Turing's machine model

Machine models of a computer always have

- means to store data
  - e.g objects in Java, files and variables in UNIX
- means to manipulate data
  - e.g. methods in Java, processes in UNIX
- ways to program data manipulation
  - e.g. JAVA programs, UNIX shell scripts

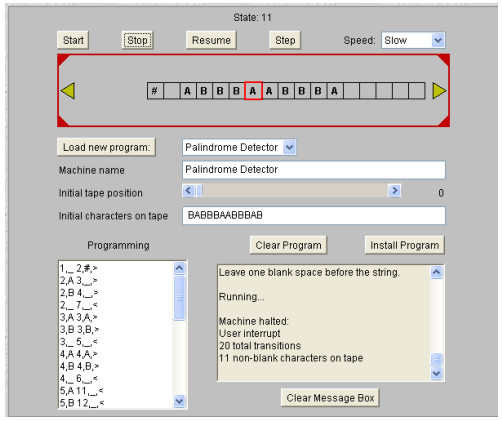
The Turing Machine model (1936)

- store is represented by an unbounded tape
- processor is represented by a read/write head
- program is represented by a set of rules

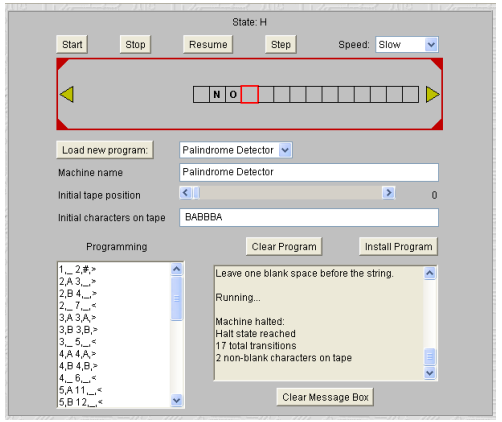
Suzanne Skinner (1996) Java applet simulator at:

http://ironphoenix.org/tril/tm/

A Turing computation in progress … recognising a palindromic string



A Turing computation in the halt state … rejecting a non-palindromic string



The Church-Turing thesis

There is no computational model that is in principle more powerful than the Turing machine …

… all algorithmic data processing is equivalent to Turing computation

… by this criterion, very simple notations can define "a full programming language"

Procedural version of isprime

```
func factors {
    para n;
    auto r, result;
    result = [];
    for (r=1; r<=n/2; r++)
        if (n % r == 0) result = result // [r];
    return result;
}

func isprime {
    para n;
    return ((factors[n])# == 1);
}
```


Functional programming (FP)

A "functional" program to compute prime numbers:

```
factors n = [r | r<-[1..n div 2]; n mod r = 0]
isprime q = (# factors q) = 1
```

functional ≡ based on specifying functions
The functions in this context are
factors() and *isprime()*

The programming language is **Miranda**

Procedural version of isprime

```
func factors {                    factors n = [r | r<-[1..n div 2]; n mod r = 0]
    para n;
    auto r, result;
    result = [];
    for (r=1; r<=n/2; r++)
        if (n % r == 0) result = result // [r];
    return result;
}

func isprime {                    isprime q = (# factors q) = 1
    para n;
    return ((factors[n])# == 1);
}
```