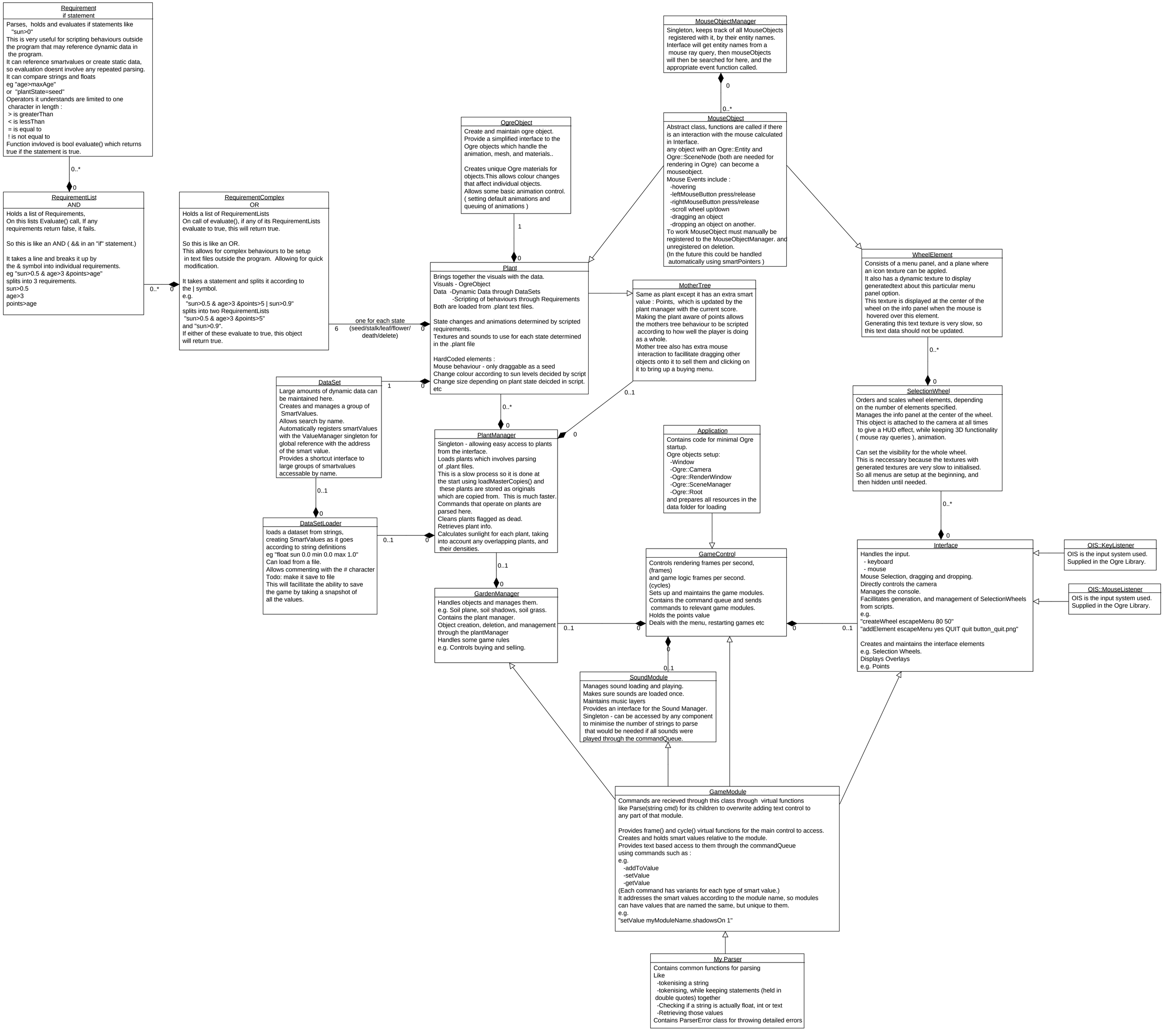
Requirement
if statement
Parses, holds and evaluates if statements like
"sun>0"
This is very useful for scripting behaviours outside
the program that may reference dynamic data in
the program.
It can reference smartvalues or create static data,
so evaluation doesnt involve any repeated parsing.
It can compare strings and floats
eg "age>maxAge"
or "plantState=seed"
Operators it understands are limited to one
character in length :
> is greaterThan
< is lessThan
= is equal to
! is not equal to
Function invloved is bool evaluate() which returns
true if the statement is true.
0..*
0
RequirementList
AND
Holds a list of Requirements,
On this lists Evaluate() call, If any
requirements return false, it fails.
So this is like an AND (&& in an "if" statement.)
It takes a line and breaks it up by
the & symbol into individual requirements.
eg "sun>0.5 & age>3 &points>age"
splits into 3 requirements.
sun>0.5
age>3
points>age
0..*
0
RequirementComplex
OR
Holds a list of RequirementLists
On call of evaluate(), if any of its RequirementLists
evaluate to true, this will return true.
So this is like an OR.
This allows for complex behaviours to be setup
in text files outside the program. Allowing for quick
modification.
It takes a statement and splits it according to
the | symbol.
e.g.
"sun>0.5 & age>3 &points>5 | sun>0.9"
splits into two RequirementLists
"sun>0.5 & age>3 &points>5"
and "sun>0.9".
If either of these evaluate to true, this object
will return true.
6
one for each state
(seed/stalk/leaf/flower/
death/delete)
0
OgreObject
Create and maintain ogre object.
Provide a simplified interface to the
Ogre objects which handle the
animation, mesh, and materials..
Creates unique Ogre materials for
objects.This allows colour changes
that affect individual objects.
Allows some basic animation control.
(setting default animations and
queuing of animations)
1
0
Plant
Brings together the visuals with the data.
Visuals - OgreObject
Data -Dynamic Data through DataSets
-Scripting of behaviours through Requirements
Both are loaded from .plant text files.
State changes and animations determined by scripted
requirements.
Textures and sounds to use for each state determined
in the .plant file
HardCoded elements :
Mouse behaviour - only draggable as a seed
Change colour according to sun levels decided by script
Change size depending on plant state deicded in script.
etc
MouseObjectManager
Singleton, keeps track of all MouseObjects
registered with it, by their entity names.
Interface will get entity names from a
mouse ray query, then mouseObjects
will then be searched for here, and the
appropriate event function called.
0
0..*
MouseObject
Abstract class, functions are called if there
is an interaction with the mouse calculated
in Interface.
any object with an Ogre::Entity and
Ogre::SceneNode (both are needed for
rendering in Ogre) can become a
mouseobject.
Mouse Events include :
-hovering
-leftMouseButton press/release
-rightMouseButton press/release
-scroll wheel up/down
-dragging an object
-dropping an object on another.
To work MouseObject must manually be
registered to the MouseObjectManager. and
unregistered on deletion.
(In the future this could be handled
automatically using smartPointers)
MotherTree
Same as plant except it has an extra smart
value : Points, which is updated by the
plant manager with the current score.
Making the plant aware of points allows
the mothers tree behaviour to be scripted
according to how well the player is doing
as a whole.
Mother tree also has extra mouse
interaction to facillitate dragging other
objects onto it to sell them and clicking on
it to bring up a buying menu.
WheelElement
Consists of a menu panel, and a plane where
an icon texture can be appled.
It also has a dynamic texture to display
generatedtext about this particular menu
panel option.
This texture is displayed at the center of the
wheel on the info panel when the mouse is
hovered over this element.
Generating this text texture is very slow, so
this text data should not be updated.
0..*
0
SelectionWheel
Orders and scales wheel elements, depending
on the number of elements specified.
Manages the info panel at the center of the wheel.
This object is attached to the camera at all times
to give a HUD effect, while keeping 3D functionality
(mouse ray queries), animation.
Can set the visibility for the whole wheel.
This is neccessary because the textures with
generated textures are very slow to initialised.
So all menus are setup at the beginning, and
then hidden until needed.
0..*
0
DataSet
Large amounts of dynamic data can
be maintained here.
Creates and manages a group of
SmartValues.
Allows search by name.
Automatically registers smartValues
with the ValueManager singleton for
global reference with the address
of the smart value.
Provides a shortcut interface to
large groups of smartvalues
accessable by name.
1
0
0..*
0
0..1
PlantManager
Singleton - allowing easy access to plants
from the interface.
Loads plants which involves parsing
of .plant files.
This is a slow process so it is done at
the start using loadMasterCopies() and
these plants are stored as originals
which are copied from. This is much faster.
Commands that operate on plants are
parsed here.
Cleans plants flagged as dead.
Retrieves plant info.
Calculates sunlight for each plant, taking
into account any overlapping plants, and
their densities.
0..1
0
0..1
0
Application
Contains code for minimal Ogre
startup.
Ogre objects setup:
-Window
-Ogre::Camera
-Ogre::RenderWindow
-Ogre::SceneManager
-Ogre::Root
and prepares all resources in the
data folder for loading
DataSetLoader
loads a dataset from strings,
creating SmartValues as it goes
according to string definitions
eg "float sun 0.0 min 0.0 max 1.0"
Can load from a file.
Allows commenting with the # character
Todo: make it save to file
This will facillitate the ability to save
the game by taking a snapshot of
all the values.
0..1
0
0..1
0
GardenManager
Handles objects and manages them.
e.g. Soil plane, soil shadows, soil grass.
Contains the plant manager.
Object creation, deletion, and management
through the plantManager
Handles some game rules
e.g. Controls buying and selling.
0..1
0
GameControl
Controls rendering frames per second,
(frames)
and game logic frames per second.
(cycles)
Sets up and maintains the game modules.
Contains the command queue and sends
commands to relevant game modules.
Holds the points value
Deals with the menu, restarting games etc
0
0..1
Interface
Handles the input.
- keyboard
- mouse
Mouse Selection, dragging and dropping.
Directly controls the camera
Manages the console.
Facillitates generation, and management of SelectionWheels
from scripts.
e.g.
"createWheel escapeMenu 80 50"
"addElement escapeMenu yes QUIT quit button_quit.png"
Creates and maintains the interface elements
e.g. Selection Wheels.
Displays Overlays
e.g. Points
OIS::KeyListener
OIS is the input system used.
Supplied in the Ogre Library.
OIS::MouseListener
OIS is the input system used.
Supplied in the Ogre Library.
SoundModule
Manages sound loading and playing.
Makes sure sounds are loaded once.
Maintains music layers
Provides an interface for the Sound Manager.
Singleton - can be accessed by any component
to minimise the number of strings to parse
that would be needed if all sounds were
played through the commandQueue.
GameModule
Commands are recieved through this class through virtual functions
like Parse(string cmd) for its children to overwrite adding text control to
any part of that module.
Provides frame() and cycle() virtual functions for the main control to access.
Creates and holds smart values relative to the module.
Provides text based access to them through the commandQueue
using commands such as :
e.g.
-addToValue
-setValue
-getValue
(Each command has variants for each type of smart value.)
It addresses the smart values according to the module name, so modules
can have values that are named the same, but unique to them.
e.g.
"setValue myModuleName.shadowsOn 1"
My Parser
Contains common functions for parsing
Like
-tokenising a string
-tokenising, while keeping statements (held in
double quotes) together
-Checking if a string is actually float, int or text
-Retrieving those values
Contains ParserError class for throwing detailed errors