Customizable Audio Kaleidoscope

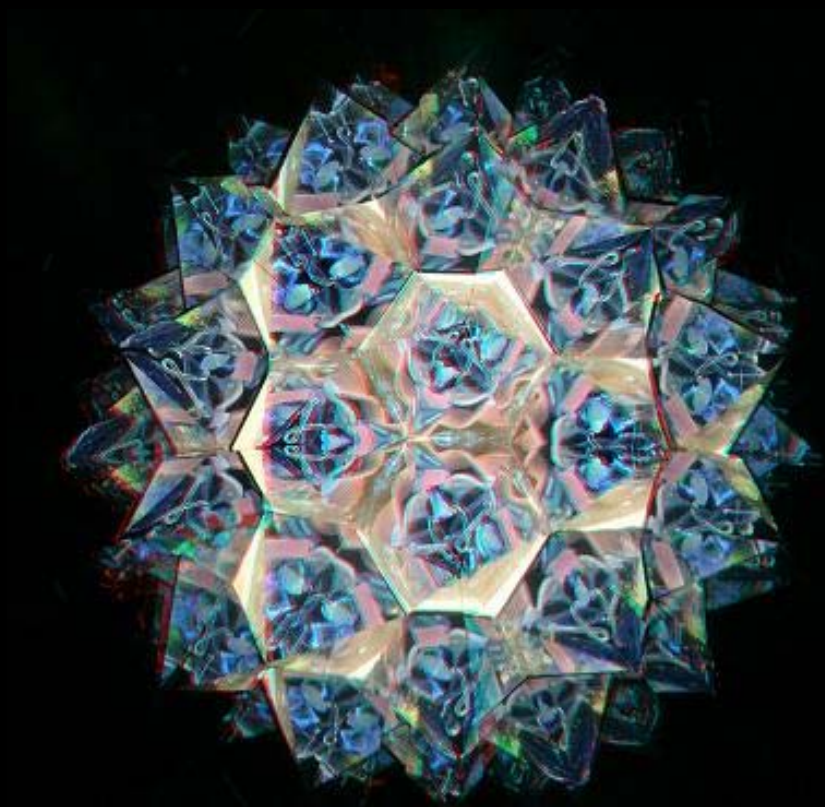
Agustya Mehta, Dennis Ramdass, Tony Hwang

6.111 Final Project

Spring 2007

Kaleidoscopes

- Produce changing, pleasing images through simple user interface
- How can we mimic (and improve upon) this idea, but with sound?



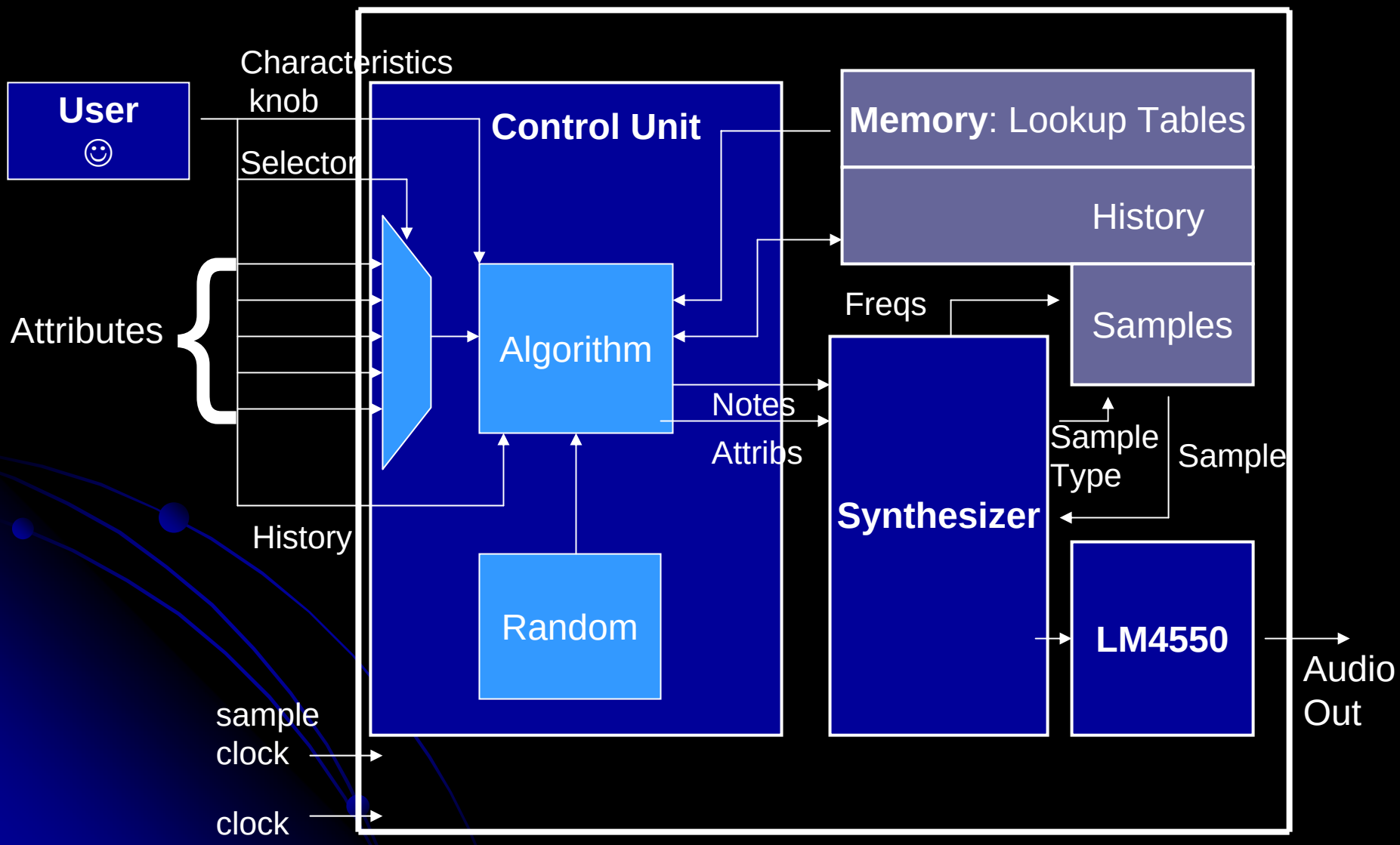
Goals

- Creating pleasing music with a certain feel conveniently
- Abstracting away technical details for the user's convenience. No musical background necessary

Audio on the fly…

- The audio system contains:
  - Convenient user features
  - A configurable algorithm for generating music
  - A versatile digital synthesizer

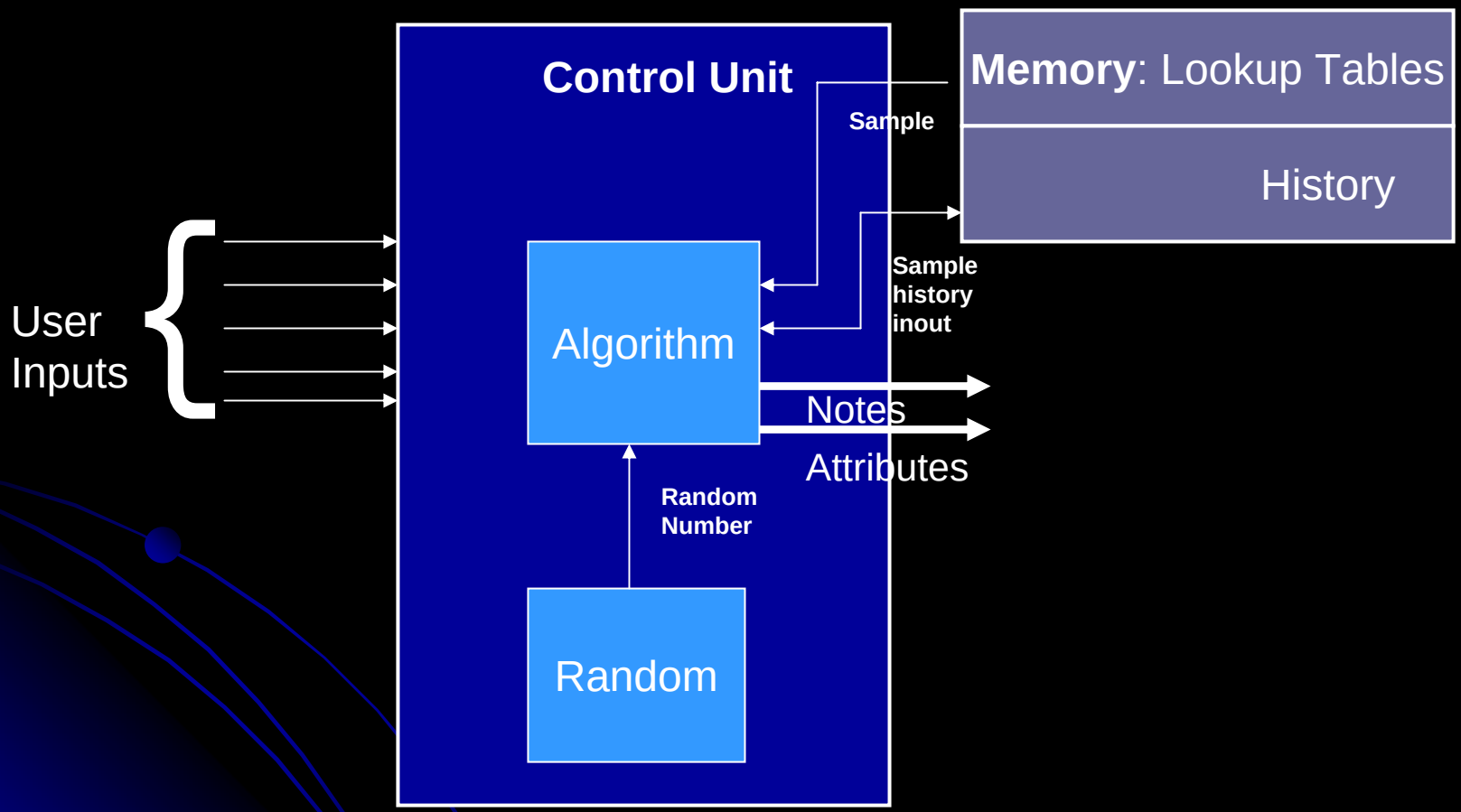
System Overview



User-adjustable Features

- Musical characteristics
  - Key, tempo, timbre, dynamics, note duration.
- Correlation knob and attribute selector
- Duration of musical history

Control Unit Block Diagram



Control Unit

- The Control Unit consists of four modules:

1. Static Memory Lookup Table
2. Memory History
3. Random Generator
4. Scoring Algorithm

Control Unit

- **Static Memory Lookup Table:**

  Restricted selection of audio samples programmed into RAM
- **Memory History:**

  Stores copies of the last sequence of audio samples played by the system
- **Random Generator:**

  Generates a random number

Control Unit

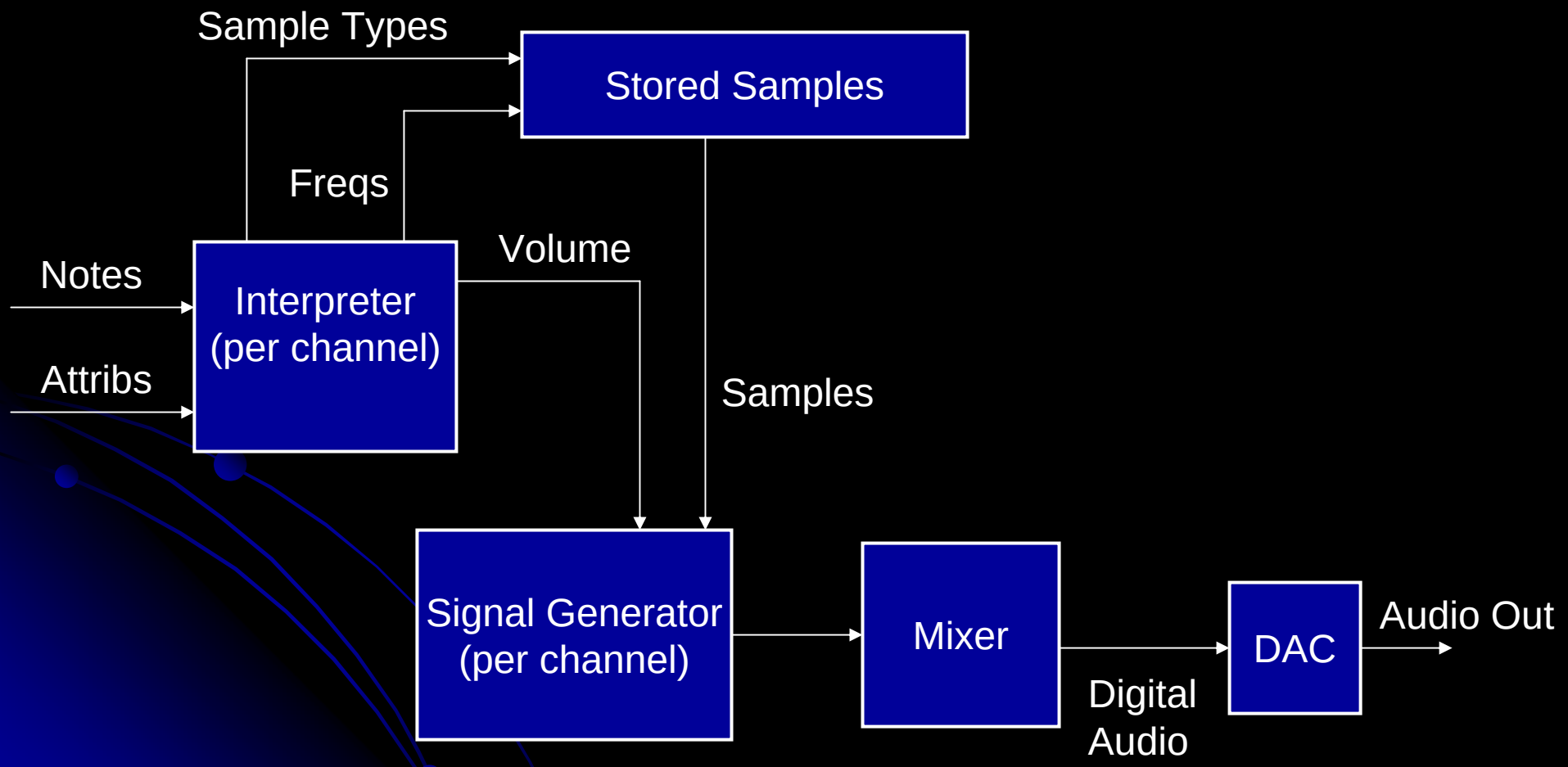
Scoring Algorithm:

- Assigns a score to each sample in the Static Memory Lookup Table
- Score calculation considers user inputs and data stored in the Memory History module
- Scoring criteria based on musical knowledge e.g. knowing which progressions sound better
- Score determines the probability of a sample being chosen as output sample
- Random number used in choosing output sample

Control Unit

- Outputs:
  1. The set of notes in the chosen sample
  2. A set of audio attributes calculated based on user inputs

Synthesizer



Interpreter

- Algorithm outputs logical values for notes and attributes
- Must convert this to audio data
  - Note values (e.g., C, G#) correspond to frequencies, with A4 = 440 Hz, and these frequencies are sent to sample memory so that stored samples can be scaled
  - Volume data is sent directly to signal generator
  - Timbres (instrument types) correspond to samples stored in memory; this attribute is used to look up sample type
  - Handle timing by outputting this data to the signal generator module for the duration of the note, and outputting null value when no note is playing

Signal Generator

- Takes in frequency-scaled samples from memory that were selected by interpreter along with volume data
- Scales the samples to the desired volume
- Separate instantiation of interpreter and signal generator for each audio channel; all data sent to mixer

Mixer

- Takes in waveform signals for each channel outputted from signal generator
- Sums these signals together
- Sends digital audio output to DAC

In summary…

- An intelligent audio synthesizer that adjusts the feel of the music in real-time based on the user's desires
- Applications: versatile

Questions?