SWARMS: Specifications

B. Hemingway and B. Mayton
Last revised November 30, 2007 2:53 PM

Background

Swarms is based on the AI concept of Swarm Intelligence:

http://en.wikipedia.org/wiki/Swarm_intelligence

An example of this is the famous BOIDS, which is a visual simulation of a flying flock of birds. It is characterized by simple rules:

- **separation**: steer to avoid crowding local flockmates
- **alignment**: steer towards the average heading of local flockmates
- **cohesion**: steer to move toward the average position of local flockmates

The variant of using separate physical agents derives from the field of Swarm Robotics:

http://www.swarm-robotics.org/

Description

A Swarm is a cloud of sound events of related character stretching a time period of ten seconds to thousands of seconds. The choices and characteristics of the sounds derive from the execution of a set of common rules, without the intervention of any central control. Therefore, the resultant effect falls under the category of *emergent behavior*.

Swarms

Swarms are defined by Pure Data patches that generate the sound associated with the swarm. There are a total of 8 patches:

- 1 patch executed when an agent is not part of a swarm (represented by swarm ID 0)
- 8 distinct patches, arranged in a continuous timbral space (represented by IDs 1-8) **(TBD)**

Implementation Details

Each group will implement their own version of a "Cooperative compositional agent" using the specifications listed in this document. Your compiled code will run on both your and your partner's "agents" for the Swarms demonstration **during the final class time, 12:30 PM, on Friday, December 7 in the Atrium.**

You will need to qualify your agent before it can participate in the Swarms demonstration. If, for some reason you are unable to qualify your agent, an alternative agent program will be provided so that you may receive your participation points for the demonstration.

Hints

- You should make use of enum types, constants for default values, and structs to simplify future modifications.
- Try to minimize the number of divides and mods used in your program. Calculate the needed values once and store the results. A memory read is a lot faster than a divide and/or mod.
- Use the `((packed))` attribute on any structs that define packets and radio protocols.

User Interface

Each group will implement a graphical interface using the LCD, jog dial, and pushbutton switches. This interface will allow the user to do the following:

- Set Zone 1-6. The zone number is a variable that is passed to the Pd patches which can affect the generated sounds.
- Manually start and stop the agent
- Display status information, such as swarm number, transmit power, average receive strength, and other parameters as necessary.

Addressing

You will be assigned a node ID that should uniquely identify your agent. This node ID is your agent's address on the radio network.

There are two special node IDs/addresses:

- `0x00` is the master controller
- `0xFF` is the broadcast address

Your agent should ignore packets that weren't sent to its address or the broadcast address.

Swarm Rules

The following rules are used to decide which swarm to perform.

```
For all entries in the circular FIFO {
   Calculate weights for swarms 0-7.  Weight is defined as the sum of
   the receive signal strengths from all agents currently performing a
   particular swarm.
}

Find the swarm with the largest weight.  Store its number in
max_swarm_num and its weight into max_weight.
```

```
Find the swarm with the smallest non-zero weight.  Store its number in
min_swarm_num and its weight into min_weight.

x = rand() % probability
y = rand() % silence

if(x==0) {
   Set current_swarm_num to min_swarm_num
} else if(y==0) {
   set current_swarm_num to zero
   goto SWARMLESS_WAIT_STATE
} else {
   if(max_weight < min_threshold || ((max_weight > threshold) && (You
   have already done max_swarm_num more than repetition times)) ) {
      Set swarm_num to a random swarm not among the last three swarms
      you've done more than repetition times
   } else {
      Set swarm_num to max_swarm_num
   }
}
```

Notes on rules

- Variables in **bold** are global parameters.
- The **threshold** and **repetition** count are meant to ensure that swarms are allowed to propagate through the performance space, but then die off after a while.
- The repetition allows a strong swarm to propagate to a large number of nodes, but once it has played for a while, it should die off. This growth and die-off is accomplished by limiting the number of repetitions once the threshold is reached.
- The zone number is used by specific swarms to pick the average center frequency, similar to how the barometer could be used to determine the floor.
- Important: Note that strength, in this protocol, is defined from 0-255, with 0 being the weakest and 255 being the strongest. *These are different semantics* than the RSSI value you get from the radio driver, so you'll have to do a simple conversion.

State Machine Definition

STOP_STATE
 	
(state machine starts in this state)
	Always {
	Wait for commands from controller and execute commands that arrive.
	
	Go to SWARMLESS_WAIT_STATE if start button is pushed, or if stop_and_listen packet received from controller
	}

SWARMLESS_WAIT_STATE	**On entering state** {
	set LED to RED
	
	set timer to random time between min_wait and max_wait
	
	Launch Pd with the wait state patch
	}
	
	Always {
	Listen for incoming packets
	
	Execute commands from controller
	
	Aggregate data from swarm messages from other agents into circular FIFO
	}
	
	On timer runout {
	Restart timer to random time between min_wait and max_wait
	
	Evaluate rules
	
	Send swarm_message packet
	
	Do result of evaluation
	}

JOIN_SWARM_STATE	**On entering state** {
	Stop Pd if it is running
	
	Start Pd with patch for the current swarm number
	
	Send initialization command to patch
	
	Set LED to GREEN
	
	Goto SWARM_STATE
	}

SWARM_STATE	**On entering state** {
	Set timer to random time between min_wait and max_wait
	}
	
	Always {
	Listen for incoming packets
	
	Execute commands from controller
	
	Aggregate data from swarm messages from other agents into circular FIFO
	}
	
	On timer runout {
	Evaluate rules
	
	Send swarm_message
	
	Do result of evaluation
	}

END_SWARM_STATE	**On entering state** {
	Set LED to BLUE
	
	Send stop command to patch
	
	Set timer to max_wait
	}
	
	Always {
	Listen for incoming packets
	
	Execute commands from controller
	
	Aggregate data from swarm messages from other agents into circular FIFO
	}
	
	On "finished" message from patch OR timer runout {
	Stop Pd
	
	Goto SWARMLESS_WAIT_STATE
	}

Radio Protocol Specification

This document describes the communications protocol and the structure of the packets that will be used by the swarm agents to communicate, and by the controller to alter various parameters of the swarm.

Packet Header

All packets, regardless of type, should carry the same 3-byte header, which is defined as follows:

`uint8_t`	`type`	The type of the packet
`uint8_t`	`src`	Node ID of the agent sending this packet
`uint8_t`	`dest`	Node ID of the destination agent, or 0xFF for broadcast. Agents should ignore packets unless the destination is their own node ID or the destination is broadcast (0xFF).

Packet Types

The `type` field in the packet is a number to specify the type of the packet and the format of the rest of the data that follows. These packet types are defined in the subsequent sections.

Set Global Parameter

This packet is a request from the controller for the agent to change one of its global parameters.

It has a `type` of `50`.

`uint8_t`	`var`	The ID of the variable/parameter to alter
`int16_t`	`value`	The new value for the parameter

Listen

This packet from the controller tells the agent to stop any current swarm and go to `SWARMLESS_WAIT_STATE`.

If an agent is currently in `STOP_STATE` and it receives a Listen packet, it should go to `SWARMLESS_WAIT_STATE`.

It has a `type` of `51`.

It has no payload.

Do Swarm

This packet from the controller tells the agent to stop any swarm currently in progress and execute a particular swarm.

It has a `type` of `52`.

`uint8_t`	`swarm_num`	The number of the swarm to execute.

Stop

This packet from the controller tells the agent to stop any current swarm and go to `STOP_STATE`.

It has a `type` of `53`.

It has no payload.

Swarm Message

This is the main packet type sent out every 3-6 seconds by your agent with information about itself, including the current swarm and what other swarms it sees.

It has a `type` of `42`.

`uint8_t`	`seq_num`	Sequence number. Start at 0 and increment on each packet sent.
`uint8_t`	`swarm_num`	The ID of the swarm of which this agent is currently a member. 0 if this agent is not part of a swarm.
`uint16_t`	`weight`	The weight of the swarm of which this agent is currently a member. 0 if this agent is not part of a swarm.
`uint8_t`	`max_swarm_num`	The ID of the swarm with the highest weight.
`uint16_t`	`max_weight`	The weight of the swarm with the highest weight.
`uint8_t`	`max2_swarm_num`	The ID of the swarm with the second highest weight.
`uint16_t`	`max2_weight`	The weight of the swarm with the second highest weight.
`uint8_t`	`min_swarm_num`	The ID of the swarm with the lowest non-zero weight.
`uint16_t`	`min_weight`	The weight of the swarm with the lowest non-zero weight.
`uint8_t`	`top_num`	The ID of the agent with the highest received signal strength.
`uint8_t`	`top_strength`	The received signal strength of the agent with the highest received signal strength.
`uint8_t`	`zone_num`	The zone number of this agent.
`uint8_t`	`txpower`	The transmit power this node is currently using.

Global Parameters

The following is a list of the parameters that the master controller should be able to set on your agent. Note that this list will probably change (particularly, new parameters may be added and the default values may change) as the project progresses.

Number	Name	Default Value
0	min_wait	3000 msec
1	max_wait	6000 msec
2	threshold	600
3	min_threshold	100
4	probability	10
5	silence	10
6	repetition	3
7	txpower	20
8	LED color	0
9	zone_num	1