Project Proposal - PLT

Manu Jain (mj2517)

Language I plan to Implement

The name of the language is DINO.

DINO is a fun, interactive, interpreted language that provides feedback to the user. It is very vaguely inspired by MOO, and the "Future Development" section takes some inspiration from LOGO.

The Problem that DINO Language can Solve

How to get young kids interested in programming?

This language is aimed at introducing young kids (ages 8 to 12) to programming, getting them interested in programming, teaching them the basics of programming, and making programming fun for them.

How It Should be Used

The basic idea is to give the young programmers some limited amount of choices and operations, with which they build up a little dinosaur world for themselves. There shouldn't be too many choices so that things are kept simple.

Therefore the language will provide one base type called called *dinosaur*. The *dinosaur* will have properties (e.g. name, height, length, position, family, diet-type, etc.). The behavior will come through keywords. Advanced users will be able to define methods.

Another type, the *thing* type, is provided so programmers can create other 'stuff' that interests them like plants, trees, rocks, buildings, etc. Anything derived from *thing* may have any number of properties, but no methods.

The language supports type-inheritance.

The properties of any type can be defined and assigned default values during the time of type-definition (types will have default values for each property). Methods may be defined during the time of type-definition.

Since the behavior comes only from keywords or methods on the type dinosaur, the language will check to make sure that only dinosaurs have behavior (e.g.

"rock eat dinosaur" will not be allowed by the language. Since it's a dinosaur world, only dinosaurs will have behavior. Sorry, the trees will not be able to 'grow'!).

Another example of 'smart' keywords could be for the 'attack' keyword to check the relative sizes of the dinosaurs, the distance between the two dinosaurs and the diet-type of the dinosaurs to determine whether an attack can be mounted and how successful an attack would be.

This approach does have its limitations, but it should suffice for this language's intended purpose and target users.

Syntax

Properties will be written with a dot following an object (e.g. dinosaur. height).

Methods will be written similar to the keywords, with a space or spaces following the object.

Full stop will indicate end of a statement.

The language keywords will be chosen such that they are English words that make sense for type creation and program execution, or mimic dinosaur behavior. For example –

Examples of Keywords for Type Creation and Program Execution

define	=> used to create new dinosaur types
define…as	=> used to create new types derived from other types.
is	=> equivalent to assignment operator
create…named	=> equivalent to 'new', but assigns a unique identifier
create	=> similar to the 'create', but gives a generated id
zap	=> equivalent to delete operator in C++

Examples of Keywords for Dinosaur Behavior

eat	=> allows dinosaur to eat a thing or another dinosaur completely.
eatfrom	=> allows dinosaur to eat part of a thing
sleep	=> puts the dinosaur to sleep
wakeup	=> wakes up the dinosaur
walk	=> moves the dinosaur slowly
run	=> moves the dinosaur quickly
attack	=> allows dinosaur to attack another dinosaur
hide	=> attempts to hide the dinosaur

There will be two main steps involved in creating a DINO program:

Type Definition (Design Time)

This is where the programmer would define types and their properties, and assign default values to properties. This is also where the user will create methods.

By default, any type being defined will be a dinosaur, unless specified to be a *thing* or a type derived from dinosaur.

Main Program

The main command window will allow programmers to bring their dinosaurs to life, and command execution will result in appropriate feedback.

The programmer may assign values to each instance's properties, which would override the default values for that type.

Future Development (not for this class!)

The future plan for this language would be to develop a graphical user interface (aka IDE), where young programmers can define types graphically and then write commands and watch their objects come to life on a canvas.

Outside the command window, the programmers would be able to create a user-interface for the objects. The *thing* and *dinosaur* types will be enhanced so that they have a 'view' property associated with them that represents their physical form. The 'view' property would follow inheritance rules.

Easy to use tools may be provided for the programmer to build up the 'view' of each object – for example, for a dinosaur they would be able to create a body (e.g. ready-made collections of faces, limbs, bodies, tails, etc. that snap to each other to create a body).

Property creation and editing will happen through a property editor. Methods may be defined in a text editor.

The above steps would create the types and assign them properties.

The actual 'main' program would be executed in the command window. The command window would be an interpreter. As each command is typed, the programmer would be able to see the result of that command in a graphical manner on the canvas.

Intellisense can be developed to provide programmatic hints to the programmers.

Having a graphical interface could lower the entry barrier down to six years of age.

An Interesting, Representative Program in DINO Language

Type Definition

```
define t-rex.
>> defined new type t-rex derived from dinosaur.

t-rex.height is 50.
t-rex.length is 100.
t-rex.position is 10.
t-rex.diet-type is Carnivore.

define sauropod.
>> defined new type sauropod derived from dinosaur.

sauropod.height is 100.
sauropod.length is 150.
sauropod.position is 200.
sauropod.diet-type is Herbivore.

define tree as thing.
>> defined new type tree derived from thing.

tree.height is 30.
tree.position is 250.
```

Main Command Window

```
create sauropod named Jumbo.
>> created sauropod named Jumbo

Jumbo eatfrom tree.
>> Jumbo has eaten from the tree.

create t-rex named Godzilla.
>>created t-rex named Godzilla

Godzilla attack Jumbo.
>> Godzilla cannot attack Jumbo. They are too far apart.
```

```
Godzilla run 150.
>> Godzilla is now at 160.

Godzilla attack Jumbo.
>> Godzilla has attacked Jumbo and injured it.

zap Jumbo.
>> Jumbo is no longer with us. Peace to his soul.
```