Nyquist Software

- Nyquist plug-in development can be done with standalone Nyquist software [https://sourceforge.net/projects/nyquist/](https://sourceforge.net/projects/nyquist/)
- The software operates in an interrogative mode:
  - Enter commands in top-left window
Nyquist Software

- Nyquist plug-in development can be done with standalone Nyquist software [https://sourceforge.net/projects/nyquist/](https://sourceforge.net/projects/nyquist/)

- The software operates in an interrogative mode:
  - Enter commands in top-left window
  - Completion List window shows possible completions for what has been typed, with links to documentation
Nyquist Software

- Nyquist plug-in development can be done with standalone Nyquist software [https://sourceforge.net/projects/nyquist/](https://sourceforge.net/projects/nyquist/).
- The software operates in an interrogative mode:
  - Enter commands in top-left window
  - Completion List window shows possible completions for what has been typed, with links to documentation
  - Results of commands show in output window.
Nyquist Software

- Nyquist plug-in development can be done with standalone Nyquist software [https://sourceforge.net/projects/nyquist/](https://sourceforge.net/projects/nyquist/).
- The software operates in an interrogative mode:
  - Enter commands in top-left window.
  - Completion List window shows possible completions for what has been typed, with links to documentation.
  - Results of commands show in output window.
  - The top-right pane is a graphical editor for envelopes.
Sample Code - Generating Sound


Examples use SAL, a newer language which is an alternative to Xlisp:

- set pitch-source = make-cycle(list(c4, d4, e4, f4))
- play seqrep(i, 13, pluck(next(pitch-source), 0.2))

Makes `pitch-source` a cyclical list of four pitches, (c4, d4, e4, f4)

Get the next pitch and play it, 13 times
i=0; i<13; i++

PLAY IT

PLAY IT
Markov Models

- From Algorithmic Composition: A Guide to Composing Music with Nyquist
  By Mary Simoni, Roger B Dannenberg, 2013.
  Sect. 13.4-13.6, pp. 170-181
Markov Modeling

“Go Tell Aunt Rhody”:

Pitches : E5 E5 D5 C5 C5 D5 D5 E5 D5 C5 G5 G5 F5 E5 E5 D5 C5 D5 E5 C5
Durations: Q E E Q Q Q E E Q Q E E Q Q E E E E H

Spoiler Alert: The old gray goose is dead.
Markov Modeling

- “Go Tell Aunt Rhody”:

```
<table>
<thead>
<tr>
<th>Pitches</th>
<th>E5</th>
<th>E5</th>
<th>D5</th>
<th>C5</th>
<th>D5</th>
<th>E5</th>
<th>D5</th>
<th>C5</th>
<th>G5</th>
<th>G5</th>
<th>F5</th>
<th>E5</th>
<th>E5</th>
<th>D5</th>
<th>C5</th>
<th>D5</th>
<th>E5</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Durations</td>
<td>Q</td>
<td>E</td>
<td>E</td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
<td>E</td>
<td>E</td>
<td>Q</td>
<td>Q</td>
<td>E</td>
<td>E</td>
<td>Q</td>
<td>Q</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
</tbody>
</table>
```

- C5 is followed by C5 D5 G5 D5
- D5 is followed by C5 D5 E5 C5 C5 E5
- E5 is followed by E5 D5 D5 E5 D5 C5
- F5 is followed by E5
- G5 is followed by G5 F5
Markov Modeling

“Go Tell Aunt Rhody”:

```
Pitches : E5 E5 D5 C5 C5 D5 D5 E5 D5 C5 G5 G5 F5 E5 E5 D5 C5 D5 E5 C5
Durations: Q E E Q Q Q Q E E Q Q E E Q Q E E E E H
```

*C5* is followed by *C5 D5 G5 D5*

*D5* is followed by *C5 D5 E5 C5 C5 E5*

*E5* is followed by *E5 D5 D5 E5 D5 C5*

*F5* is followed by *E5*

*G5* is followed by *G5 F5*

**A First-Order Markov Model:**
The next state is a random variable based entirely on the previous state.

\[
\text{prob}(C5 \rightarrow C5) = 1/3 \\
\text{prob}(C5 \rightarrow D5) = 2/3
\]
Markov Modeling

“Go Tell Aunt Rhody”:

Pitches : E5 E5 D5 C5 C5 D5 D5 E5 D5 C5 G5 G5 F5 E5 E5 D5 C5 D5 E5 C5
Durations: Q E E Q Q Q Q E E Q Q E E Q Q E E E E H

QQ is followed by Q Q E E E
QE is followed by E E E E
EQ is followed by Q Q Q
EE is followed by Q Q Q E E H
EH is followed by ?

A Second-Order Markov Model:
The next state is a random variable based entirely on the previous two states.

\[ \text{prob}(QQ \rightarrow Q) = \frac{2}{5} \]
\[ \text{prob}(QQ \rightarrow E) = \frac{3}{5} \]
Sample Code

```lang-sal
SAL> set rules = markov-create-rules(
    {e5 e5 d5 c5 c5 d5 d5 e5 d5 c5 g5 g5 f5 e5 e5 d5 c5 d5 e5 c5} ,1)

SAL> exec pprint(rules)

(((E5 -> (E5 2) (D5 3) (C5 1)))
(D5 -> (C5 3) (D5 1) (E5 2)))
(C5 -> (C5 1) (D5 2) (G5 1))
(G5 -> (G5 1) (F5 1))
(F5 -> (E5 1)))

"1" indicates a First-Order Markov Model
Sample Code

SAL> set iois = markov-create-rules({q i i q q q q i i q q i i q q i i i i h q},2)

SAL> exec pprint(iois)

(((Q I -> (I 4))
  (I I -> (Q 3) (I 2) (H 1))
  (I Q -> (Q 3))
  (Q Q -> (Q 2) (I 3))
  (I H -> (Q 1)))

Fictitious quarter note added to prevent dead-end from occurring.

“2” indicates a Second-Order Markov Model
Sample Code - Alternative

SAL> set iois = markov-create-rules(
    {q i i q q q i i q q i i q q i i i i h},
    2, #t)

SAL> exec pprint(iois)

((Q I -> (I 4))
 (I I -> (Q 3) (I 2) (H 1))
 (I Q -> (Q 3))
 (Q Q -> (Q 2) (I 3))
 (* * -> (H 1) (Q 8) (I 9))))

Fictitious quarter note added to prevent dead-end from occurring.

This optional parameter says to “generalize” to make sure the Markov Chain has no dead-end states.

“2” indicates a Second-Order Markov Model

A general “wild card” rule is added, which picks a pitch at random according to frequency of occurrence.
begin
with
pitch-pat  = make-markov(rules, past: {C5}, produces: :eval),
rhythm-pat = make-markov(iois, past: {Q Q}, produces: :eval)
exec score-gen(save: quote(markov-rhody),
    score-len: 22,
    pitch: next(pitch-pat),
    ioi: 0.5 * next(rhythm-pat),
    vel: 100)
end
exec pprint(markov-rhody)

Start with a C5 pitch
And two quarter notes
Sample Code - Putting it together

begin
  with
    pitch-pat = make-markov(rules, past: {C5}, produces: :eval),
    rhythm-pat = make-markov(iois, past: {Q Q}, produces: :eval)
  exec score-gen(save: quote(markov-rhody),
                  score-len: 22,
                  pitch: next(pitch-pat),
                  ioi: 0.5 * next(rhythm-pat),
                  vel: 100)
  end

  exec pprint(markov-rhody)
exec score-play(markov-rhody)

Start with a C5 pitch
And two quarter notes
This is the Markov model we created

Begin with a C5 pitch and two quarter notes. This is the Markov model we created. Now, generate the score:
Sample Code - Putting it together

((0 0 (SCORE-BEGIN-END 0 10))
 (0 0.5 (NOTE :VEL 100 :PITCH 74))
 (0.5 0.25 (NOTE :VEL 100 :PITCH 76))
 (0.75 0.25 (NOTE :VEL 100 :PITCH 74))
 (1 0.5 (NOTE :VEL 100 :PITCH 72))
 (1.5 0.5 (NOTE :VEL 100 :PITCH 74))
 (2 0.25 (NOTE :VEL 100 :PITCH 72))
 (2.25 0.25 (NOTE :VEL 100 :PITCH 74))
 (2.5 1 (NOTE :VEL 100 :PITCH 72))
 (3.5 0.5 (NOTE :VEL 100 :PITCH 74))
 (4 0.5 (NOTE :VEL 100 :PITCH 72))
 (4.5 0.5 (NOTE :VEL 100 :PITCH 72))
 (5 0.5 (NOTE :VEL 100 :PITCH 74))
 (5.5 0.25 (NOTE :VEL 100 :PITCH 74))
 (5.75 0.25 (NOTE :VEL 100 :PITCH 72))
 (6 0.25 (NOTE :VEL 100 :PITCH 72))
 (6.25 0.5 (NOTE :VEL 100 :PITCH 72))
 (6.75 0.5 (NOTE :VEL 100 :PITCH 74))
 (7.25 0.25 (NOTE :VEL 100 :PITCH 76))
 (7.5 0.25 (NOTE :VEL 100 :PITCH 76))
 (7.75 1 (NOTE :VEL 100 :PITCH 74))
 (8.75 0.25 (NOTE :VEL 100 :PITCH 76))
 (9 1 (NOTE :VEL 100 :PITCH 76)))

exec score-play(markov-rhody)