Homosapien Modeling Language (HML)

Language Reference Manual

COMS W4115: Programming Languages and Translators

Professor Stephen A. Edwards

Computer Science Department

Summer 2008 Columbia University

Dated: 2008-06-19

Derek Ng

dn2150@columbia.edu

Columbia Video Network

Introduction

The Homosapien Modeling Language (HML) is a language meant to allow programmers the ability to model the nature of human beings. With HML, a programmer could model the effects of external forces on the human body. Some of HML's syntax are similar to the C programming language syntax. The basis for the language involves a Human data type which can be manipulated by a Force data type. In real life, there are numerous external forces in the world and this language hopefully will help us to model how Humans will react.

Lexical conventions

Like the C programming language, HML will have the following tokens: identifiers, keywords, operators, and separators. In this initial version of HML, there will be a limited set of supported tokens as there is a limited of resources available (just me!).

Separators

Tokens are separated by blanks, tabs, and newlines.

Comments

Comments are preceded and followed by a control sequence. Comments are started with a forward, double slash and are ended with a backward double slash. An example comment would be as follows:

```
// THIS IS A COMMENT \\
```

Identifiers

A HML identifier is a string that begins with a letter and is then followed by any pattern of letters or numbers. Example identifiers would be: a, a00, a00a00, and so forth. Identifiers cannot be *a (starting with an asterisk) or 00a (starting with a number).

Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

if
else
while
do
Human
Force
Display
TRUE
FALSE

Constants

Current HML only has integer and Boolean constants. An integer constant is any sequence of digits from 0 to 9, but if the sequence begins with 0, it may only be of length 1. A Boolean constant may only be True or False.

Built in Operators

HML has a set of built in operators to manipulate variables.

- Assignment (=) is used to define a variable to take on a given value denoted by variable = value.
- Addition (+) is used to apply a Force to a Human. If A is a Force and B is a Human, than B + A would apply the Force to the Human. This operation returns a Human variable.
- Is Equal To (==) is used to compare the equality of two variables. It returns a Boolean value.
- Is Greater Than (>), Is Less Than (<), and Is Greater Than or Equal To (>=) are similar to the Is Equal To function but compare two variables in accordance with the operation's name. They too return Boolean values.

Assignment has the lowest priority of any of the built in operators. All other operators are of equal priority and are evaluated from left to right.

Control Statements

Most statements are expression statements:

expression

Expressions can be used for HML's has two forms of control statements. The first control statement is the if-else statement which takes the form:

if (*expression*) {*statement*} else {*statement*}

Like in the C programming language, if the first statement is valid, then the second statement executes. If the first statement is false, then the statement after the else executes.

The second control statement is a while-do loop which allows for repetitive operations while a given control statement holds.

while (<*expression*>) do { <*statement*> }

Scope

At this time, the role of scope in HML is TBD.

Token replacement

HML does not support token replacement.

File inclusion

HML does not support file inclusion and is limited to the syntax defined in this document.

Sample Code

foo.hml

```
Human Derek
{
  Height 400
  Age 10
  Weight 410
  Happiness
}
Force Burger.Weight 10
Derek += Burger
Display Derek
```