GC in Smalltalk

MarkAndCompactGC

Javier Burroni

gera

ESUG 2010

```
Object subclass: #GenerationalGC
```

```
collect
  self
    followRoots;
    followStack;
    rescueEphemerons;
    fixWeakContainers;
    flipSpaces
```

to

from

- generational purgeRoots
- generational follow: object
- generational moveToOldOrTo: object
- generational fixReferencesOrSetTombstone: weakContainer
- generational addInterrupt

ESUG 2011

Object subclass: **#VMGarbageCollector**

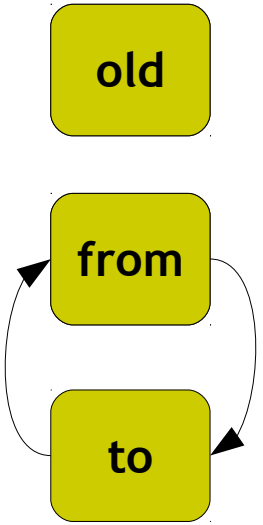
VMGarbageCollector subclass: **#GenerationalGC**

VMGarbageCollector subclass: **#MarkAndCompactGC**

ESUG 2011

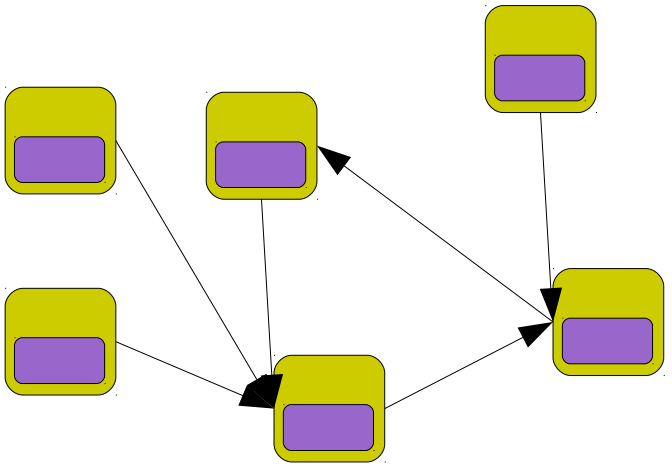
VMGarbageCollector subclass: **#MarkAndCompactGC**

```
collect
  self
    unseeWellKnownObjects;
    followAll;
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
    prepareForCompact;
    compact: oldSpace;
    compact: fromSpace;
    updateOldSpace;
    makeRescuedEphemeronsNonWeak;
    resetFrom;
    decommitSlack;
    addInterrupt
```



MarkAndCompactGC

- Three phases
  - Follow (mark)
  - Set new positions
  - Compact (move)



```
collect
  self
    ...
    followAll;
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
    prepareForCompact;
    compact: oldSpace;
    compact: fromSpace;
    ...
```

MarkAndCompactGC

followAll

- Precondition: **all** objects unseen

VMGarbageCollector subclass: **#MarkAndCompactGC**

MarkAndCompactGC

followAll

- Precondition: **all** objects unseen

VMGarbageCollector subclass: **#MarkAndCompactGC**

```
unseeWellKnownObjects
   nil _beUnseenInLibrary.
   true _beUnseenInLibrary.
   false _beUnseenInLibrary.
   self unseeLibraryObjects;
      unseeCharacters;
      unseeSKernel
```

Libraries

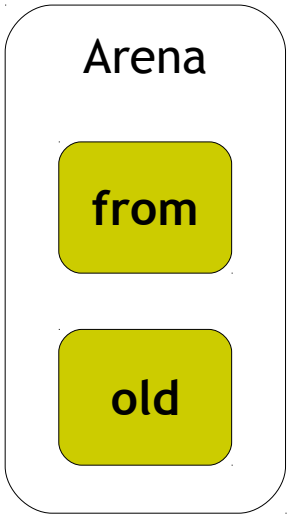
Characters

true
false
nil

MarkAndCompactGC

followAll

- Precondition: **all** objects unseen



VMGarbageCollector subclass: **#MarkAndCompactGC**

- *unseen*: “natural” state in Arena

MarkAndCompactGC

followAll

- Recognize & Mark **all** living objects

VMGarbageCollector subclass: **#MarkAndCompactGC**

MarkAndCompactGC

followAll

- Recognize & Mark **all** living objects

VMGarbageCollector subclass: **#MarkAndCompactGC**

```
followAll
   self
      followSKernel;
      followStack;
      followExtraRoots;
      rescueEphemerons;
      fixWeakContainers;
      followRescuedEphemerons
```

MarkAndCompactGC

followAll

- Recognize & Mark **all** living objects

VMGarbageCollector subclass: **#MarkAndCompactGC**

```
followsSKernel
  self
    follow: self sKernel
    count: self sKernelSize
    startingAt: 1
```

MarkAndCompactGC

```
follow: root count: size startingAt: base
```

- Recognize & Mark **all** living objects

```
(self arenaIncludes: object)
        ifFalse: [
          object _hasBeenSeenInLibrary ifFalse: [
            object _beSeenInLibrary.
            self follow: object...
```

Libraries

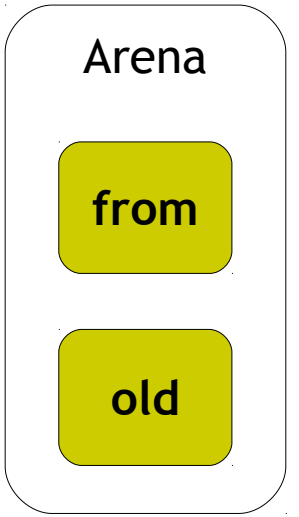
Characters

true
false
nil

MarkAndCompactGC

```
follow: root count: size startingAt: base
```

- Recognize & Mark **all** living objects

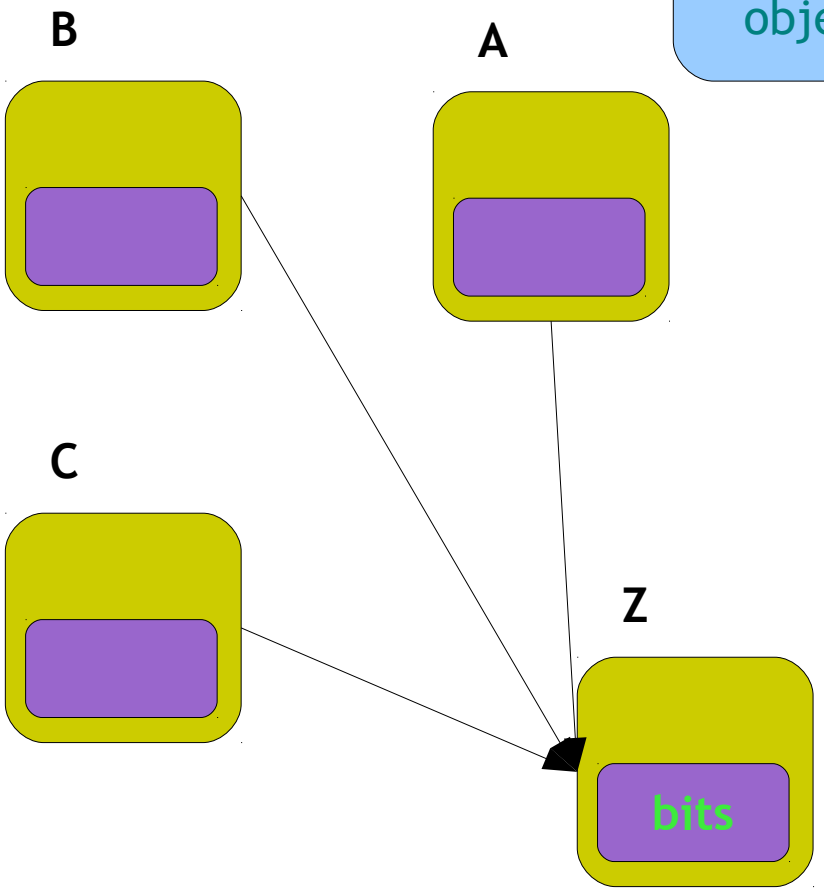



```
(self arenaIncludes: object)
        ifFalse: [
          object _hasBeenSeenInLibrary ifFalse: [
            object _beSeenInLibrary.
            self follow: object...
```

```
        ifTrue: [
          object _threadWith: reference at: oldIndex.
          object _hasBeenSeenInSpace ifFalse: [
            " object _beSeenInSpace "
            self follow: object...]]
```

MarkAndCompactGC

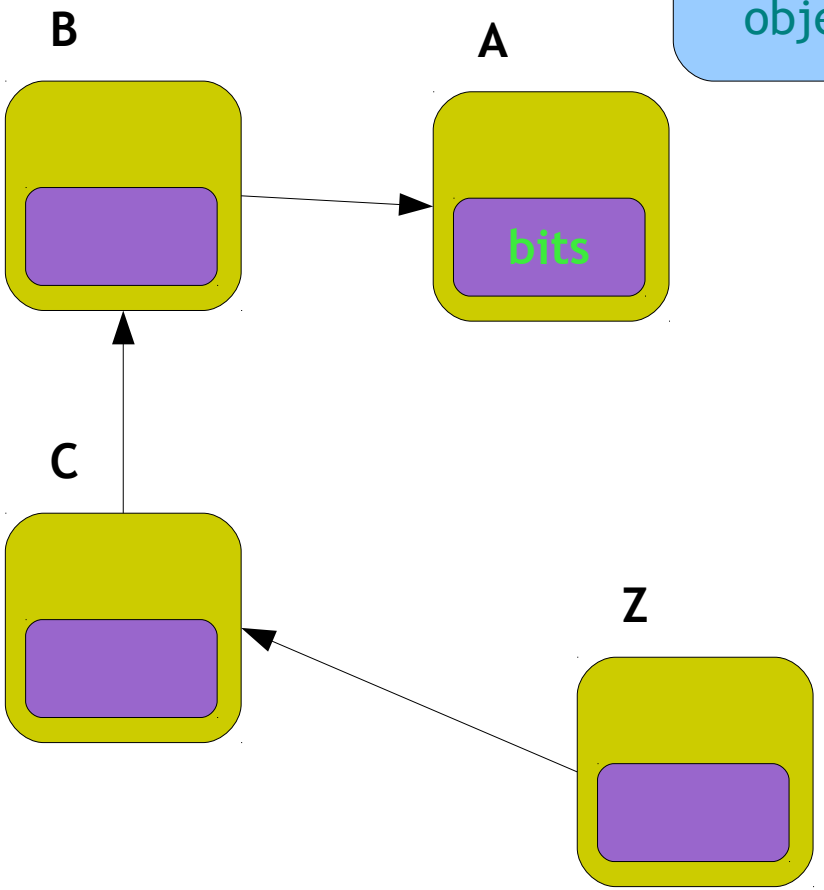
```
follow: root count: size startingAt: base
    ...
    object _threadWith: reference at: oldIndex.
```



Morris, F. L. 1978. A time-and space-efficient garbage compaction algorithm. Communications of the ACM. 21, 8, 662-665.

MarkAndCompactGC

```
follow: root count: size startingAt: base
    ...
    object _threadWith: reference at: oldIndex.
```



Morris, F. L. 1978. A time-and space-efficient garbage compaction algorithm. Communications of the ACM. 21, 8, 662-665.

MarkAndCompactGC

followAll

- Recognize & Mark **all** living objects

VMGarbageCollector subclass: **#MarkAndCompactGC**

followExtraRoots
```
self follow: self extraRoots count: 3 startingAt: 1
```

followRescuedEphemerons
```
self follow: rescuedEphemerons count:...
```

MarkAndCompactGC

followAll

- Recognize & Mark **all** living objects

VMGarbageCollector subclass: **#MarkAndCompactGC**

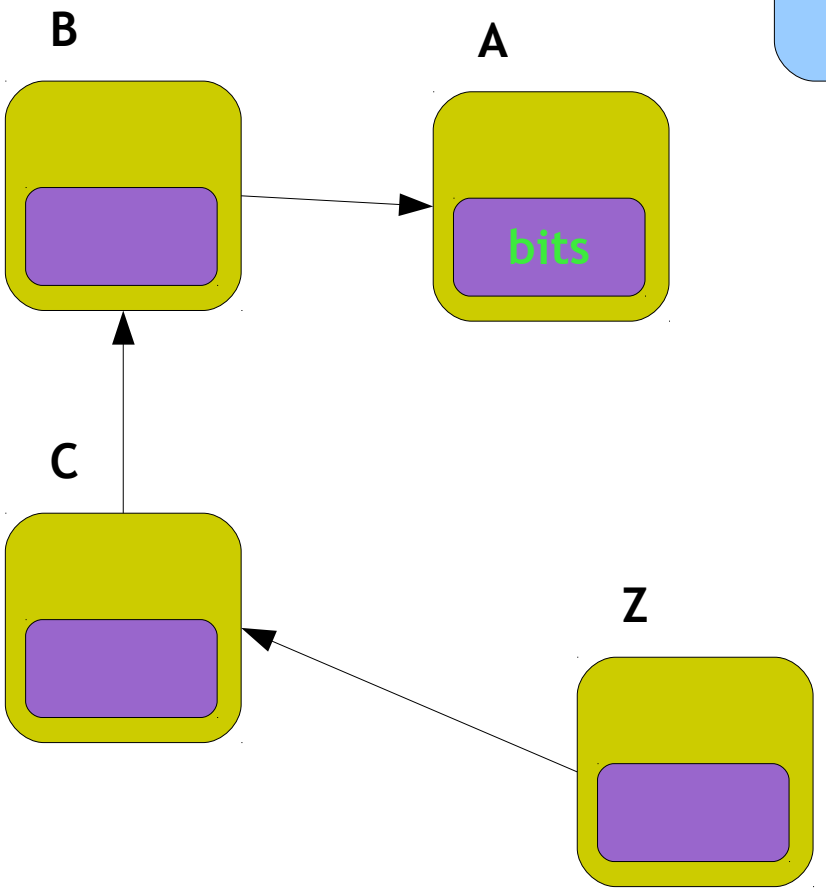
followStack
  super followStack

rescueEphemerons
  super rescueEphemerons

- Implemented **VMGarbageCollector**

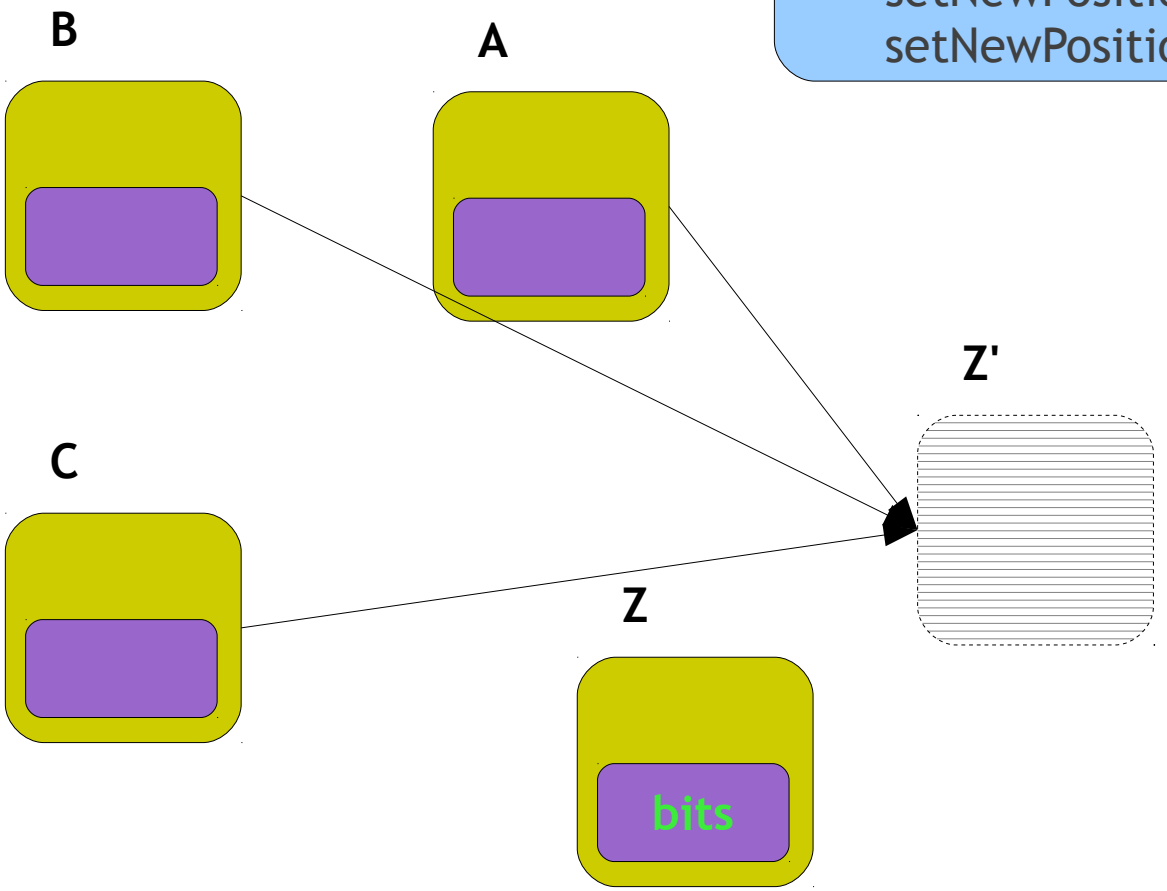
MarkAndCompactGC

```
collect
  ...
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
```



MarkAndCompactGC

```
collect
  ...
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
```



MarkAndCompactGC

```
collect
  ...
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
```

```
setNewPositions: space
  self
    seenObjectsFrom: space base
    to: space nextFree
    do: [:object :headerSize | | newPosition nextReference reference headerBits |
      newPosition := auxSpace nextFree + headerSize.
      reference := object _headerBits _unrotate.

        headerBits := reference _basicAt: 1.
        reference _basicAt: 1 put: newPosition _toObject.



      object _basicAt: -1 put: headerBits; _beSeenInSpace.
      auxSpace nextFree: newPosition + object _byteSize]
```

MarkAndCompactGC

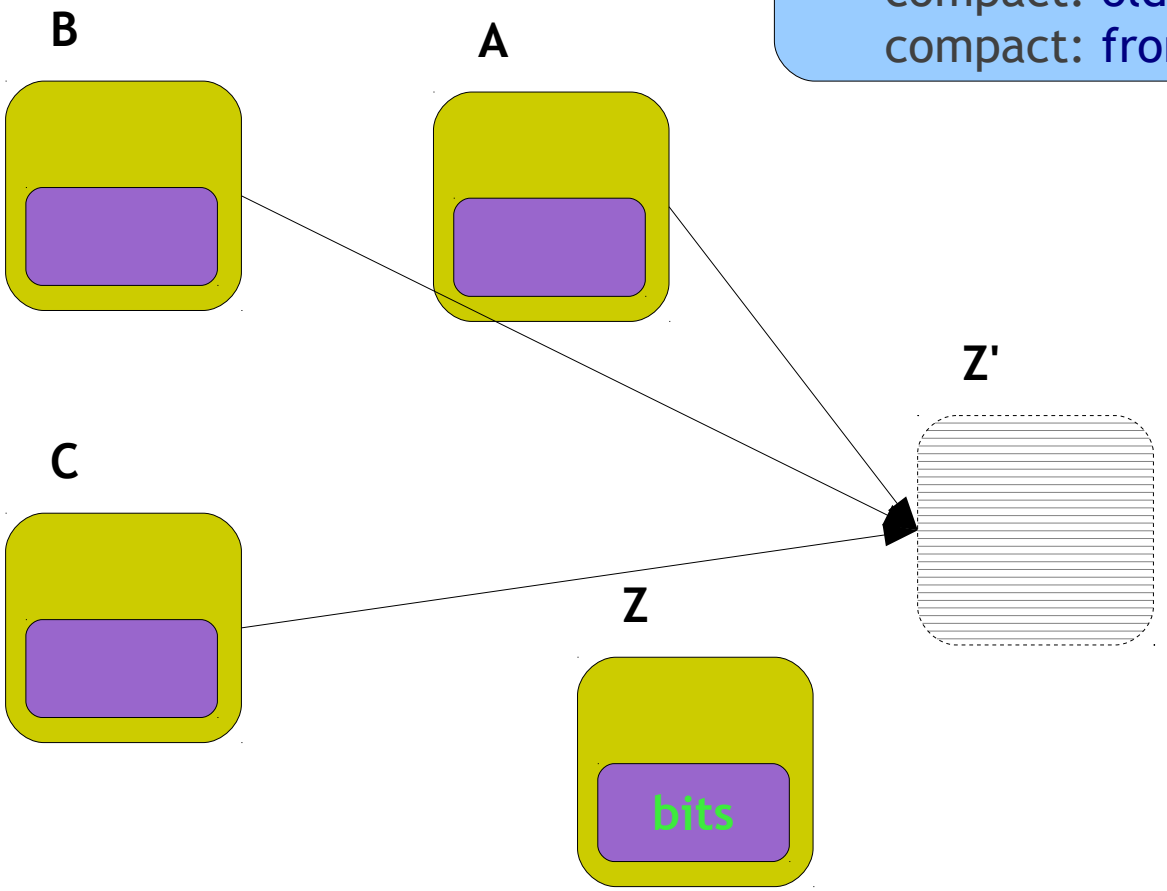
- Unthread references

```
collect
  ...
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
```

```
setNewPositions: space
  self
    seenObjectsFrom: space base
    to: space nextFree
    do: [:object :headerSize | | newPosition nextReference reference headerBits |
      newPosition := auxSpace nextFree + headerSize.
      reference := object _headerBits _unrotate.
      [
        headerBits := reference _basicAt: 1.
        reference _basicAt: 1 put: newPosition _toObject.
        nextReference := headerBits _unrotate.
        nextReference _isSmallInteger]
        whileFalse: [reference := nextReference].
      object _basicAt: -1 put: headerBits; _beSeenInSpace.
      auxSpace nextFree: newPosition + object _byteSize]
```

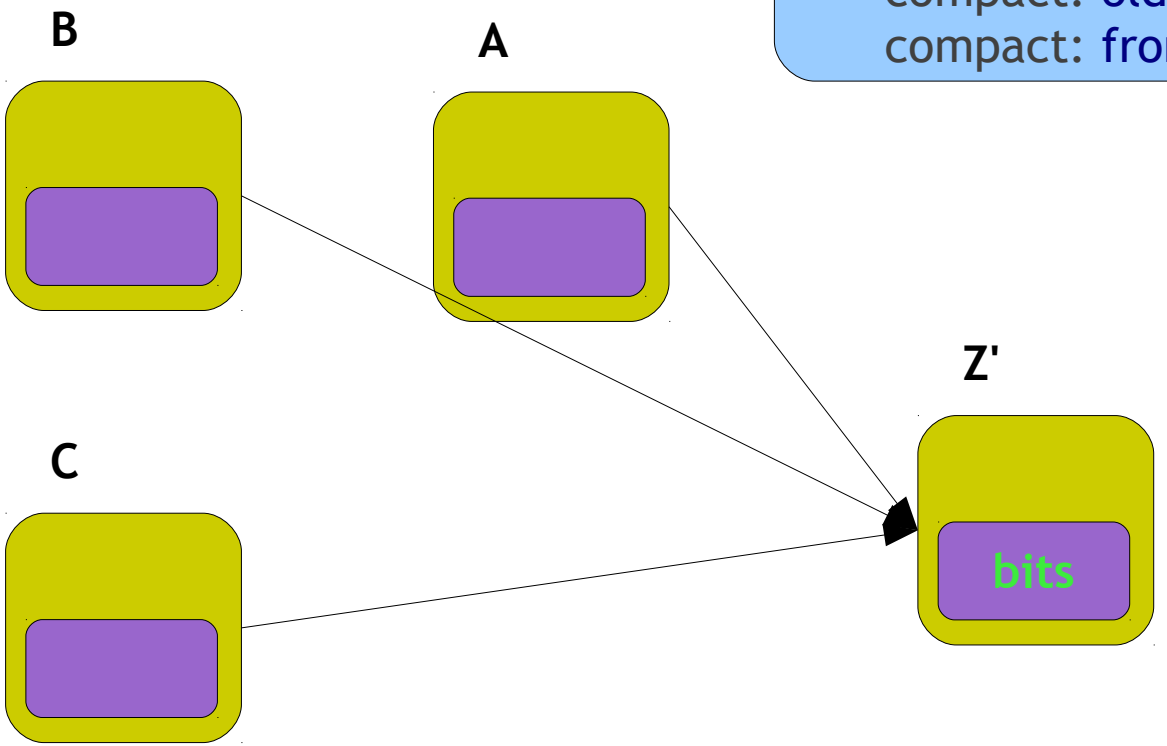
MarkAndCompactGC

```
collect
  ...
    compact: oldSpace;
    compact: fromSpace;
```



MarkAndCompactGC

```
collect
  ...
    compact: oldSpace;
    compact: fromSpace;
```



MarkAndCompactGC

```
collect
  ...
    compact: oldSpace;
    compact: fromSpace;
```

```
compact: space
  self objectsFrom: space base to: space nextFree do: [:object |
    object _hasBeenSeenInSpace ifTrue: [| moved |
      moved := auxSpace shallowCopy: object.
      moved _beUnseenInSpace]]
```

MarkAndCompactGC

```
collect
    ...
        makeRescuedEphemeronsNonWeak;
        updateOldSpace;
```

```
makeRescuedEphemeronsNonWeak
    rescuedEphemerons do: [:ephemeron | ephemeron _haveNoWeaks]
```

```
updateOldSpace
    oldSpace loadFrom: auxSpace
```

MarkAndCompactGC

- Free resources

```
collect
  ...
    decommitSlack;
```

```
decommitSlack
  | limit delta |
  limit := self nextFree + 16rFFF bitAnd: -16r1000.
  delta := self commitedLimit - limit.
  delta < 0 ifTrue: [^self grow].
  delta > 0 ifTrue: [
    limit _decommit: delta.
    self commitedLimit: limit]
```

MarkAndCompactGC

- Free resources

```
collect
  ...
    decommitSlack;
```

```
assembleDecommit
  ...
  return := assembler
    pushConstant: 16r4000;
    pushArg;
    pushR;
    callTo: 'VirtualFree' from: 'KERNEL32.DLL';
    convertToNative;
    shortJump
```

Stop

- We have a Smalltalk Scavenger

Stop

- We have a Smalltalk Scavenger
  - Written in Smalltalk

Stop

- We have a Smalltalk Scavenger
  - Written in Smalltalk
    - Using objects

Stop

- We have a Smalltalk Scavenger
  - Written in Smalltalk
    - Using objects
      - Instantiating objects

Stop

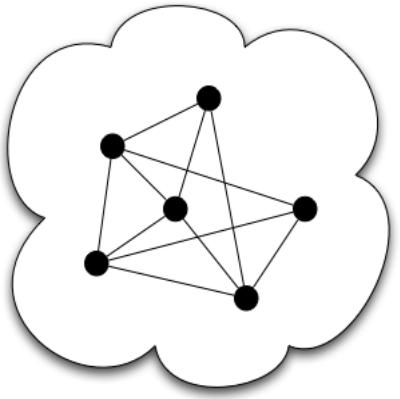
- We have a Smalltalk Scavenger
  - Written in Smalltalk
    - Using objects
      - Instantiating objects
- How can **it** work?

Stop

- We have a Smalltalk Scavenger
  - Written in Smalltalk
    - Using objects
      - Instantiating objects
- How can **it** work?

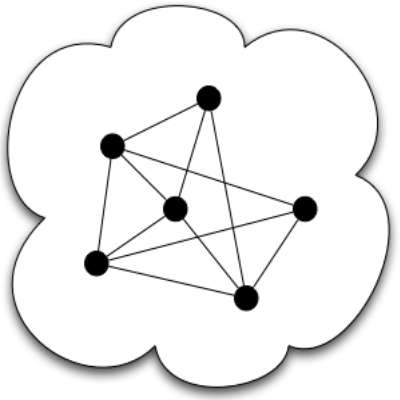
Object Closure

Object Closure



- What happens in ~~Vegas~~GC stays in ~~Vegas~~GC
  - Self contained objects graph
  - Created objects do not live after GC
  - No message *new* in our code
  - Created objects:
    - Block Closures
    - Environment Contexts
    - Arrays
      - To interface with the **Host VM**

Object Closure



- BlockClosure

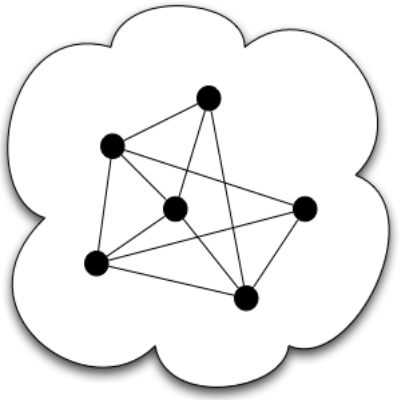
```
makeRescuedEphemeronsNonWeak
  rescuedEphemerons do: [:ephemeron | ephemeron _haveNoWeaks]
```

- Environment context

```
compact: space
  | moved |
  self objectsFrom: space base to: space nextFree do: [:object |
    object _hasBeenSeenInSpace ifTrue: [
      moved := auxSpace shallowCopy: object]
```

The VM will instantiate both at end of young space

Object Closure



- BlockClosure

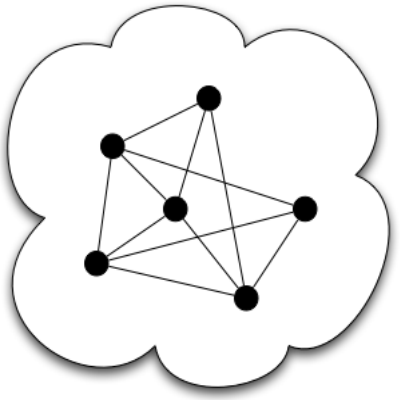
```
makeRescuedEphemeronsNonWeak
  rescuedEphemerons do: [:ephemeron | ephemeron _haveNoWeaks]
```

- Environment context

```
compact: space
  | moved |
  self objectsFrom: space base to: space nextFree do: [:object |
    object _hasBeenSeenInSpace ifTrue: [
      moved := auxSpace shallowCopy: object]
```

The VM will instantiate both at end of young space
we only scan up to space's size when the GC starts

Object Closure

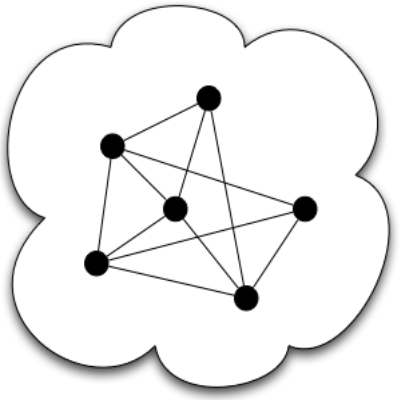


- Host VM interface arrays

```
allocateArrays
   rememberSet on: oldSpace; emptyReserving: 16r100.
   literalsReferences emptyReserving: 16r200.
   classCheckReferences emptyReserving: 16r100.
   nativizedMethods emptyReserving: 16r100.
   self
      allocateWeakContainersArray;
      allocateEphemeronsArray;
      forgetNativeArrays
```

allocated at end of *oldSpace*, just before returning

Object Closure



- Host VM interface arrays

```
at: index
  ^contents _basicAt: index + 1
```

```
at: index put: value
  ^contents _basicAt: index + 1 put: value
```

use *_primitives* instead of primitives

Demo

```
[Audience hasQuestions] whileTrue: [
    self answer: Audience nextQuestion].

Audience do: [:you | self thank: you].

self returnTo: Audience
```