vs. computer

for percussion and interactive electronics

Cristyn Magnus

Instructions to accompany software score.

Introduction

This is a game piece. Not a strategy game or a social game, but a good, old-fashioned, arcade-type game—with the emphasis on old fashioned. Your task is to use your interpretation of the algorithmically generated score to move the green square around the game-field to pick up black squares before they disappear. If you pick up a square, you get a point. If a square fades out before you pick it up, the computer gets a point. The further ahead the computer is, the more it will mangle your sounds. If you lose, you have to stop playing and the computer gets to continue for a while. If you win, you will entirely shake off the computer's processing and get to play the last few phrases with no interference from the computer.

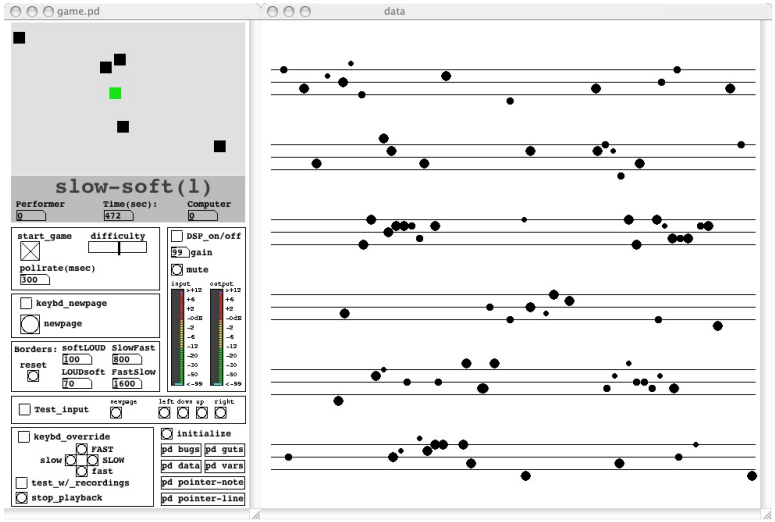

Figure 1: screenshot of patch

The piece and the user interface are described in more detail below.

Notation

The score is generated based on an underlying grammar and a developmental algorithm. This gives it the flexibility to be mapped on up to 22 instruments and to develop material progressively over the course of the game, regardless of how long a game takes. See figure 1 for a sample of algorithmically generated score.

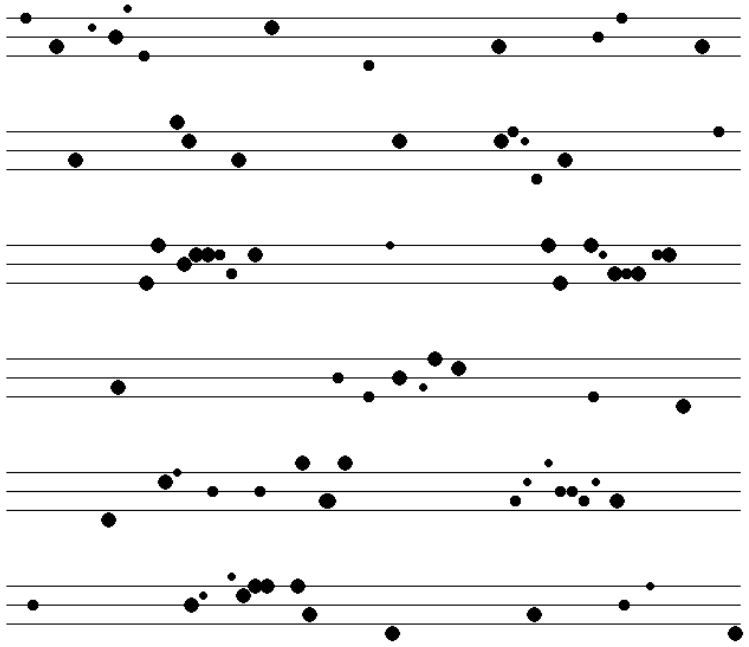

Figure 2: A sample algorithmically generated score.

Instrumentation

The score can be generated for 3–22 percussion instruments. The gestures described in the underlying grammar rely on a sonic continuum of some sort. The instruments should be arranged in order to produce a sonic continuum. That is, each instrument should share more sonic qualities with its neighbors than it does with instruments further away with it: If you play each instrument in order, it should produce a progression.

The exact instrumentation is unspecified for two reasons. First, the practice among percussionists is to modify instrumentation as part of the interpreting a piece. It makes more sense for me to describe the features I want highlighted than to ask for specific instruments without explaining my motivation. Second, I want to allow versatility in performance. This piece is designed to work with instruments on hand. It can expand to an array of larger instruments at home or map onto a small setup on the road.

Interpreting the Notation

There are four ways to interpret the notation; these translate into moves in the game. Playing fast and loud moves up; playing fast and soft moves down; playing slow and loud moves right, and playing slow and soft moves left. The notation is proportional but relative. The distance between notes translates to *relative* duration; the size of a note indicates its *relative* loudness. For example, to play soft you might map the smallest note size to *pp*, the medium note size to *p*, and the largest note size to *mp*; to play loud you would map the note sizes onto *mf*, *f*, and *ff*, respectively. An analogous mapping would take place with duration.

The form of the piece emerges from the interaction between you, the notation, and the game. The difficulty in the game

lies in interpreting the score so that the computer understands your moves. The score deliberately contains extremes to create tension between the notation and the move you're trying to make. To play slowly, you must play slowly enough that the densest passage is perceived as slow; to play quickly, you must play fast enough for the sparsest passage to be perceived as fast. You shouldn't change your speed and loudness mappings during a single move. For instance, you shouldn't compensate for dense passages by playing a *ritardando*. You either need to explicitly change your movement direction and play slower next time you attempt the move or accept any incidental direction change the computer might perceive and continue with the same mapping. You are allowed, however, to fool the computer by playing anything that is legal under the notation. For example, if you find that a sustained attack on a particular drum is often picked up as multiple, fast attacks, you could play a sustain during an extremely sparse section of the score to trick the computer into thinking you are playing faster than the notation suggests.

Visual Cues

Several visual cues are provided for the performer. The game-field (figure 3) shows most of the information needed to play the game. The game should not be projected for the audience, since this would distract them from the sound.

The top of the game-field contains the game itself. You move the green square around the area trying to pick up black squares. There will be a maximum of five black squares on the screen at a time. The black squares will fade to grey before they disappear.

If you pick up a square, you get a point. This is displayed in the bottom left-hand corner of the game-field. If a square fades out before you get it, the computer gets a point. This is displayed in the bottom right-hand corner of the game-field.

To calibrate your playing, the move you are currently making is displayed. You are told whether the computer thinks you are playing fast or slow, loud or soft. The direction of movement—up (u), down (d), left (l), right (r)—is also given here.

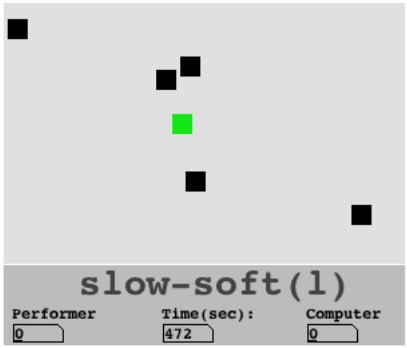

Figure 3: The game-field contains the game itself, the move you are currently making, the score, and the time left in the game.

The center-bottom of the game-field displays the amount of time (in seconds) that is left in the game. When you time runs out, the words "`game_over`" will appear in the center of the game-field.

Aural Cues

The state of the game is reflected in electronic processing. Each time the performer gets a point, a sound that rapidly increases in pitch is played; each time the computer gets a point, a sound that rapidly decreases in pitch is played.

The point distribution is reflected in the amount of processing the performer's sound undergoes. If the performer is well ahead of the computer, subtle effects like reverb are applied to his/her sound. As the computer catches up, the amount of reverb will increase. Then increasingly intrusive effects are added; first pitch shift, then delay (again, of increasing intensity). If the performer loses, the settings on the processing will become sufficiently extreme that the performer's sound will be swallowed up in gritty, aggressive sounds.

If the performer wins, the computer's processing will be completely turned off. The performer will then continue to play the next portion of the score, free from the constraints required to make moves in the game. He/She should play until he reaches a satisfying gesture on which to end the piece. Unless the game ended at the end of a page, he should find an end point on the current page. Otherwise, he should end somewhere on the next page.

The time left in the game also affects processing. There is a multiplier that reflects the amount of time left. There is always less processing early in the game, and processing reflects the point distribution with increasing accuracy over the course of the game.

The game-field itself is projected spatially onto the audience. The processed sounds will be spatialized to reflect the location of the green square. The sounds made by the computer getting points will be spatially located analogously to the location of the black squares that just faded out. For flexibility, the piece works with 2, 4, 6, or 8 speakers. Note, however, that a stereo projection will lose the vertical dimension of the game.

Technical Setup

The patch takes two audio inputs, but if more mics are desired for a particularly large percussion setup, mix them to two busses on the mixer.

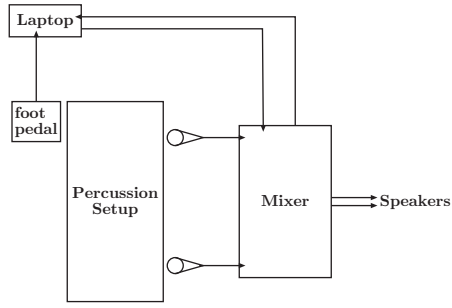

Figure 4: Technical setup.

In development, we

used a MOTU 828 to get sound in and out of the computer. This has a foot pedal input that is mapped onto a keyboard input. We mapped it to the space bar, which is what the patch is set up to take. If you want to switch to a midi foot pedal input, you'll need to edit the patch. Of course, you can just get a cheap usb keyboard and set it on the ground and step on it.

The patch is designed to work with 2, 4, 6, or 8 speakers. See § for configuration instructions.

Patch Overview

All of the relevant controls can be accessed from the main `game.pd` patch (figure 5). The remainder of this section discusses the controls in detail.

Game-field (a).
The game-field is described in §. It displays all of the information relevant to playing the game. In addition, you can set the duration of the game with the box `Time(sec)` in the bottom center of the game-field. The time box works like a microwave interface. If you type in 90, the game will last 90 seconds. If you type in 130, the game will last a minute and thirty seconds.

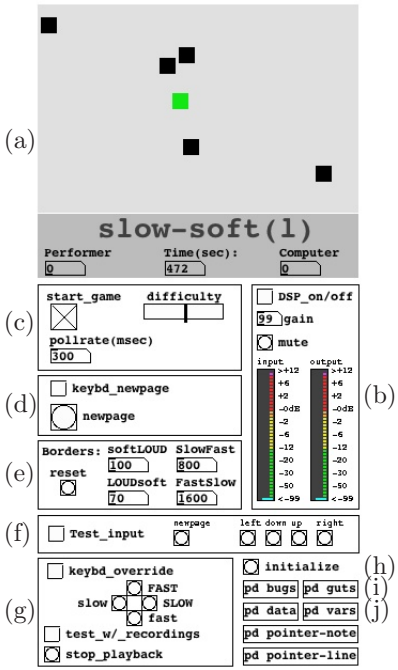

Figure 5: This is the main `game.pd` patch.

General Controls (b). The general controls let you turn the DSP engine on and off, adjust the volume, and meter input and output. The DSP must be **on** for the patch to analyze input or make sounds.

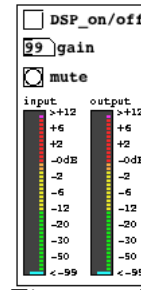

Figure b: Meters and General Controls

Output `gain` also must be turned up for the patch to make noise, but it isn't necessary for analysis. Gain is in dB, so 100 is a reasonable maximum value. However, the patch lets you go higher in case you need it. The gain will automatically fade out to zero at the end of the piece.

There is a `mute` button in case of emergency. Pushing it the first time will bring the gain to 0dB. Pushing it a second time will return the gain to its previous value.

There are two meters that reflect the `input` and `output` levels. The two inputs and n outputs are condensed onto a single input and output meter pair to conserve screen real estate.

Game Controls (c).
The `start_game` button starts the game. You should set a duration using the control at the bottom center of the game-field before pushing this button. In order to have the patch operate properly, turn the DSP on and bring the gain up to a reasonable level before pressing `start_game`.

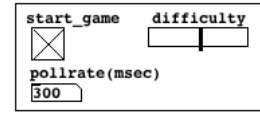

Figure c: Game Controls

The `pollrate` determines the rate at which moves are made if you are continuing in the same direction. The direction is polled every n msec to repeat the last move. If the movement direction changes, this takes effect as soon as the patch de-

tects a new movement. The use of the word "immediately" is a bit misleading. It changes as soon as it detects a new movement. Since speed is calculated based on the duration between notes, there can be a significant lag in determining a transition from fast to slow. It's a bit like driving a boat: you'll need to plan ahead.

The `difficulty` slider adjusts the difficulty of the piece. To make the game easier, move the slider to the left; to make the game harder, move the slider to the right. Since the piece is substantially less interesting if you can win too easily or can't win at all, you ought to adjust this to reflect your current skill. The game should pose a sufficient challenge that you win occasionally with significant effort. Technically, `difficulty` adjusts the number of turns the black squares tend to stick around before vanishing. With the default difficulty, it is possible to cross the entire game-field before squares start disappearing. If you could play perfectly, you would win every game at default difficulty. Further levels of difficulty lessen the amount of time the squares stay around such that you aren't guaranteed to be able to reach them all before they disappear, even if you move with maximal efficiency.

New Page (d). Pushing the `newpage` button generates a new page of score. This is generated based on earlier material, so you shouldn't use it to start the piece over without first initializing. The `keybd_newpage` button turns on and off the ability to use the space bar to display a new page. You'll probably want this on when playing the piece but off if you need to edit or test the patch or without advancing the score.

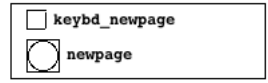

Figure d: New Page

Borders (e). The borders between states are quite wide: ambiguous material will be considered a move in the previous direction unless it is sufficiently decisive. If the computer is failing to properly classify your loudnesses and durations, you may need to adjust these thresholds.

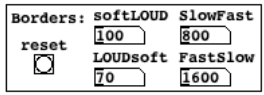

Figure e: Borders between loud/soft and fast/slow.

You will probably not need to adjust the duration borders. These default to 800 msec to transition from slow to fast and 1600 msec to transition from fast to slow.

The transition thresholds for volume default to 100 dB to transition from soft to loud and 70 dB to transition from loud to soft. These values are based on the particular setup used in the development of the patch. Appropriate settings will vary widely depending on your mics, instruments, room, *etc.*, and you will almost certainly need to adjust these before playing the game.

Test Interface (f). Since you don't want the act of making sure pedals, keyboards, and whatnot are properly plugged in to advance the piece in any way, here's a nifty test interface. Turn on `Test_input` and the buttons to the right will blink when the patch detects key presses.

Figure f: Test Interface

Override (g). This is a general control to override the computer and the game. You can click on the four buttons `FAST`,

`SLOW`, `fast`, and `slow`, to force the green square to move a particular direction. They also provide a nice, visual mnemonic in case you forget the mapping—the buttons labeled in caps are loud and the buttons labeled in lower case are soft.

The `keybd_override` button switches on the ability to use the keyboard to make moves. This uses the standard letter to direction mapping from vi and some old-school computer games: J is down; K is up; H is left; L is right. This is a relic from testing the patch during development, but it's nice to know it's there. If anything goes terribly wrong during a performance, you can whack the keys with a mallet to get things to work. It might be bad for your computer, but it is guaranteed to be cathartic.

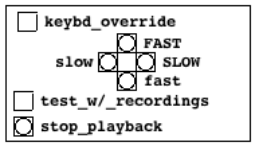

Figure g: Override

The `test_w/_recordings` and `stop_playback` buttons are relics from initially debugging the patch. `Test_w/_recordings` causes the override buttons or keys to play samples of the appropriate playing style. `Stop_playback` stops the playback of whatever sample is currently playing.

Initialize (h). Initialize resets all variables to their initial position. You should push this if you want to start the piece all over again. If you're going to do this a lot, you may want to adjust the default settings to fit your own setup. You can do this by editing the `pd preset` sub-patch inside the `pd vars` sub-patch.

Spatialization (i). The `guts` subpatch will open up if you click the `pd guts` box in the bottom right corner of the main `game.pd` patch. To change the number of output channels, you need to actually edit the patch. There are two things you need to change. If you want to be in stereo, the fourth box down should say `audio-stereo`; if you want to be in 4 or more channels, it should say `audio-multi`. The sixth box down should say `spat2` for 2 channels, `spat4` for 4 channels, `spat6` for 6 channels, and `spat8` for 8 channels.

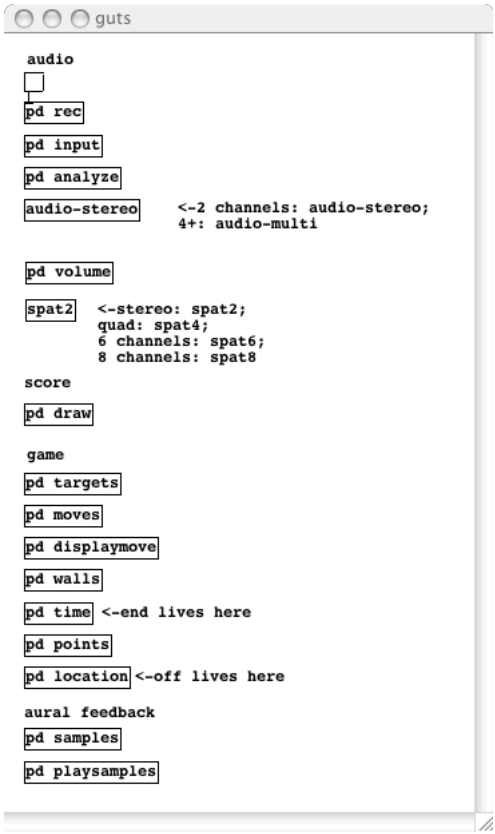

Figure i: Where most of the underlying program lives.

Adjusting the score (j). The `vars` subpatch will open up if you click the `pd vars` box in the bottom right corner of the main `game.pd` patch. The score defaults to seven instruments. You can have as many as twenty-two instruments. The display will draw notes on staves of five or less lines and will use the spaces as instruments as well. If you choose more

than eleven instruments, the display will have two staves per system, with a larger gap between systems.

The number of systems per page defaults to six, which is a reasonable setting for 7 instruments on a 1024x768 display. You may want to adjust this if you are using a different number of instruments or have a different display size.

Depending on the size and resolution of the display, you may want to adjust other display parameters. You can change the number of pixels between lines of the staves and between systems. You can adjust the top margin by setting the number of pixels before the first line appears. You can change the horizontal spacing of the notes—this shouldn't affect how you interpret them, but you may decide that you're turning pages more often than you like and want to have them closer together or that they are too close together to be legible. You can also set the width of the staves in pixels so you can change screen resolution.