The context
THE CONTEXT
THE CONTEXT
The context
The context
THE CONTEXT
Problem

- Use more memory than needed.
- Make OOP languages unsuitable for memory limited devices.
- Existence of unused but referenced objects.
In OOP primary memory is represented by an object graph
Garbage Collector

Only collects objects that nobody else points to.
Garbage Collector

Only collects objects that nobody else points to.
Garbage Collector

Only collects objects that nobody else points to.
Garbage Collector

Only collects objects that nobody else points to.
But...what happens with referenced yet unused objects?
Swap out (not remove) unused objects to disk.

Automatically load them back when needed.
**Related work**

- Large object oriented memory (LOOM).
- Melt - Supporting memory leaks.
- ImageSegments.
But...no one solves all the problems
Main Challenges

- Not to use more memory than the one released by swapping.
- Low overhead penalty.
- Group objects in an smart way.
Key aspects

- Mark and trace unused/used objects at runtime.
- The usage of proxies.
- Group unused objects (subgraphs).
Why we need to group objects?

Because if we replace each object by a proxy, we release little memory.

We want to replace a whole group by one or a few proxies.
Why subgraphs?

- Group of objects that are used (or not used) together.

- We need few proxies (for the roots) for several objects.
Experiments done

- Modify Smalltalk VM to mark and trace objects usage.
- Visualize objects and memory usage.
- Take statistics from different scenarios.
DEPLOYED WEB APPLICATION EXAMPLE

Amount of objects

- Used: 4%
- Unused: 96%

Amount of memory

- Used: 15%
- Unused: 85%
Swapping steps and challenges

1. Identify sets of objects and serialize them. Problems: cycles, speed, etc.

2. Write the serialized objects into a file. Problems: file format, encoding, speed, etc.

3. Load the objects from a file. Problems: class reshape, avoid duplicates, speed, etc.
Subgraphs

Roots  Inner  Shared  External

Subgraph to process

A (Root)  B (Root)  C (Root)

D  E  F  G  H  I  J  K  L

Y  Z  X

Wednesday, September 15, 2010
More problems

- Should shared objects be included or not?
- GC moves objects.
- Pointers update.
- Class changes.
- Recreate and reinitialize objects.
- Code executed after loading.
ImageSegment
**ImageSegment basis**

- Only write/swap roots and inner objects.
- Shared objects are NOT swapped.
- Keep an array in memory for the shared objects.
- Update object pointers to point to a relative address inside the arrays (offset).
- Roots are replaced by proxies.
- Uses GC facilities to detect shared objects.
Subgraph traverse

Serialized objects WordArray

Shared objects Array
SUBGRAPH TRAVERSE

Serialized objects WordArray

Shared objects Array
Subgraph traverse

Serialized objects WordArray

Shared objects Array

Wednesday, September 15, 2010
Subgraph traverse

Serialized objects WordArray

Shared objects Array
Subgraph traverse

Serialized objects WordArray

Shared objects Array

A'  B'  C'

D

Wednesday, September 15, 2010
Subgraph traverse

Serialized objects WordArray

Shared objects Array
Subgraph traverse

Serialized objects WordArray
offset

Shared objects Array
Subgraph traverse

Serialized objects WordArray
offset

Shared objects Array
Serialized objects: WordArray

Shared objects: Array
Serialized objects WordArray

Shared objects Array

Subgraph traverse
Subgraph traverse

Serialized objects WordArray

Shared objects Array

Wednesday, September 15, 2010
Subgraph traverse

Serialized objects WordArray

Shared objects Array
Serialized objects WordArray

Shared objects Array

Wednesday, September 15, 2010
Serialized objects WordArray

Shared objects Array

Binary file
**ImageSegment Conclusions**

- ✅ Good speed.
- ✅ Graph traverse is done in VM side.
- ✅ Good use of GC facilities.
- ✗ You have to be aware of shared objects.
- ✗ Bad granularity level.
- ✗ Implicit needed information in object graphs.
Thanks!

Mariano Martinez Peck
marianopeck@gmail.com