**Class 29: Vectors**

**Held:** Friday, 13 October 2006

**Summary:** Today we consider vectors, which serve as alternates to lists. Vectors differ from lists in three primary ways: vectors have fixed size, vectors are indexed, and vectors are mutable.

**Related Pages:**
- EBoard.
- Lab: Vectors.
- Reading: Vectors.

**Due**
- Project 1 Writeup.

**Notes:**
- Be careful: Friday the 13th falls on a Friday today.

**Overview:**
- Problems with lists.
- A solution: Vectors.
- Lab.

**List Deficiencies**
- Now that we’ve worked with lists for a while, we’ve identified some things that make lists inappropriate for some situations.
  - Lists are *expensive* to use; to find the length of a list or to access an element late in the list, you need to cdr down the list.
  - Lists are *fixed*; you can’t easily change an element of a list.
- At the same time, there are some advantages to lists:
  - Lists are *dynamic*; it is easy to grow and shrink a list.
  - Lists are *inherently recursive*; the type is defined recursively.
  - Lists are *simple*; you can build almost every list operation through just a few basic operations (*car*, *cdr*, *null*, and *null?*).
An Alternative: Vectors

- Vectors provide an alternative to lists.
- They have two primary advantages:
  - Vectors are indexed: You can quickly access elements by number.
  - Vectors are mutable: You can change the elements of a vector.
- In order to obtain these benefits, vectors lack some key features of lists. In particular,
  - Vectors are static: Once you’ve created a vector, you cannot change its length.
- Some key vector procedures:
  - `(vector val1 ... valn)`: Create a vector
  - `(make-vector length val)`: Make a vector of specified length, with duplicates of `val` as the contents.
  - `(vector-ref vector position)`: Extract a value from a vector.
  - `(vector-set! vector position newvalue)`: Change an element of a vector.

Lab

- Do the lab.