolo
CB	bassflute_vibrato	trombone
CB_martele	bassoon	trombone_muted
CB_muted	tuba	violin_vibrato
CB_pizz	cello_martele	viola_martele
C_trumpet	cello_muted_vibrato	viola_muted_vibrato
C_trumpet_muted	cello_pizzicato	viola_pizzicato
Eb_clarinet	cello_vibrato	viola_vibrato
English_horn	contrabass_clarinet	violin_martele

French_horn	contrabassoon	violin_muted_vibrato
French_horn_muted	flute_vibrato	violin_pizzicato
alto_trombone	oboe	violinensemb

INFORMATION ABOUT THE SOURCE SOUNDS
The McGill University Master Samples (MUMS) is a library of compact discs made and sold by Frank Opolko and Joel Wapnick of McGill University. To obtain the CDs or obtain information about them, write to:

McGill University
Faculty of Music
555 Sherbrooke Street West
Montrel, Quebec
Canada H3A 1E3
phone: (514) 398-4548
email: CXJW@MUSICA.MCGILL.CA (Joel Wapnick)

The naming of instruments, the organisation of the directories, and the information in each directory's CONTENTS files reflects the contents of the MUMS CDs as precisely as possible. I would like to gratefully acknowledge the permission of the makers of MUMS use their product in this manner. Other than customer, I have no financial relationship to MUMS or McGill University.

HOW IS THIS DATA USEFUL?
One of the most mportant aspects of a musical instrument sound that determines its timbre are the spectrum of its steady state portion. Other critical features are the rapid spectral changes at attack and decay time, and slowly varying changes in spectrum during the steady state. The fact that timbre depends so critically on the latter three aspects makes the study of timbre a challenge, because of the increased complexity of including the temporal dimension. To create a database of all the instruments of the orchestra with complete spectrotemporal descriptions of individual notes is not currently feasible, not as an archive to be shared through current network means, at least; several gigabytes, or a few CD-ROM discs would be required. A library of steady state spectra, however, is feasable. Admittedly, the study of steady state spectra is nothing new. However, prior to the use of computer analysis of sound, spectral analyses were expensive and hard-won operations; the unfortunate practice of analysing one note and drawing conclusions about the entire instrument's timbre was sometimes seen in manuals of acoustics from that time. The balance struck in this collection between economy of representation (steady state spectra) and completeness (complete chromatic scales for each instrument) offers researchers new opportunity for timbral discovery. Specifically, it puts the study of the "macro timbre" of an instrument, i.e. its spectral content of its entire pitch range, within the grasp of the researcher.
Some of the ways in which this data might be used are:

- Calculating the spectral centroids of the notes, and plotting this as estimated "brightness" over the range of the instrument.
- Similarly, a algorithm for estimating "roughness" or acoustic dissonance may be applied to compare one note or instrument to another.
- Because the information on the relative amplitude of notes is available (in each instru-

ment's CONTENTS file), one can form hypotheses about the dynamic nature of an instrument's performing range.

- Spectra may be combined to simulate harmonies; dissonances of combined spectra could be calculated, and the database searched to find, for example, the most consonant example of a semitone interval, or the most dissonant perfect fifth.
- These analyses might be the basis for a particular orchestration used by a composer; or, they could be used by Music Theorists and Musicologists for analysis of orchestration.
- Acousticians may with to test propositions such as the claim that the oboe and English horn possesses "formants."

Perhaps most important of all is that users of this database understand how to interpret this data correctly. For users with backgrounds in acoustics and Digital Signal Processing, the section HOW THE ANALYSES WERE DONE may be sufficient; for others, the implications of this approach are more explicitly spelled out below. For each note analysed, the user should keep in mind that:

- The portion of the tone which has been analysed has been very carefully selected for "representativeness", but nevertheless the analysis represents the spectrum of the note at single, brief moment in time.
- While this data, by itself, cannot be used to synthesise realistic sounding musical instruments, it does represent a real moment in time from the instrument, and a resynthesis of the waveform from this data will faithfully reproduce that moment. Although such moments are usually rather flat and "electronic" sounding, there are many occasions in which the steady state alone produces a quite recognisable musical instrument sound. Wind instruments are a frequent example. In fact, with a few added features (such as adding a characteristic attack-sustain-decay-release envelope to the sound), I have even been able to generate a tone that can "fool the listener." However, as a general rule, this will be very poor strategy for musical instrument tone synthesis.
- Any given note, say, a flute c4, is by no means the "final word" on the spectrum of a flute c4. There are an infinite number of ways a flute c4 can be played, and the instance of it on the MUMS CD represents only one manner of playing from a particular player, instrument, and recording conditions. There are some who take a rather pessimistic stance and believe that, because of the infinite number of possibile performances of a note, there is no use to having information on a single note. I believe that so long as one interprets the data appropriately, data on one note is far better than data on no notes at all.

HOW THE ANALYSES WERE DONE

For each analysed note, the objective was to provide Fourier spectra for a portion of the tone that was maximally "representative" of the steady portion of the tone.
Tremendous care was taken in finding a "representative" portion of each note. The procedure was as follows:

- The samples for the note were taken from the CD and put in a computer sound file. Only one of the CD's stereo channels were saved. Leading and trailing silence was removed. The sampling rate of the file (44100) was converted to 22050.

- The sound file was analysed with a Phase Vocoder.
- The longest continuous stretch of time in which the note was at 75% or more of its maximum amplitude was identified from the PV information. This located the steady portion of the tone.
- An average spectrum was calculated from all the PV frames identified in step 3. Then least squares was used to find the actual PV frame most closely resembling this average spectrum. The point in time corresponding to this PV frame was designated the "representative point".

Analysis then proceded as follows:

- A chunk of samples corresponding to five periods of the nominal fundamental (ie. according to the equal-tempered frequency of the note) were taken from the sound file, from a point in time symmetrically about the "representative point".
- Autocorrelation was used to estimate the actual fundamental frequency of the sample chunk. Once determined, the chunk was trimmed to four periods of this fundamental. The starting point of the four periods was selected to be at a zero crossing.
- The length of the sample chunk was changed to the next largest power of two by the method of bandlimited interpolation. This step was taken to make it possible to use an FFT.
- The sample chunk was Hamming-windowed.
- The samples were analysed with an FFT, and the real and imaginary values converted to power spectra in decibels. In order to save only partials at harmonic multiples of the fundamental, only every fourth bin was saved (because the sample chunk contained not just one period, but four). All bins greater than 10 kHz were discarded.

ORGANISATION OF THE DIRECTORIES

Each instrument has its own directory; within each directory is a separate file for each of the notes that were available for analysis. The organisation is such that in order to interpret each individual note file completely, you need to reference a file titled "CONTENTS" within the same directory. The individual note files have N rows, where N is the number of harmonics for that note (all possible harmonics below 10 kHz are included). There are two columns, one for the amplitudes (given in decibels relative to the amplitude of the loudest harmonic for that note) and one for the phases (-PI to +PI). The frequencies of the harmonics are integer multiples of the note's fundamental. The actual frequencies of the harmonics are simply the row number multiplied by the fundamental frequency for that note (as found in the "CONTENTS" file).

The CONTENTS file contains a line containing information about each of the notes in the directory. There are ten columns in each line:

- Column 1: The pitch (which identifies what file in the directory this line refers to). The pitch naming system is the Acoustical Society of America standard, i.e. c4 = middle C.
- Column 2: The note number of this pitch (where c4 = 48)
- Column 3: Number of harmonics in the file (hence the number of lines as well)
- Column 4: The maximum absolute value of the sample segment used in the analysis of this tone (i.e. the raw samples as read off of the CD). This is useful for compar-

ing the levels between notes. The possible range of samples on a CD are, of course, -32767 to 32768.

- Column 5: The nominal fundamental frequency for the pitch, according to equal-tempered tuning.
- Column 6: The actual fundamental frequency, as measured from the samples for this tone.
- Column 7: Volume number of the McGill University Master Samples (MUMS) CDs from which this note comes
- * Column 8: MUMS track number
- * Column 9: MUMS index number
- * Column 10: Total duration (in seconds) of the performed note on the CD, from onset to end of decay.
- * Column 11: The point in time (in seconds), relative to the onset of the note, from which the analysis was taken.
- * Column 12: the Spectral centroid in hertz

A BRIEF HISTORY OF THIS PROJECT

I began this project while a PhD student at Northwestern University (USA) in 1990. All the orchestral tones from the MUMS CDs were analysed. I reported on the project at the 1991 International Computer Music Conference in Montreal (Sandell, 1991, "A library of orchestral instrument spectra," Proceedings of the 1991 International Computer Music Conference, 98-101), but this data was never made publically available.

After a few years of looking at the data and thinking of ways in which the project could have been done better, and after finding a way in which to automate the task using a CD-ROM drive, I re-did the entire project from scratch. This was done in 1993-94 at Sussex University.

I would like to acknowledge the support of the graduate school of Northwestern University for a Dissertation Year Grant that helped fund the original project.

INFORMATION FOR WORLD WIDE WEB USERS

The top level URL for SHARC is ftp.ep.susx.ac.uk/pub/sandell/. Once there, you can read this documentation file by clicking on README.html, or browse a small example (just two of the 39 instruments) of what the database actually looks like when installed by clicking on examples. Note that the README.html is always the most up to date version of the documentation that is available.

My homepage is http://ep56c.ep.susx.ac.uk/Greg.Sandell.html. Be warned that all of these URLs may be changing in the near future.

BUGS, INACCURACIES, WISHLIST

- The autocorrelation method was not always successful in determining the fundamental frequency. The pizzicato and marcato string notes in particular seemed to pose the biggest problem. The most serious errors of this sort have been fixed, but a few modest errors (fundamentals off by several Hertz) remain to be corrected. In several cases the problem was due to a strong resonance or vibration from an open string that produced a second tone that competed with the nominal tone, or noisiness in the note such as a prominent bow scraping sound.

- This previous point highlights the fact that the analysis approach used in SHARC assumes all the instruments to produce harmonic spectra. Obviously this is not true in the case of the strings, in which the vibration of other strings may be an essential part of the timbre of the note in question. Inharmonicities in instruments can also occur when a strong native resonance for the particular instrument is active. The choice to save only harmonic information was made because

  + The vast majority of instruments do not evince strong inharmonic partials
  + A Fourier analysis will show energy at frequency locations that are not harmonic partials for any instrument; to decide whether to accept or reject a given inharmonic partial requires an ad hoc decision. Because of the large size of this database, this sort of attention to individual notes is not feasable.
  + I wanted to avoid having multiple formats for the files (one for harmonic notes, another for inharmonic notes)

- Some instruments have notes "missing" in the series (for example, tuba e2). This is because these notes are missing on the MUMS CDs as well.

- I have no idea how the MUMS engineers set the level from one day to the next over the course of the recordings. I have a hunch that, within one chromatic scale for each instrument, the same level and mike placement was used. Otherwise all bets are off: different instruments may have had different levels and mike placements, so it would be wise to practice caution in comparing the levels across instruments (which you can do by consulting the CONTENTS files for each instrument).

- A few instruments that I am analysing have not made their way into SHARC yet, but will soon: these include piano, celesta, harp, and some early instruments (all from MUMS).

- I plan to make graphic plots of the data available on the World Wide Web. Part of this will come very soon. Later, when advances in the Web make it possible, I hope to have a "graphical timbre server" where users are able to make individual requests for certain types of plots.

Gregory J. Sandell (sandell@epunix.sussex.ac.uk)

Appendix 8-3: Frame stage

The frame data is calculated from the SHARC database of spectroid measurements. The calculation is done by a csh script which expects a subdirectory for each instrument, with files for each note that have the naming convention instrument_notenum.spect, as is standard in the SHARC distribution. I have written a system of scripts to calculate brightness in Acums from the SHARC spectroids, and to plot the results. The process is automated by the script GreyAcumMap.csh which produces the frame as a text data file as follows

GreyAcumMap.csh > frame.dat

The programs in the system are listed below.

GreyAcumMap.csh	get the acumoid for Sample Pitches for the Grey timbres from SHARC files
GreyAcum.csh	get Acum data from Grey SHARC files
AcumPlot.csh	go through SHARC directories creating Acum plots
AcumData.csh	go through SHARC directories calculating Acumoids
Acumoid.cc	calculate Sharpness in Acums from SHARC timbre database files
midi2hz.cc	convert midi notenumber to Hz
name2midi.cc	convert note name (e.g. c4) to midi notenumber

```
#! /bin/csh -f
#/*
#*****************************************************************************
#*     COPYRIGHT (C) CSIRO DIT CSIS 1991
#*
#*     SOURCE FILE     GreyAcumMap
#*
#*     MODULE          Sonify
#*
#*     SYSTEM          research
#*
#*     ENVIRONMENT     Sun SPARC and SUNOS4.1
#*
#*     AUTHOR          Stephen Barrass, CSIS
#*
#*     HISTORY
#*     19/4/95: first written   s.b.
#*
#*****************************************************************************
#*/
# filenames are in form instr_nnn.acum

if ($1 == “-h”) then
goto synopsis
endif

set lo=$1
if ($lo == ““) then
set lo=0
endif

set hi=$2
if ($hi == ““) then
set hi=127
endif

foreach dir ( flute_vibrato cello_muted_vibrato bass_clarinet English_horn bassoon trombone_muted )
set notenum=$lo
while ($notenum < $hi)
     set notename=`midi2name $notenum`
     GetAcum.csh $dir $notename
      @ notenum++
   end
end

goto end

synopsis:
echo “$0 [lo <0>] [hi <127>]”
echo “lo = low notenumber”
echo “hi = high notenumber”
echo “gets the acumoid for Sample Pitches in range lo-hi for the Grey timbres from SHARC files”

end:

#! /bin/csh -f
#/*
#*****************************************************************************
#*     COPYRIGHT (C) CSIRO DIT CSIS 1991
#*
#*     SOURCE FILE     GreyAcum
#*
#*     MODULE          Sonify
#*
```

```
#*      SYSTEM          research
#*
#*      ENVIRONMENT     Sun SPARC and SUNOS4.1
#*
#*      AUTHOR          Stephen Barrass, CSIS
#*
#*      HISTORY
#*      19/4/95: first written   s.b.
#*
#*******************************************************************************
#*/
# filenames are in form instr_nnn.acum

if ($1 == "-h") then
goto synopsis
endif

foreach dir ( flute_vibrato cello_muted_vibrato bass_clarinet English_horn bassoon trombone_muted )
 cd $dir
foreach notename ($argv)
     set midi=`name2midi $notename`
     set hz=`midi2hz $midi`
     set acumoid=`Acumoid -b $hz < *_$notename.spect`
     if ($acumoid != "") then
               echo $dir $midi $acumoid
             else
               echo $dir $midi NULL
             endif
   end
  cd ..
end

goto end

synopsis:
echo "$0 notename (e.g. c4 c5 c6)"
echo "gets the acumoid for the Grey timbres from SHARC files"

end:


#! /bin/csh -f
#/*
#*******************************************************************************
#*      COPYRIGHT (C) CSIRO DIT CSIS 1991
#*
#*      SOURCE FILE     AcumPlot
#*
#*      MODULE          Sonify
#*
#*      SYSTEM          research
#*
#*      ENVIRONMENT     Sun SPARC and SUNOS4.1
#*
#*      AUTHOR          Stephen Barrass, CSIS
#*
#*      HISTORY
#*      19/4/95: first written   s.b.
#*
#*******************************************************************************
#*/
# filenames are in form instr_nnn.acum

if ($1 == "-h") then
goto synopsis
endif
```

```
set f=`ls *.spect`
set i=`expr index $f[1] '_'`
@ i--
set instr=`expr substr $f[1] 1 $i`
AcumData.csh > ${instr}_acumoid.data
graph -m 0 -g 1 -y 0 4 1 -l "SHARC $instr Acumoid" < ${instr}_acumoid.data | plot -Tdumb > ${instr}_acumoid.plot

goto end

synopsis:
echo "$0 produces a plot of acumoid data extracted from SHARC files"

end:

#! /bin/csh -f
#/*
#*****************************************************************************
#*     COPYRIGHT (C) CSIRO DIT CSIS 1991
#*
#*     SOURCE FILE     AcumData
#*
#*     MODULE          Sonify
#*
#*     SYSTEM          research
#*
#*     ENVIRONMENT     Sun SPARC and SUNOS4.1
#*
#*     AUTHOR          Stephen Barrass, CSIS
#*
#*     HISTORY
#*     19/4/95: first written   s.b.
#*
#*****************************************************************************
#*/
# filenames are in form instr_nnn.spect

if ($1 == "-h") then
goto synopsis
endif

foreach f ( *.spect )
set i=`expr index $f '_'`
set j=`expr index $f '.'`
@ i++
set l=`expr $j - $i`
set nnn=`expr substr $f $i $l`
set midi=`name2midi $nnn`
set hz=`midi2hz $midi`
set acumoid=`Acumoid -b $hz < $f`
echo $midi $acumoid $nnn
end

goto end

synopsis:
echo "$0 reads all SHARC analysis files with names instr_nnn.spect
and writes Acumoid data points to stdout"

end:
```

```
/*
*****************************************************************************
*COPYRIGHT (C) CSIRO DIT CSIS 1995
*
*SOURCE FILEAcumoid.cc
*
*MODULE     research
*
*SYSTEM       Sonify
*
*ENVIRONMENTSun SPARC and SUNOS4.1.2
*
*AUTHORStephen Barrass, CSIS
*
*HISTORY
*:FirstwrittenApril1995.
*****************************************************************************
*/
/* RCS history log
$Log$
*/
/* RCS revision identifier and equivalent string
$ID$
*/
static char rcsid[] = "$Id:$";

/*****************************************************************************/
#include <stream.h>
#include <math.h>
#include <string.h>

//---------------------------------------------------------------------
// critical band rate weighting factor
//
//            17       18       19       20       21       22       23       24
float w[] = {1.1,1.2,1.3,   1.5,      1.7,      2.0,      3.0,     4.0};

float
g(int i)
{
if (i < 17) return 1;
return(w[i-17]);
}

//---------------------------------------------------------------------
// excitation mask
// Zwicker & Fastl pp. 151
const MASK_SIZE = 10;
//               -2   -1   0    12    3    4    56    77
float mask[MASK_SIZE] = {0.10, 0.50, 1.00, 0.80, 0.60, 0.40, 0.20, 0.10, 0.05, 0.02 };

//---------------------------------------------------------------------
// synopsis
//
void
synopsis(char * prog)
{
cerr<<"Synopsis\n";
cerr<<"-------- \n";
cerr<<prog<<" [-h help] -b baseHz < stdin > stdout" << endl;
cerr <<"Arguments[default]// description\n";
cerr << "-----------------------------------------------------------------\n";
cerr<<"-h\t\t\t\tthis help message\n";
cerr<<"-b baseHz\t <100>\t\t\tbase frequency in Hz\n";
cerr<<"stin\t\t\tspectral analysis in SHARC format\n";
cerr<<"stdout\t\t\tspectral centroid\n";
```

```
cerr<<“Description\n”;
cerr<<“----------- \n”;
cerr<<“Calculate the Sharpness in ACUMs. (1 Acum is the sharpness of a narrow band noise centred at 1 kHz. Algorithm from Psychoacoustics by Zwicker & Fastl.\nThe input file consists of relative harmonic amplitudes wrt to peak in dB in column 1, and relative phase in column 2.\n”;
cerr<<“Stephen Barrass\n”;
cerr<<“CSIRO DIT\n”;
cerr<<“stephen@cbr.dit.csiro.au\n”;
cerr<<“\nCOPYRIGHT (c) CSIRO DIT 1995\n”;
}

//--------------------------------------------------------------------------
// main
//
main(int argc, char * argv[])
{
  int i;
float baseKHz = 100;

// process args
for (i=1; i<argc; i++)
    {
     if (strcmp(“-h”, argv[i]) == 0)
               {
               synopsis(argv[0]);
                    return 0;
               }
     else if (strcmp(“-b”, argv[i]) == 0)
               {
               baseKHz = atof(argv[++i])/1000.0;
               }

    }

const BARK_MAX = 24;// maximum Bark
float relativeAmp;
float Ndz = 0;
float Nzdz = 0;
float dB, radians;
float bandrate[BARK_MAX];
for (i=0; i<BARK_MAX; i++)
     bandrate[i] = 0;

float f = baseKHz;
float Bark;
int bin,binI;
cin >> ws;
while (!cin.eof())
    {
     cin >> dB >> ws >> radians >> ws;
// ignore phase in power spectrum
// dB = 20 * log(Comparison/Reference)
// therefore Relative = Comparison/Reference = 10 ^ (dB / 20)
     relativeAmp = exp10(dB/20);
// convert kHz frequency of this harmonic into Bark 1..24
     Bark = 13*atan(0.76*f)+3.5*atan((f*f)/56.25);
// integer Bark is boundary between critical Bands
     bin = nint(floor(Bark))-1;
       binI = bin-2;
// replace each bin with a mask
// use simple triangle shape for Loudness = 60
// (Zwicker & Fastl, pp 151)
     for (i=0; i<MASK_SIZE; i++, binI++)
               {
                  if (binI < 0)
                            continue;
```

```
                bandrate[binI] += mask[i]*relativeAmp;
               }
       f += baseKHz;
     }
cerr << endl;

// find see-saw point (first moment)
// moment calculation

Ndz = Nzdz = 0;
cerr << "Bandrate = ";
float bandAmp;
// the barks have been band-rate binned
// apply a log function to bin amplitudes for saturation effect
for (i=0; i<BARK_MAX;)
     {
// want a logarithmic saturation effect
//    bandAmp = log(bandrate[i]+1);
      bandAmp = bandrate[i];
      cerr << bandAmp << " ";
      Ndz += bandAmp;
// weighting factor  (aka Zwicker)
             ++i;
      Nzdz += bandAmp * i * g(i);
     }
cerr << endl;

float Acumoid =  Nzdz / Ndz;

// scale factor from Zwicker pp 218
Acumoid *= 0.11;

cerr << "Sharpness = " << setprecision(2) << Acumoid << " Acum" << endl;
cout << setprecision(2) << Acumoid << endl;

return 1;
}



/*
*********************************************************************************
*COPYRIGHT (C) CSIRO DIT CSIS 1995
*
*SOURCE FILEmidi2hz.cc
*
*MODULE     research
*
*SYSTEM        Sonify
*
*ENVIRONMENTSun SPARC and SUNOS4.1.2
*
*AUTHORStephen Barrass, CSIS
*
*HISTORY
*:FirstwrittenApril1995.
*********************************************************************************
*/
/* RCS history log
$Log$
*/
/* RCS revision identifier and equivalent string
$ID$
*/
static char rcsid[] = "$Id:$";
```

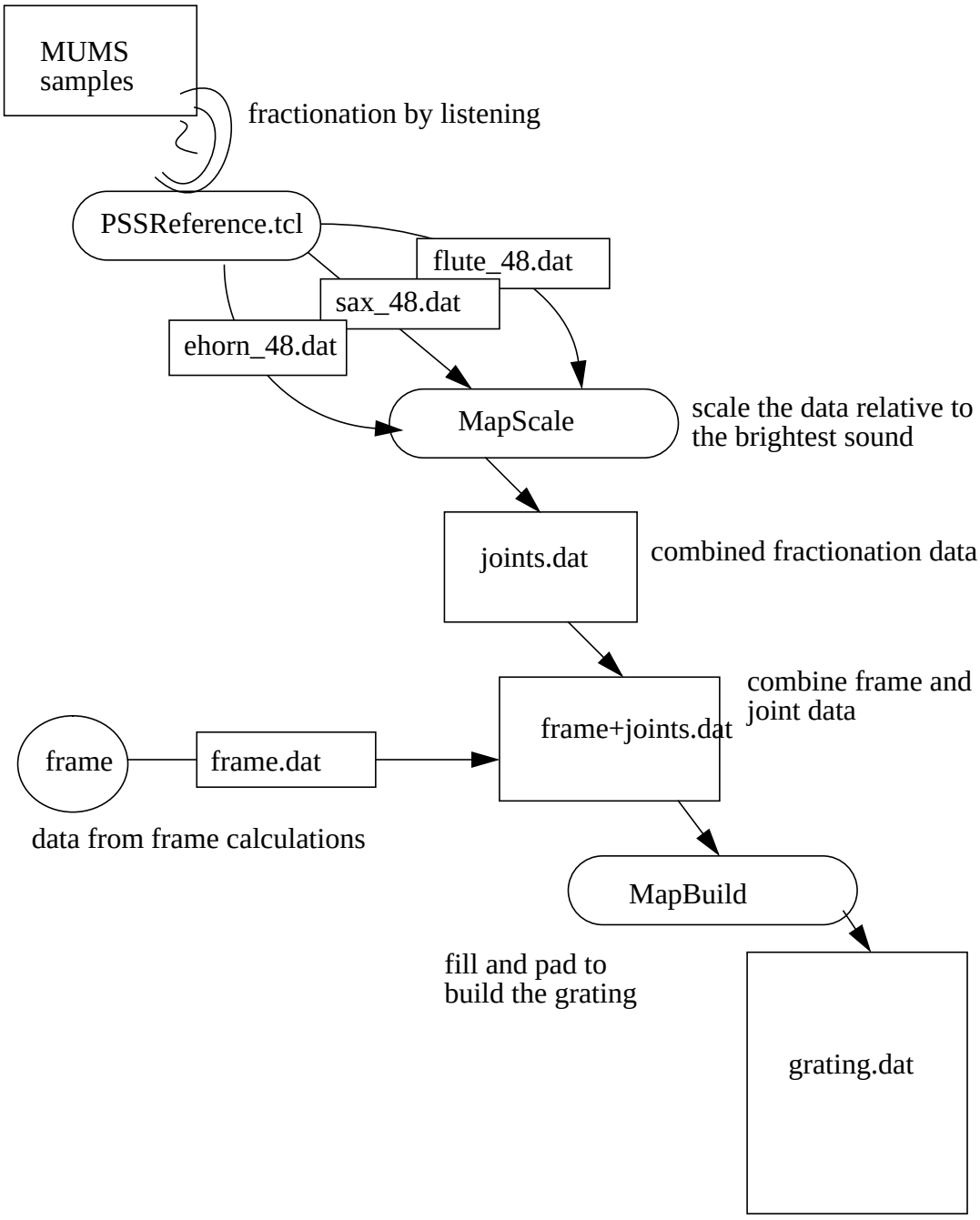
```
/*****************************************************************************/
#include <stream.h>
#include <math.h>
#include <string.h>


//---------------------------------------------------------------------------
// synopsis
//
void
synopsis(char * prog)
{
cerr<<“Synopsis\n”;
cerr<<“-------- \n”;
cerr<<prog<<“ [-h help] midinote > stdout” << endl;
cerr <<“Arguments[default]// description\n”;
cerr << “-----------------------------------------------------------------\n”;
cerr<<“-h\t\t\t\tthis help message\n”;
cerr<<“midinote\t\t\t0-127 midi note number\n”;
cerr<<“stdout\t\t\tfrequency in Hz\n”;
cerr<<“Description\n”;
cerr<<“----------- \n”;
cerr<<“Convert midi notenumber to frequency in Hz\n”;
cerr<<“Stephen Barrass\n”;
cerr<<“CSIRO DIT\n”;
cerr<<“stephen@cbr.dit.csiro.au\n”;
cerr<<“\nCOPYRIGHT (c) CSIRO DIT 1995\n”;
}

//---------------------------------------------------------------------------
// main
//
main(int argc, char * argv[])
{
  int i;

// process args
for (i=1; i<argc; i++)
    {
     if (strcmp(“-h”, argv[i]) == 0)
               {
                synopsis(argv[0]);
                     return 0;
               }
    }
float notenum = atof(argv[argc-1]);
float hz = 16.351625 * pow(2, notenum/12);
cout << hz << endl;
return 1;
}
```

```
/*
*****************************************************************************
*COPYRIGHT (C) CSIRO DIT CSIS 1995
*
*SOURCE FILEname2midi.cc
*
*MODULE     research
*
*SYSTEM       Sonify
*
*ENVIRONMENTSun SPARC and SUNOS4.1.2
*
*AUTHORStephen Barrass, CSIS
*
*HISTORY
*:FirstwrittenApril1995.
*****************************************************************************
*/
/* RCS history log
$Log$
*/
/* RCS revision identifier and equivalent string
$ID$
*/
static char rcsid[] = "$Id:$";

/*****************************************************************************/
#include <stream.h>
#include <math.h>
#include <string.h>


//---------------------------------------------------------------------------
// synopsis
//
void
synopsis(char * prog)
{
cerr<<"Synopsis\n";
cerr<<"-------- \n";
cerr<<prog<<" [-h help] notename > stdout" << endl;
cerr <<"Arguments[default]// description\n";
cerr << "-----------------------------------------------------------------\n";
cerr<<"-h\t\t\t\tthis help message\n";
cerr<<"notename\t\t\tnote name e.g. c4, g#2, D6, A#1 etc.\n";
cerr<<"stdout\t\t\tcorresponding midi notenumber 0..127\n";
cerr<<"Description\n";
cerr<<"----------- \n";
cerr<<"Convert note name to midi notenumber\n";
cerr<<"Stephen Barrass\n";
cerr<<"CSIRO DIT\n";
cerr<<"stephen@cbr.dit.csiro.au\n";
cerr<<"\nCOPYRIGHT (c) CSIRO DIT 1995\n";
}

//---------------------------------------------------------------------------
// main
//
main(int argc, char * argv[])
{
  int i;

// process args
for (i=1; i<argc; i++)
    {
```

```
    if (strcmp("-h", argv[i]) == 0)
            {
            synopsis(argv[0]);
                return 0;
            }
    }
char * nnP = argv[argc-1];
int base;
switch(*nnP)
    {
        case 'a':
        case 'A':
                base = 9;
                  break;
        case 'b':
        case 'B':
                base = 11;
                  break;
        case 'c':
        case 'C':
                base = 0;
                  break;

        case 'd':
        case 'D':
                base = 2;
                  break;
        case 'e':
        case 'E':
                base = 4;
                  break;
        case 'f':
        case 'F':
                base = 5;
                  break;
        case 'g':
        case 'G':
                base = 7;
                  break;
         default:
              cerr << "Error : Bad note name\n";
              synopsis(argv[0]);
                  return 0;
    }
 nnP++;
if (*nnP == '#')
    {
         base++;
         nnP++;
    }
int offset = atoi(nnP)*12;
cout << base+offset << endl;
return 1;
}
```

Appendix 8-4: Grating stage

The grating is built by filling in the frame with perceptually measured joints that are equal steps in perceived brightness. The process has the following steps.



These programs are listed below

PSSReference.tcl	GUI for fractionating the brightness of a sound sample
MapScale.csh	scale the data relative to the maximum brightness
MapBuild.cc	fill and pad the data to build a grating

```
##############################################################################
#COPYRIGHT (C) CSIRO DIT CSIS 1993
#
#SOURCE FILEPSSReference.tcl
#
#MODULE       Sonify
#
#SYSTEM      PostGrad
#
#ENVIRONMENTSun SPARC and SUNOS4.1.2
#
#AUTHORStephen Barrass, CSIS
#
#HISTORY
#               : First written Feb 1994
##############################################################################

#

# Source file identification for debugging ###################################
#

if {[info exists DEBUG]} {
if {$DEBUG} {
     puts stderr {$Header: /proj/vis/mira/stephen/camroot/src/app/sonify/tk/RCS/PSSReference.tcl,v 1.1 1994/03/29
00:51:13 stephen Exp stephen $}
    }
}

#- public variables---------------------------------------------

proc PSSReference::public {} {
Debug::proc {PSSReference::public}

}

#- private variables--------------------------------------------

proc PSSReference::private {path} {
Debug::proc {PSSReference::private}

uplevel 1 “global $path.num”
}

#- PSSReference::init-------------------------------------------
# initialise privates
proc PSSReference::init {path} {
Debug::proc {PSSReference::init}
PSSReference::private $path

set $path.num 0
}

#- PSSReference-------------------------------------------
# create a PSSReference
#

proc PSSReference {path args} {

Debug::proc {PSSReference}
Environment::global
Perceptual::public
Device::public
```

```
# start Csound with the PSSReference
Csound $path.cs

toplevel $path
wm title $path Fractionation

PSSReference::public
PSSReference::private $path
PSSReference::init $path

# heading stuff
frame $path.head

FileName $path.head.file \
     -load “PSSReference::read $path” \
     -save “PSSReference::write $path” \
       -mask *.map


NumericEntry $path.head.notenum \
     -width 128 -height 22 -labwidth 80 \
     -text Notenum \
     -from [set Device::NOTENUM_MIN] \
     -to [set Device::NOTENUM_MAX] \
     -command “PSSReference::setNotenum $path”

button $path.head.refharmonic \
     -text Harmonic \
     -command “$path.head.patch set 500”

button $path.head.refnoise \
       -text Noise \
     -command “$path.head.patch set 501”

button $path.head.refacum \
       -text Acum \
     -command “$path.head.patch set 502”

NumericEntry $path.head.patch \
     -width 128 -height 22 -labwidth 80 \
       -text Patch \
         -from 0 \
         -to 9999 \
     -command “PSSReference::setPatch $path”

NumericEntry $path.head.dur \
     -width 128 -height 22 -labwidth 80 \
     -text Duration \
      -allowfloat 1 \
     -from [set Device::DUR_MIN] \
     -to [set Device::DUR_MAX] \
     -command “PSSReference::setDur $path”

pack append $path.head \
     $path.head.file {top expand fillx} \
     $path.head.notenum {left} \
     $path.head.refharmonic {left} \
     $path.head.refnoise {left} \
     $path.head.refacum {left} \
     $path.head.patch {left} \
     $path.head.dur {left}

# scale
frame $path.scale
frame $path.scale.label
```

```
label $path.scale.label.filter \
     -width 10 -height 1 \
        -anchor w \
       -text {Filter}

button $path.scale.label.play \
         -text Play \
     -bitmap @$CAMROOT/lib/bitmap/Play.xbm \
     -command “PSSReference::play $path”

pack append $path.scale.label \
     $path.scale.label.play {top expand fillx} \
     $path.scale.label.filter top

# create the Tuner bank
frame $path.scale.bank
set num 0
set timbre 0
set referenceInstrument 1

set bright [set Perceptual::BRIGHT_MIN]
while {$bright <= [set Perceptual::BRIGHT_MAX]} {
     set keynum [expr “$num + 1”]
     Tuner $path.scale.bank.tuner$num \
               -timbre $timbre \
               -bright $bright \
               -timbreNoShow 1 \
               -pitchNoShow 1 \
               -notenumNoShow 1 \
               -velNoShow 1 \
               -brightNoShow 1 \
                    -inv 1 \
                -key $keynum

     pack append $path.scale.bank $path.scale.bank.tuner$num {left}

     Tuner::setEqbw $path.scale.bank.tuner$num [expr “$bright*[set Device::EQBW_RANGE]/[set Perceptu-
al::BRIGHT_RANGE]”]
     Tuner::setInstrument $path.scale.bank.tuner$num $referenceInstrument

        incr bright
         incr num
    }

set $path.num $num
pack append $path.scale \
     $path.scale.label left \
     $path.scale.bank top

# pack it all
pack append $path \
     $path.head {top expand fillx} \
     $path.scale {top}

$path.head.notenum set 60
}

#- PSSReference::toplevel----------------------------------------------
# create a toplevel pss reference
proc PSSReference::toplevel {path} {
Debug::proc {PSSReference::toplevel}

toplevel $path
PSSReference $path.pss
pack append $path $path.pss top
}
```

```
#- PSSReference::play-----------------------------------------------
# play the tagged sounds
proc PSSReference::play {path} {
Debug::proc {PSSReference::play}
PSSReference::private $path

set num 0
while {$num < [set $path.num]} {
    Tuner::play $path.scale.bank.tuner$num
        sleep 1
    Tuner::stop $path.scale.bank.tuner$num
        incr num
    }
}

#- PSSReference::setPitchH-------------------------------------------
# set the Pitch
proc PSSReference::setPitchH {path {pitchH 5}} {
Debug::proc {PSSReference::setPitchH}
PSSReference::private $path

# change the pitch
PSSReference::private $path

set num 0
while {$num < [set $path.num]} {
    Tuner::setPitchH $path.scale.bank.tuner$num $pitchH
    Tuner::setNotenum $path.scale.bank.tuner$num [Perceptual::getPitchN $path.scale.bank.tuner$num.perceptual]
        sleep 1
        incr num
    }
}

#- PSSReference::setNotenum------------------------------------------
# set the Notenum
proc PSSReference::setNotenum {path {notenum 60}} {
Debug::proc {PSSReference::setNotenum}
PSSReference::private $path

# change the pitch
PSSReference::private $path

set num 0
while {$num < [set $path.num]} {
    Tuner::setNotenum $path.scale.bank.tuner$num $notenum
        incr num
    }
}

#- PSSReference::setPatch--------------------------------------------
# set the Patch
proc PSSReference::setPatch {path {patch 0}} {
Debug::proc {PSSReference::setPatch}
PSSReference::private $path

# change the patch
PSSReference::private $path

set num 0
while {$num < [set $path.num]} {
    Tuner::setPatch $path.scale.bank.tuner$num $patch
        incr num
    }
}
```

```
#- PSSReference::setDur----------------------------------------------
# set the Duration
proc PSSReference::setDur {path {dur 1.0}} {
Debug::proc {PSSReference::setDur}
PSSReference::private $path

# change the duration
PSSReference::private $path

set num 0
while {$num < [set $path.num]} {
     Tuner::setDur $path.scale.bank.tuner$num $dur
          incr num
     }
}

#- PSSReference::read-----------------------------------------------
# read from file
proc PSSReference::read {path filename} {
Debug::proc {PSSReference::read}
PSSReference::private $path

# open the file
set fd -1
set fd [open $filename r]
if {$fd == -1} {
          return
     }
# read the header
# timbre leaf
set header ""
set timbre 0
gets $fd header
gets $fd header
set notenum 0
scan $header "# Notenum %d" notenum
#
# read each tuner
set num 0
while {$num < [set $path.num]} {
     Tuner::read $path.scale.bank.tuner$num $fd
          incr num
     }
close $fd
$path.head.notenum set $notenum
}

#- PSSReference::write----------------------------------------------
# write to file
proc PSSReference::write {path filename} {
Debug::proc {PSSReference::write}
PSSReference::private $path

# open the file
set fd -1
set fd [open $filename w]
if {$fd == -1} {
          return
     }
# write the header
puts $fd "# PSSReference"
puts $fd "# Notenum [$path.head.notenum get]"

# write each scale
# write each tuner
```

```
set num 0
while {$num < [set $path.num]} {
     Tuner::write $path.scale.bank.tuner$num $fd
          incr num
     }
close $fd
}

#- PSSReference::test----------------------------------------------
# write all leaves to a file
proc PSSReference::test {path {filebase tleaf}} {
Debug::proc {PSSReference::test}
PSSReference::private $path

}

#!/bin/csh -f
#
##############################################################################################
#COPYRIGHT (C) CSIRO DIT CSIS 1994
#
#SOURCE FILEMapScale
#
#MODULE       Sonify
#
#SYSTEM       Thesis
#
#ENVIRONMENTSun SPARC and SUNOS4.1.2
#
#AUTHORStephen Barrass, CSIS
#
#HISTORY
#               : First written May 1995
##############################################################################################

# Synopsis
#
if ($1 == “-h”) then
goto synopsis
endif

set Bmax = $1
set notenum = $2
set infile = $3
gawk -v Bmax=$Bmax -v notenum=$notenum -f MapScale.gawk < $infile

goto end

synopsis:
echo “$0 Bmax notenum mapfile”
echo “Bmax is maximum brightness in Acums”
echo “Create a pnf to TBP mapping from a PSSReference map”

end:

/*
;##############################################################################;#
;#COPYRIGHT (C) CSIRO DIT CSIS 1993
;#
;#SOURCE FILEMapBuild.cc
;#
;#MODULE     Sonify
;#
;#SYSTEM     PostGrad
;#
;#ENVIRONMENTSun SPARC and SUNOS4.1.2
```

```
;#
;#AUTHORStephen Barrass, CSIS
;#
;#HISTORY
;#              : First written May 1995
;#############################################################################
*/

#include <iostream.h>
#include <fstream.h>
#include <strstream.h>
#include <math.h>
#include <string.h>

int loNotenum = -100;
int hiNotenum = 200;
int hiFilter = 1000;
int OutOfGamutFilter = 140;
int extraB = 5;  // extra Brightness steps to confirm B boundary
float BMax = 2.0;
int PFillCnt = 0;// extra pitch steps to improve pitch accuracy
int BFillCnt = 6;// number of interpolated B points
int OutOfGamutPatch = 0;
int InGamutPatch = 100;
const PitchPadStep = 5;

char * helpString = "\n\
Arguments[default]// description\n\
---------------------------------------------------------------------- \n\
-h                      // print this help line \n\
-loP          0         // out-of-gamut lo pitch\n\
-hiP         127        // out-of-gamut hi pitch\n\
-hiFilter  200// out-of-gamut hi filter value\n\
-extraB  5   // number of B extrapolations\n\
-BMax     2.0         // maximum Brightness for extrapolation\n\
stdin       stdin        // read from stdin\n\
stdout  stdout// write to stdout\n\
\n\
Description \n\
----------- \n\
read Acum data and output pnfTBP \n\
\n\
Author \n\
----------- \n\
Stephen Barrass\n\
Visualisation \n\
CSIRO DIT \n\
stephen@csis.dit.csiro.au \n\
\n\
COPYRIGHT (C) CSIRO DIT CSIS 1995 \n\
";


//-BScale-----------------------------------------------------
// scale the equal brightness step points so that maximum brightness
// is calibrated with unfiltered brightness of this sample at this pitch
//
// the modelling software needs lots of points to do a good job,
// particulary for the non-linear (exponential) filter values
// fill-in extra filter values using interpolation between neighbours
// initially try simple linear interpolation
// extend filter values beyond gamut boundary to support the mapping
//
#ifdef TEST
int global_p = 0;
#endif
```

```
void
BScale
(
char * MeasureFile,// datafile with B for each P of this T
float Bacum127,// brightness with no filter (i.e filter = 127)
float Pitch, // pitch of this brightness measurement
int OutOfGamutNotenum = 0 // out-of-gamut Pitch sets this
)
{
  int p;
 float n;
int f = 0;
int fPrev;
int fFill;

float T,B;
B = 0.0;
// 8 equal steps up to the Brightness at this Pitch
float Bstep = Bacum127/8.0;
float Bdelta = Bstep/BFillCnt;

int i = 0;

// BScale
// open the Measurement file
ifstream MeasureStream(MeasureFile);
int startF = 1;

// skip whitespace
MeasureStream >> ws;
// end-of-file ?
while (!MeasureStream.eof())
    {
// skip commented lines
     if (MeasureStream.peek() == '#')
               {
                MeasureStream.ignore(1024,'\n');
                    continue;
               }
// save previous f value
          fPrev = f;
// get a point
     MeasureStream >> p >> n >> f >> ws;
// skip to end of line
     MeasureStream.ignore(1024,'\n');
     MeasureStream >> ws;

// if first line get next line so we can start interpolating
          if (startF)
               {
// use patch to find Timbre angle
#ifdef TEST
                T = (global_p-1)*45;
#else
                  T = (p-1)*45;
#endif
                    startF = 0;
                     continue;
               }
// ignore the notenum in the file for the moment since using just
// one measurement file for all pitches
// use Pitch instead
          n = Pitch;
// out-of-gamut Pitch
     p = InGamutPatch;
```

```
    if (OutOfGamutNotenum != 0)
            {
// XXX
//              n = OutOfGamutNotenum;
                p = OutOfGamutPatch;
//              f = OutOfGamutFilter;
                }
// heres the interpolated B fill
    for (i=0; i < BFillCnt; i++, B += Bdelta)
            {
            fFill = fPrev+i*(f - fPrev)/BFillCnt;
// write it out
            cout << p << " " << n << " " << fFill << " " << T << " " << B << " " << Pitch << endl;

            }
    }
// last f point
cout << p << " " << n << " " << f << " " << T << " " << B << " " << Pitch << endl;


// add extra B points to force gamut shape
int fStep = (hiFilter-f)/extraB;
float BStep = (BMax-B)/extraB;
f = OutOfGamutFilter;
p = OutOfGamutPatch;
//B = B*1.01;
B += BStep;
for (i=0; i<extraB; i++)
    {
//      f += fStep;
    cout << p << " " << n << " " << f << " " << T << " " << B << " " << Pitch << endl;
      B += BStep;
    }


}

//-PrePad-----------------------------------------------------


void
PrePad
(
char * MeasureFile,// datafile with B for each P of this T
float Bacum127,// brightness with no filter (i.e filter = 127)
float PitchStart,//
float PitchEnd
)
{
float Pitch;
for (Pitch = PitchStart; Pitch <= PitchEnd; Pitch+=PitchPadStep)
    {
    BScale(MeasureFile, Bacum127, Pitch, loNotenum);

    }
}

//-AfterPad---------------------------------------------------
void
AfterPad
(
char * MeasureFile,// datafile with B for each P of this T
float Bacum127,// brightness with no filter (i.e filter = 127)
float PitchStart,//
float PitchEnd
)
```

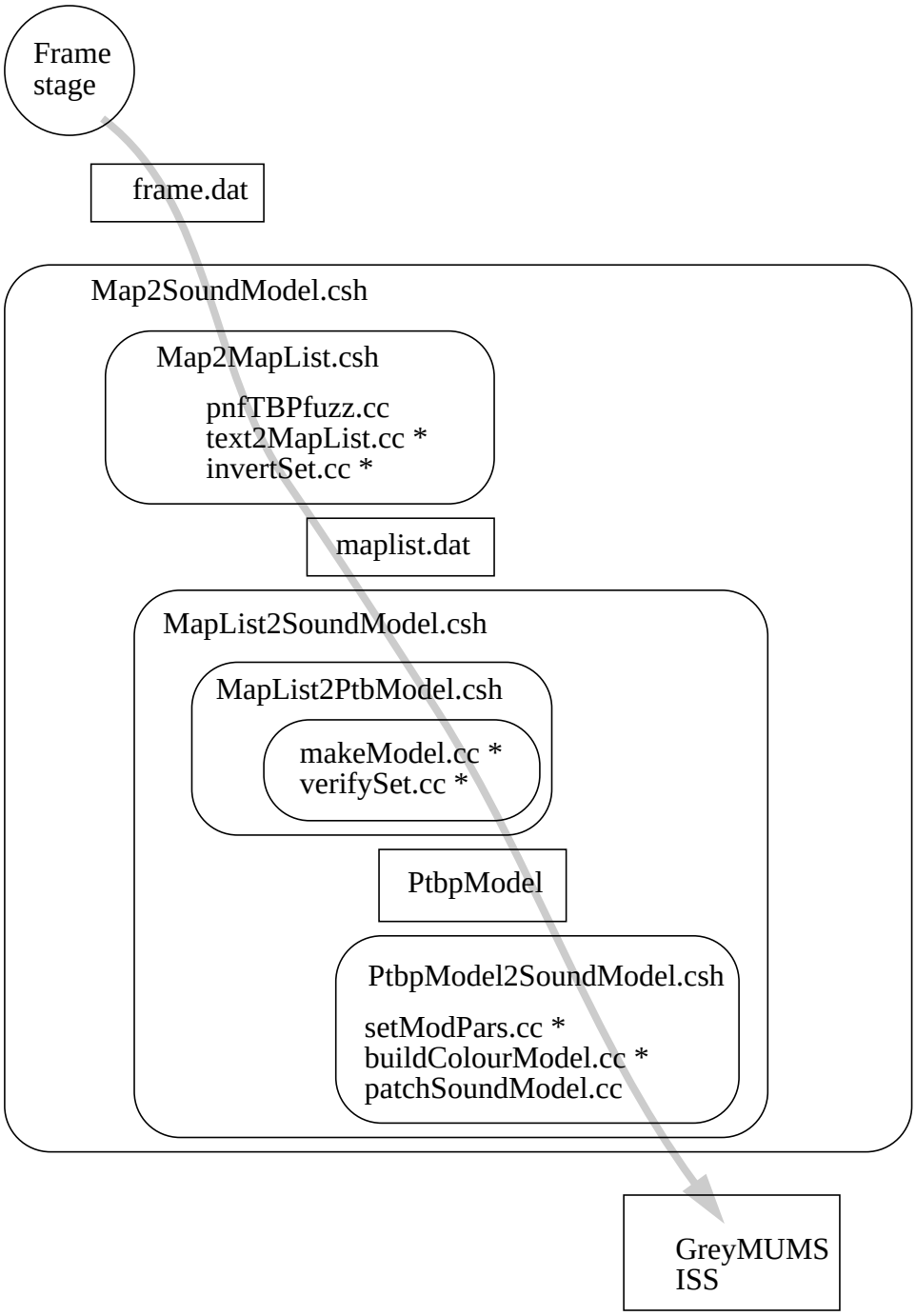
```
{
float Pitch;
for (Pitch = PitchStart; Pitch <= PitchEnd; Pitch+=PitchPadStep)
    {
     BScale(MeasureFile, Bacum127, Pitch, hiNotenum);

    }
}

//-main       ----------------------------------------------------
//

// this routine converts SHARC data to pnfTBP data suitable for the
// modelling routines. This process includes padding to help the
// modelling software smoothly drape flesh onto the bones
// which requires consideration of the hyper-linear interpolation algorithm
// The SHARC data consists of Brightness values for a Pitch range of a
// particular Timbre
// file format is
// stringfloatfloat
// Timbre pitchbrightness
//
// the Timbre data is arranged in increasing pitch order
// flute 48 0.5
// flute 49 0.6
// flute 50 0.45
// .. etc.
//
// when a new Timbre is started
// the last data from the previous Timbre is copied up the pitch axis
// to establish the upper half of the cylindrical contour
// then the new Timbre data is prepended is copied down the pitch axis
// to establish the lower cylindrical contour
//
//
main(int argc, char **argv)
{
  int i;
// parse the commandline
for (i=1; i < argc; i++)
    {
     if (strcmp(argv[i],"-h") == 0)
               {
                        i++;
                cerr << argv[0] << " " << helpString;
                    return 1;
               }
     else if (strcmp(argv[i],"-loP") == 0)
               {
                loNotenum = atoi(argv[++i]);
               }
     else if (strcmp(argv[i],"-hiP") == 0)
               {
                hiNotenum = atoi(argv[++i]);
               }
     else if (strcmp(argv[i],"-hiFilter") == 0)
               {
                hiFilter = atoi(argv[++i]);
               }
     else if (strcmp(argv[i],"-BMax") == 0)
               {
                BMax = atof(argv[++i]);
               }
     else if (strcmp(argv[i],"-extraB") == 0)
               {
                extraB = atoi(argv[++i]);
```

```
            }
    }

// output format
long formatF = cout.flags();
cout.flags(formatF | ios::fixed);
cout.precision(3);

char Timbre[64];
char TimbrePrev[64];
ostrstream(Timbre, sizeof(Timbre)) << ends;
float Bacum;
float BacumPrev = 0;
float Pitch;
float PitchPrev = 0;
float PitchMax = 127;
float p, pInc;

char MeasureFile[256];

unsigned char startFlag = 1;

// read SHARC database file from stdin
cin >> ws;
while (!cin.eof())
    {
// skip commented lines
    if (cin.peek() == '#')
            {
            cin.ignore(1024,'\n');
                continue;
            }

// read the line
// instrument_name pitch brigtness
    cin >> ws >> Timbre >> Pitch >> Bacum >> ws;

// is it a new Timbre ?
    if ((strcmp(Timbre, TimbrePrev) != 0))
            {
            if (strcmp(Timbre, "change") == 0)
                    {
                    cin >> ws >> Timbre >> Pitch >> Bacum >> ws;
                    }
            cerr << "change to " << Timbre << "," << Pitch << "," << Bacum << endl;

// change Timbre
// after-pad old Timbre
             if (!startFlag)
                    {
                    AfterPad(MeasureFile, Bacum, PitchPrev+1, PitchMax);
                    }
             startFlag = 0;

// get the new measurement file MeasureFile
// the measurement file contains the equal brightness filter values
// for this Timbre
// using just one file for all pitches at the moment
#ifdef TEST
              global_p++;
#endif
            ostrstream(MeasureFile, sizeof(MeasureFile)) << Timbre << "_" << 48 << ".map" << ends;

// pre-pad new Timbre
            PrePad(MeasureFile, Bacum, 0, Pitch-1);
// update old Timbre name
```

```
                ostrstream(TimbrePrev, sizeof(Timbre)) << Timbre << ends;

                }
        pInc = 1.0/(PFillCnt+1);
        p = Pitch-(PFillCnt/2.0)*pInc;
        for (i=0; i<=PFillCnt; i++)
                {
                BScale(MeasureFile, Bacum, p);
                    p += pInc;
                }
        PitchPrev = Pitch;
        BacumPrev = Bacum;
       }
// end of Timbre at eof
// after-pad
AfterPad(MeasureFile, Bacum, Pitch+1, PitchMax);

}
```

Appendix 8-5: Plasticine stage

The plasticine stage is a 3D spline fit to the grating data points. A set of scripts collects the necessary parameters from the user and automates the file conversion processes. In the process diagram below the colour modelling tools developed by Done Bone have an * next to them [Bone D. (1993)].



```
#!/bin/csh -f
#
#*******************************************************************************
#       COPYRIGHT (C) CSIRO DIT CSIS 1992
#
#       SOURCE FILE     Map2SoundModel.csh
#
#       MODULE          personify
#
#       SYSTEM          Research
#
#       ENVIRONMENT     Sun SPARC and SUNOS4.1.2
#
#       AUTHOR          Stephen Barrass, CSIS
#
#       HISTORY
#                       : First written May 1995
#*******************************************************************************
#
#
echo "**** Map2SoundModel.csh"

source ModelRange

#------------------------------------------------------------------------------
# command line
#
set files =
set fuzz = false

while ($#argv > 0)

        switch ($1)
                case -h:
                        goto synopsis
                        breaksw

                case -g:
                        shift
                set grid = $1
                breaksw

                case -f:
                set fuzz = true
                breaksw

                case -r:
                        shift
                set res = $1
                breaksw

                case -s:
                        shift
                set stiff = $1
                breaksw

                case -t:
                        shift
                set tight = $1
                breaksw

            default:
                set files = ($files $1)
                        breaksw
```

```
	endsw
	shift
end

#------------------------------------------------------------------------------
# main
#
main:
if ($fuzz == true) then
Map2MapList.csh -f
else
Map2MapList.csh
endif
MapList2SoundModel.csh -g $grid -s $stiff -t $tight -r $res

goto end

#------------------------------------------------------------------------------
# help
#
synopsis:
echo "$0 [-g grid <$grid>] [-f <$fuzz> (fuzz)] [-r res <$res>] [-s stiff <$stiff>] [-t tight <$tight>]"
echo "from pnfTBPGrey.map to SoundModel"

end:



#!/bin/csh -f
#
#*******************************************************************************
#	COPYRIGHT (C) CSIRO DIT CSIS 1992
#
#	SOURCE FILE	Map2MapList.csh
#
#	MODULE		personify
#
#	SYSTEM		Research
#
#	ENVIRONMENT	Sun SPARC and SUNOS4.1.2
#
#	AUTHOR		Stephen Barrass, CSIS
#
#	HISTORY
#			: First written May 1995
#*******************************************************************************
#
echo "**** Map2MapList.csh"

#
#------------------------------------------------------------------------------
# command line
#
set files =
set fuzz = false
set BMax = 2.0

while ($#argv > 0)

	switch ($1)
		case -h:
			goto synopsis
			breaksw

		case -f:
```

```
                set fuzz = true
                breaksw

            case -BMax:
                shift
                set BMax = $1
                breaksw

        default:
            set files = ($files $1)
                breaksw

    endsw
    shift
end

#-------------------------------------------------------------------------
# main
#
main:

if ($fuzz == true) then
pnfTBPfuzz -o1 5 < pnfTBPGrey.map | pnfTBPtoDpnfPtbp -BMax $BMax | text2MapList | sed 1,10s/TRUE_CAM/
FALSE_CAM/ | invertSet > PtbpDpnf.ml
else
cat pnfTBPGrey.map | pnfTBPtoDpnfPtbp -BMax $BMax | text2MapList | sed 1,10s/TRUE_CAM/FALSE_CAM/ | in-
vertSet > PtbpDpnf.ml
endif

MapListView.csh

goto end

#-------------------------------------------------------------------------
# help
#
synopsis:
echo "$0 [-f fuzz] [-BMax <$BMax>]"
echo "make DpnfPtbp.ml & PtbpDpnf.ml from pnfTBPGrey.map"

end:


#!/bin/csh -f
#
#*****************************************************************************
#    COPYRIGHT (C) CSIRO DIT CSIS 1992
#
#    SOURCE FILE    MapList2SoundModel.csh
#
#    MODULE         personify
#
#    SYSTEM         Research
#
#    ENVIRONMENT    Sun SPARC and SUNOS4.1.2
#
#    AUTHOR         Stephen Barrass, CSIS
#
#    HISTORY
#                   : First written May 1995
#*****************************************************************************
#
echo "**** MapList2SoundModel.csh"

source ModelRange
```

```
#
#-------------------------------------------------------------------------
# command line
#
set files =

while ($#argv > 0)

      switch ($1)
            case -h:
                  goto synopsis
                  breaksw

             case -g:
                     shift
            set grid = $1
             breaksw

             case -r:
                     shift
            set res = $1
             breaksw

             case -s:
                     shift
            set stiff = $1
             breaksw

             case -t:
                     shift
            set tight = $1
             breaksw

          default:
              set files = ($files $1)
                  breaksw

      endsw
      shift
end

#-------------------------------------------------------------------------
# main
#

main:
MapList2PtbpModel.csh -g $grid -s $stiff -t $tight
PtbpModel2SoundModel.csh -r $res
SoundModelView.csh -small

goto end

#-------------------------------------------------------------------------
# help
#
synopsis:
echo “$0 [-g grid <$grid>] [-r res <$res>] [-s stiff <$stiff>] [-t tight <$tight>]”
echo “from pnfTBPGrey.map to SoundModel”

end:


#!/bin/csh -f
#
#*****************************************************************************
#       COPYRIGHT (C) CSIRO DIT CSIS 1992
```

```
#
#       SOURCE FILE     MapList2PtbpModel.csh
#
#       MODULE          personify
#
#       SYSTEM          Research
#
#       ENVIRONMENT     Sun SPARC and SUNOS4.1.2
#
#       AUTHOR          Stephen Barrass, CSIS
#
#       HISTORY
#                       : First written May 1995
#*******************************************************************************
#
#
echo "**** MapList2PtbpModel.csh"
source ModelRange

#
#-------------------------------------------------------------------------
# command line
#
set files =
set viewF = true

while ($#argv > 0)

        switch ($1)
                case -h:
                        goto synopsis
                        breaksw

                case -g:
                        shift
                set grid = $1
                breaksw

                case -s:
                        shift
                set stiff = $1
                breaksw

                case -t:
                        shift
                set tight = $1
                breaksw

                case -Pt:
                        shift
                set PtMin = $1
                        shift
                set PtMax = $1
                breaksw

            case -Pb:
                        shift
                set PbMin = $1
                        shift
                set PbMax = $1
                        breaksw

            case -Pp:
                        shift
                set PpMin = $1
                        shift
```

```
            set PpMax = $1
                breaksw

        case -noV:
            set viewF = false
                breaksw

          default:
            set files = ($files $1)
                breaksw

      endsw
      shift
end

#-------------------------------------------------------------------------------
# main
# -C = monoticity constraint
#

main:
nice -20 makeModel -M PtbpDpnf.ml -o PtbpModel -a 0 $PtMin $PtMax -a 1 $PbMin $PbMax -a 2 $PpMin $PpMax
-W $stiff $tight -g $grid $grid $grid

verifySet -i PtbpDpnf.ml -m PtbpModel -o outliers.ml -d 10

# have a look at the results
if ($viewF == true) ModelView.csh -g $grid

goto end

#-------------------------------------------------------------------------------
# help
#
synopsis:
echo "$0 [-g <$grid>] [-Pt PtMin <$PtMin> PtMax <$PtMax>] [-Pb PbMin <$PbMin> PbMax <$PbMax>] [-Pp Pp-
Min <$PpMin> PpMax <$PpMax>] [-s stiff <$stiff>] [-t tight <$tight>] [-noV no view]"
echo "make a model from a maplist"

end:


#!/bin/csh -f
#
#*******************************************************************************
#       COPYRIGHT (C) CSIRO DIT CSIS 1992
#
#       SOURCE FILE     MakeSoundModel.csh
#
#       MODULE          personify
#
#       SYSTEM          Research
#
#       ENVIRONMENT     Sun SPARC and SUNOS4.1.2
#
#       AUTHOR          Stephen Barrass, CSIS
#
#       HISTORY
#                       : First written May 1995
#*******************************************************************************
#
#
echo "****** MakeSoundModel.csh"

source ModelRange
```

```
#-------------------------------------------------------------------------------
# command line
#
set files =
set slices = false

while ($#argv > 0)

	switch ($1)
		case -h:
			goto synopsis
			breaksw

		 case -s:
		set slices = true
		 breaksw

		 case -r:
			shift
		set res = $1
		 breaksw

	    default:
		 set files = ($files $1)
			breaksw

	endsw
	shift
end

#-------------------------------------------------------------------------------
# main
#

main:

setModPars -s .colour.UCS.Luv.Hsl -R 0 $DpMin $DpMax -R 1 $DnMin $DnMax -R 2 $DfMin $DfMax  < PtbpModel
| buildColourModel -r $res | patchSoundModel > SoundColourModel
cmodelVu -b -r -m SoundColourModel -o horizontal.ras -s 0
cmodelVu -b -r -m SoundColourModel -o vertical.ras -s 1

if ($slices == true) then
SoundModelView.csh
endif


goto end

#-------------------------------------------------------------------------------
# help
#
synopsis:
echo “$0 [-r <$res> gamut resolution] [-s = make slices]”
echo “make a SoundColourModel from a PtbpModel”

end:
```

Appendix 10-1: HeadSpace.orc

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Stephen Barrass
;;  research
;;  June 14 1995
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
; 24 tracks of 3d sound
; uses binaural delay, low-pass filter and amplitude
; to provide rough position and distance cues
;
; uses the zak gen to do array routing
;
sr = 22050
kr = 2205
ksmps = 10
nchnls = 2

; global        ------------------------------------------------
ginyq = sr*0.4
gipi = 3.14159265
giattack = 0.005
girelease = 0.005

; channels
gichannels = 24
zakinit gichannels,gichannels

; pseudo HRTF
gidelaymax = 0.001
gidelaymin = 0.0001
gifmin = 2.0/gidelaymax
gifscale = ginyq/2.171
; pinnae notch cue
ginotchf = 11000
ginotchw = 200

giamplut = 90                ; distance-to-amplitude lookup table

; angle of the ear relative to the side of the head (in degrees)
giearangle = 30
; ellipse
gie = 0.6                    ; eccentricity
gik = (1-gie)/gie            ; normalise

;
; cable
; plug a sound into a channel
; instr start   dur     channel patch   pitch
;
;==========================================================
; channel
instr   1

; input parameters           ---------------------------------------------
idur = p3
ich = p4                     ; channel
ipatch = p5                  ; patch
ipitch = p6                  ; pitch octave.class format 8.00 = middle C
; derived parameters         ---------------------------------------------
```

```
ifroot = cpspch(ipitch)         ; root frequency in hz
print idur, ich, ipatch, ipitch, ifroot

kamp    linen  1, giattack, idur, girelease ; declicking envelope
; use vibrato to bind streams
ivib    iunirand 40; random vibrato frequency
iamp= 4                 ; amplitude of vibrato
print ivib
kvib    oscil  iamp,ivib+40,91     ; holds stream together
; use pitchbend to point to a stream
kheyzkr    ich
ach     loscil  1, ifroot+kvib+khey, ipatch
zaw     ach, ich                ; route to the z channel
endin

;=========================================================
; attract attention to a stream
; by a brief pitchbend
; this writes to a zak variable which is read by the stream instr
; POINTER
; i2     start   dur    channel
instr   2

; input parameters              ---------------------------------------------
idur = p3
ich = p4                        ; channel

; derived parameters            ---------------------------------------------
print idur, ich

iamp =30                    ; bend size
khey lineniamp, idur/3, idur, idur/3; bend
zkwkhey, ich          ; write to the zk variable

endin

;=========================================================
; this placer uses the declick envelope
; placer
; place a channel in the scene
; and write to stereo output
;
;
; instr start   dur     channel distance        angle   height

instr   3

; input parameters              ---------------------------------------
idur = p3
ach      zar p4                ; channel to play
idistance = p5                  ; distance 0 to 127 metres
iangle = p6                     ; angle 0_360 degrees where 0 = right hand
iheight = p7                    ; height -10_10 below_level_above
print idur,idistance,iangle,iheight

; derived parameters            ---------------------------------------
idb     table   idistance, giamplut     ; convert distance-to-amplitude
iamp = ampdb(idb)
irevtime = idistance/5                  ; reverb in seconds
iradians = iangle*2*gipi/360            ; convert degreees-to-radians
print idb, iamp, iradians, irevtime

; delay
ix = cos(iradians)
idelay = abs(ix)*gidelaymax + gidelaymin
```

```
irdelay = (ix < 0 ? idelay : gidelaymin)
ildelay = (ix < 0 ? gidelaymin : idelay)

printks “ix = %f, idelay = %f, ildelay = %f, irdelay = %f\n “, 10, ix, idelay, ildelay, irdelay

; rough up a HRTF using elliptical functions for each ear
; right ear
irearangle = iangle-giearangle
irhrtf = gik*gie/(1-gie*cos(irearangle*2*gipi/360))

; left ear
ilearangle = iangle-180+giearangle
ilhrtf = gik*gie/(1-gie*cos(ilearangle*2*gipi/360))

print ilhrtf,irhrtf

; k-rate          ------------------------------------------------
; de-clicking envelope
kamp linen iamp, giattack, idur, girelease

; a-rate          ------------------------------------------------

; distance related reverb
;arev   reverb ach, irevtime

; angle positioning by binaural delay
; note : there really should be frequency dependence here...
; used fixed delay lines because deltap is stuffed !
al0 = ach*ilhrtf
ar0 = ach*irhrtf

al1   delay   al0, ildelay
ar1   delay   ar0, irdelay

; attenuate upper frequencies for head shadow
al2  tone    al1, exp(ilhrtf)*gifscale
ar2  tone    ar1, exp(irhrtf)*gifscale

; put in a frequency notch as a pinnae cue
;al3    areson al2, ginotchf, ginotchw
;ar3    areson ar2, ginotchf, ginotchw

;dispfft al2, 0.3, 1024
;dispfft ar2, 0.3, 1024

outs    al2*kamp, ar2*kamp

endin
```

Appendix 10-2: cOcktail.prl

```
#!/local/bin/perl5 -w
#
###############################################################################
#COPYRIGHT (C) CSIRO DIT CSIS 1994
#
#SOURCE FILEbatch.prl
#
#MODULE       Sonify
#
#SYSTEM       Thesis
#
#ENVIRONMENTSun SPARC and SUNOS4.1.2
#
#AUTHORStephen Barrass, CSIS
#
#HISTORY
#              : First written October 1995
###############################################################################

$dur = 10;
$scorefile = 'tada.sco';
$backdistance = 70;
$backscale = 20;
$angleplus = 13;

#------------------------------------------------------------------------
# help message
#
sub help {
print STDERR "Synopsis : $0\nGenerate the tada cocktail score\n";
print STDERR "[-h]\t\t\t\tthis help message\n";
print STDERR "[-d duration <$dur>]\t\tduration of each section\n";
}

###############################################################################
# subroutines
#

#------------------------------------------------------------------------
# placer
# place a channel in the scene
# and write to stereo output
#
$channelInst = 1;
$placerInst = 3;

# start duration distance angle height
sub placerHead {
print "\n;circle\n;instr\tstart\tdur\tchannel\tdist\tangle\theight\n";
}


sub placer {
local($timeBegin, $duration, $channel, $distance, $angle, $height) = @_;
print "i$placerInst\t$timeBegin\t$duration\t$channel\t$distance\t$angle\t$height\n";

}

#------------------------------------------------------------------------
# circle
# do circle movement
# timeBegin timeDur channel arcBegin arcAngle steps distBegin distEnd scorefile
```

```
sub dec2 {
my($in)=@_;
return (int($in*100)/100);
}
sub remains {
my($nominator, $divisor)=@_;
while ($nominator > $divisor) {
     $nominator -= $divisor;
     }
return dec2($nominator);
}
sub circle {
local($timeBegin,$timeDur,$channel,$arcBegin,$arcAngle,$steps,$distBegin,$distEnd,$scorefile) = @_;
$angle = $arcBegin;
$angleStep = $arcAngle/$steps;
$durStep = $timeDur/$steps;
$height = 0;

placerHead;
placer $timeBegin,$durStep,$channel,$distBegin,$angle,$height;

for ($i = 1; $i < $steps-1; $i++)
     {
     $angle = remains($angle+$angleStep,360);
#    $angle = ($angle+$angleStep)%360;
     placer '+', '.', '.', '>', $angle, '.';
     }
$angle = remains($angle+$angleStep,360);
#$angle = ($angle+$angleStep)%360;
placer '+', '.', '.', $distEnd, $angle, '.';
}

#-------------------------------------------------------------------------------
# readData
#

$channel = 1;

sub readData {
print STDERR readData;

local($datafile) = @_;
local($cm, $amount);
print STDERR "readData file $datafile\n";
# get each data line from the file
open(F,"$datafile") || die "Can't open file $!\n";

#local($angle) = 0;
local($angleStep);
local($distance) = 30;
local($height) = 0;
local($cmPrev) = -10;
local(@dataline);
placerHead;
placer 0,0,$channel,$distance,$angle,$height;
  $i=0;

while (<F>) {
        next if /^#/;
       next if /FILE/;
      next if /COLUMN/;
            $i++;
      if ($i>$length) {last;}
      @dataline = split();
# just get the last 2 columns in cm, amount
```

```
      $cm = $dataline[$#dataline-1];
      $amount = $dataline[$#dataline];
#     print STDERR "$cm,$amount\n";
#     $angle = ($angle+$amount+360)%360.0;

#     $distance = 1/exp(-abs($amount)/5);
#     $distance = int($distance*5);
      $distance = $backdistance-$backscale*abs($amount);
      $dur = (int($cm - $cmPrev))/10.0;

      placer '+',$dur,$channel,1+abs($distance),$angle,$height;
      $cmPrev = $cm;
     }
close(F);
$channel++;
$angle += $angleplus;
}

#--------------------------------------------------------------------------------
# lounge
#
sub lounge {
local($dur, $scorefile) = @_;
print STDERR "LOUNGE,$dur,$scorefile\n";
#system 'ls c*'
# start duration channel arcBegin arcAngle steps distBegin distEnd scorefile
circle 0,$dur,1,0,360,10,5,6;
circle 0,$dur,2,90,360,10,3,6;
circle 0,$dur,3,180,360,10,4,8;

}

#--------------------------------------------------------------------------------
# main
#
print STDERR "Cocktail\n";

#--------------------------------------------------------------------------------
# parse command line
#
$length = 10000;
$angle = 0;
while ($_ = $ARGV[0], /^-/) {
  shift;
if (/^-l/) {
      $length = shift;
     }
elsif (/^-h/)
     {
            help;
           exit 1;
     }
   else
     {
            help;
           exit 1;
     }
}

foreach $file (@ARGV) {
print STDERR "$file\n";
readData($file);
}

exit;
```

```
# add background
$backdur = 140;
$arcAngle = 360;
$backstep = int($backdur*4);
$backchn = 16;
# timeBegin timeDur channel arcBegin arcAngle steps distBegin distEnd scorefile
$placerInst = 3;
#circle(20,$backdur,$backchn,90,$arcAngle,$backstep, $backdistance, $backdistance);
#circle(30, 20, 17, 30, 1000, 20, 50, 100);
```