```
readXML.py xmlGenera...
                   ~/xmlMusicGen — -zsh
                                                                                     matrixMusi... requiremen... refactored...
                                                                                                                                       mxlConvert... musicGen.py
                                                                                                                                                                   log.txt
                                                                                                                                                                                         forgetTest....
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import xmlGenerate
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import random
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import numpy as np
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import fluidsynth
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import time
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  matrix = xmlGenerate.getMatrix()
Now playing: ('A4',)
                                                                  chord_list = xmlGenerate.getChordList()
Now playing: ('A4',)
Now playing: ('A4',)
                                                                  chord_index = xmlGenerate.getChordIndex()
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  # MATRIX MANIPULATION
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  # Higher = more entropy, Lower = less changes
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  # Emphasizes/Minimizes the existing row probability vectors
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  def scale_temperature(matrix, temperature=1.0):
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      assert temperature > 0, "Temperature must be positive"
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      log_matrix = np.log(matrix + 1e-9) # Avoid log(0)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled = np.exp(log_matrix / temperature)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled = np.maximum(scaled, 0)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled /= scaled.sum(axis=1, keepdims=True)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      return scaled
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                  def inject_noise(matrix, epsilon=0.01):
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      noisy = matrix + epsilon * np.random.rand(*matrix.shape)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      noisy /= noisy.sum(axis=1, keepdims=True) # Renormalize rows
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      return noisy
Now playing: ('G4',)
Now playing: ('G4',)
Now playing: ('A3', 'A4', 'C4*', 'D3', 'F3', 'F4')
Now playing: ('B3', 'D4', 'F3', 'G2%, 'G4')
Now playing: ('B3', 'D4', 'G4')
Now playing: ('B3', 'D4', 'G1', 'G4')
                                                                  matrix = scale_temperature(matrix, 0.5)
Now playing: ('B3', 'D4', 'G1', 'G4')
                                                                  # matrix = inject_noise(matrix, 0.0001)
Now playing: ('B3', 'D4', 'G1', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C4', 'E4', 'G4')
                                                                                 ---- TRAJECTORY THROUGH THE ROW STOCHASTIC MATRIX --
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                  initial = random.choice(chord_list)
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                  generated = [initial]
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                  for _ in range(1000):
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                      i = chord_index[initial]
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                      probs = matrix[i]
^CPlayback interrupted by user (Ctrl+C).
                                                                      j = np.random.choice(len(chord_list), p=probs)
Fluidsynth resources cleaned up.
(fooMUSIC) ethansie@Ethans-MacBook-Pro xmlMusicGen %
                                                            matrixMusic.py 29:37
                                                                                                                                        LF UTF-8 Python & main S Fetch G GitHub 🗢 Git (1) 🌣
```



```
readXML.py xmlGenera...
                   ~/xmlMusicGen — -zsh
                                                                                     matrixMusi... requiremen... refactored...
                                                                                                                                       mxlConvert... musicGen.py
                                                                                                                                                                   log.txt
                                                                                                                                                                                         forgetTest....
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import xmlGenerate
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import random
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import numpy as np
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import fluidsynth
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  import time
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                  matrix = xmlGenerate.getMatrix()
Now playing: ('A4',)
                                                                  chord_list = xmlGenerate.getChordList()
Now playing: ('A4',)
Now playing: ('A4',)
                                                                  chord_index = xmlGenerate.getChordIndex()
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  # MATRIX MANIPULATION
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  # Higher = more entropy, Lower = less changes
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  # Emphasizes/Minimizes the existing row probability vectors
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                  def scale_temperature(matrix, temperature=1.0):
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      assert temperature > 0, "Temperature must be positive"
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      log_matrix = np.log(matrix + 1e-9) # Avoid log(0)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled = np.exp(log_matrix / temperature)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled = np.maximum(scaled, 0)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled /= scaled.sum(axis=1, keepdims=True)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      return scaled
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                  def inject_noise(matrix, epsilon=0.01):
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      noisy = matrix + epsilon * np.random.rand(*matrix.shape)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      noisy /= noisy.sum(axis=1, keepdims=True) # Renormalize rows
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      return noisy
Now playing: ('G4',)
Now playing: ('G4',)
Now playing: ('A3', 'A4', 'C4*', 'D3', 'F3', 'F4')
Now playing: ('B3', 'D4', 'F3', 'G2%, 'G4')
Now playing: ('B3', 'D4', 'G4')
Now playing: ('B3', 'D4', 'G1', 'G4')
                                                                  matrix = scale_temperature(matrix, 0.5)
Now playing: ('B3', 'D4', 'G1', 'G4')
                                                                  # matrix = inject_noise(matrix, 0.0001)
Now playing: ('B3', 'D4', 'G1', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C4', 'E4', 'G4')
                                                                                 ---- TRAJECTORY THROUGH THE ROW STOCHASTIC MATRIX --
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                  initial = random.choice(chord_list)
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                  generated = [initial]
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                  for _ in range(1000):
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                      i = chord_index[initial]
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                      probs = matrix[i]
^CPlayback interrupted by user (Ctrl+C).
                                                                      j = np.random.choice(len(chord_list), p=probs)
Fluidsynth resources cleaned up.
(fooMUSIC) ethansie@Ethans-MacBook-Pro xmlMusicGen %
                                                            matrixMusic.py 29:37
                                                                                                                                        LF UTF-8 Python & main S Fetch G GitHub 🗢 Git (1) 🌣
```

```
readXML.py xmlGenera...
                   ~/xmlMusicGen — -zsh
                                                                                    matrixMusi... requiremen... refactored...
                                                                                                                                     mxlConvert... musicGen.py
                                                                                                                                                                 log.txt
                                                                                                                                                                                       forgetTest....
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                 import xmlGenerate
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                 import random
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                 import numpy as np
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                 import fluidsynth
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                 import time
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
Now playing: ('A3', 'A4', 'B2', 'B4', 'E4b')
                                                                 matrix = xmlGenerate.getMatrix()
Now playing: ('A4',)
                                                                 chord_list = xmlGenerate.getChordList()
Now playing: ('A4',)
Now playing: ('A4',)
                                                                 chord_index = xmlGenerate.getChordIndex()
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                 # MATRIX MANIPULATION
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                 # Higher = more entropy, Lower = less changes
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                 # Emphasizes/Minimizes the existing row probability vectors
Now playing: ('B3', 'B4', 'D4', 'E3', 'G3', 'G4')
                                                                 def scale_temperature(matrix, temperature=1.0):
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      assert temperature > 0, "Temperature must be positive"
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      log_matrix = np.log(matrix + 1e-9) # Avoid log(0)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled = np.exp(log_matrix / temperature)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled = np.maximum(scaled, 0)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      scaled /= scaled.sum(axis=1, keepdims=True)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      return scaled
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                 def inject_noise(matrix, epsilon=0.01):
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      noisy = matrix + epsilon * np.random.rand(*matrix.shape)
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      noisy /= noisy.sum(axis=1, keepdims=True) # Renormalize rows
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
Now playing: ('A2', 'A4', 'C4#', 'E4', 'G3')
                                                                      return noisy
Now playing: ('G4',)
Now playing: ('G4',)
Now playing: ('A3', 'A4', 'C4*', 'D3', 'F3', 'F4')
Now playing: ('B3', 'D4', 'F3', 'G2%, 'G4')
Now playing: ('B3', 'D4', 'G4')
Now playing: ('B3', 'D4', 'G1', 'G4')
                                                                 matrix = scale_temperature(matrix, 0.5)
Now playing: ('B3', 'D4', 'G1', 'G4')
                                                                 # matrix = inject_noise(matrix, 0.0001)
Now playing: ('B3', 'D4', 'G1', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C4', 'E4', 'G4')
                                                                                ---- TRAJECTORY THROUGH THE ROW STOCHASTIC MATRIX --
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                 initial = random.choice(chord_list)
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                 generated = [initial]
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                                                                                                                  Moon River
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                 for _ in range(1000):
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                      i = chord_index[initial]
                                                                                                                                                                    (low temp)
Now playing: ('C2', 'C4', 'E4', 'G4')
Now playing: ('C2', 'C4', 'E4', 'G4')
                                                                      probs = matrix[i]
^CPlayback interrupted by user (Ctrl+C).
                                                                      j = np.random.choice(len(chord_list), p=probs)
Fluidsynth resources cleaned up.
(fooMUSIC) ethansie@Ethans-MacBook-Pro xmlMusicGen %
                                                                                                                                      LF UTF-8 Python & main Setch (7) GitHub 🗢 Git (1) 🌣
                                                            matrixMusic.py 29:37
```

```
~/xmlMusicGen — -zsh
                                                          readXML.py xmlGenera...
                                                                                 matrixMusi... requiremen... refactored...
                                                                                                                                                                                forgetTest...
                                                                                                                                mxlConvert... musicGen.py
                                                                                                                                                           log.txt
Now playing: ('B2', 'D4')
                                                               import xmlGenerate
Now playing: ('B2', 'D4', 'G2')
                                                               import random
Now playing: ('D4', 'G2')
Now playing: ('E2', 'E4')
                                                               import numpy as np
Now playing: ('E4', 'G2')
                                                               import fluidsynth
Now playing: ('E4', 'G2')
Now playing: ('A4', 'C5', 'F2', 'F4')
                                                               Import time
Now playing: ('A2', 'A4', 'C5', 'F4')
Now playing: ('A4', 'B2', 'C5')
Now playing: ('D3', 'F4')
                                                               matrix = xmlGenerate.getMatrix()
Now playing: ('C5', 'D3', 'E3', 'E5', 'F4')
                                                              chord_list = xmlGenerate.getChordList()
Now playing: ('C5', 'E3', 'E5')
Now playing: ('C5', 'E3', 'E5')
                                                               chord_index = xmlGenerate.getChardIndex()
Now playing: ('C5', 'E3', 'E5')
Now playing: ('B4', 'D3', 'D5')
Now playing: ('B4', 'D3', 'D5')
                                                               # MATRIX MANIPULATION
Now playing: ('B4', 'D3', 'D5')
                                                               # Higher = more entropy, Lower = lass channel
Now playing: ('B4', 'D3', 'D5')
Now playing: ('A4', 'C3', 'C5')
                                                               # Emphasizes/Ninimizes the existing row probability vertors
Now playing: ('A4', 'B2', 'B4', 'C3', 'C5', 'G4')
                                                               def scale_temperature(matrix, temperature=1.0);
Now playing: ('B2', 'B4', 'G4')
Now playing: ('B2', 'B4', 'G4')
                                                                   assert temperature > 0, "Temperature must be positive"
Now playing: ('A3', 'B3', 'D4', 'G3')
                                                                   Now playing: ('B3', 'D4', 'F3', 'G3')
Now playing: ('B3', 'D4', 'F3', 'G3')
                                                                   scaled = np.exp(log_matrix / temperature)
Now playing: ('A3', 'B3', 'D4', 'G3')
                                                                   scaled = np.maximum(scaled, 0)
Now playing: ('A3', 'B3', 'D4', 'G3')
Now playing: ('A3', 'B3', 'D4', 'G3')
                                                                   scaled /= scaled.sum(axis=1, keepdims=True)
Now playing: ('A3', 'B3', 'D4', 'G3')
                                                                   return scaled
Now playing: ('B3', 'D4', 'F3', 'G3')
Now playing: ('B3', 'D4', 'F3', 'G3')
Now playing: ('B3', 'D4', 'F3', 'G3')
                                                              def inject_noise(matrix, epsilon=0.01):
Now playing: ('B3', 'D4', 'F3', 'G3')
Now playing: ('B3', 'D3', 'D4', 'F3', 'G3')
                                                                   noisy = matrix + epsilon * np.random.rand(*matrix.shape)
Now playing: ('B3', 'D3', 'D4', 'G3')
                                                                   Now playing: ('B3', 'D3', 'D4', 'G3')
Now playing: ('B3', 'D3', 'D4', 'F3', 'G3')
                                                                   return noisy
Now playing: ('B3', 'D4', 'F3', 'G3')
Now playing: ('B3', 'D4', 'F3', 'G3')
^CPlayback interrupted by user (Ctrl+C).
Fluidsynth resources cleaned up.
(fooMUSIC) ethansie@Ethans-MacBook-Pro xmlMusicGen % python
3 matrixMusic.py extracted_mxl/moon.xml
                                                              matrix = scale_temperature(matrix, 2.5)
First measure, grid of notes:
                                                          # matrix = in/ecc onise(matrix, 0,000)
Now playing: ('C3', 'E5')
Now playing: ('E3', 'E5')
Now playing: ('E3', 'E5')
Now playing: ('E5', 'G3')
Now playing: ('D3', 'D5', 'G3')
                                                                                   TRAJECTORY THROUGH THE ROW STOCHASTIC MATRIX
Now playing: ('D3', 'D5')
                                                               initial = random.choice(chord_list)
Now playing: ('D3', 'D5')
Now playing: ('A4', 'C5', 'F2', 'F4')
                                                               generated = [initial]
Now playing: ('A4', 'C3', 'C5', 'F4')
                                                                                                                                                           Moon River
Now playing: ('A4', 'C3', 'C5', 'F4')
Now playing: ('A2', 'A4', 'C3', 'C5', 'F4')
                                                               for _ in range(1000):
Now playing: ('A4', 'C3', 'C5', 'F4')
                                                                   i = chord_index[initial]
                                                                                                                                                            (high temp)
Now playing: ('A4', 'C3', 'C5', 'F3', 'F4')
Now playing: ('A4', 'C3', 'C5', 'F4')
                                                                   probs = matrix[i]
^CPlayback interrupted by user (Ctrl+C).
                                                                   j = np.random.choice(len(chord_list), p=probs)
Fluidsynth resources cleaned up.
(fooMUSIC) ethansie@Ethans-MacBook-Pro xmlMusicGen %
                                                                                                                                 LF UTF-9 Python F main C Fetch (7) GitHub C Git (1)
                                                         matrixMusic.pv 107:45
```