A little Journey in a Virtual Machine

Igor Stasenko

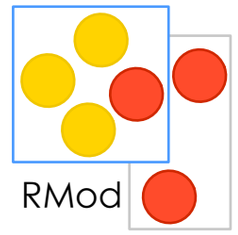
RMoD Team
INRIA Lille Nord-Europe

INRIA-CEA-EDF School Deep into Smalltalk
March 2011

Goals

- VMMaker internals
- VM Internals
- Plugins
- Primitives
- Object format
- Build your own VM!
- Build your first plugin

Squeak Virtual Machine



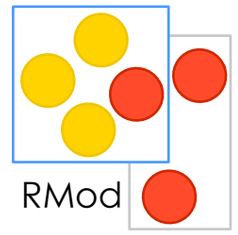
- Can load and run image file - an object memory snapshot

- Runs on multiple platforms

- Written in smalltalk (VMMaker)

- platform-specific code are kept separately (‘platforms’ code)

Getting started

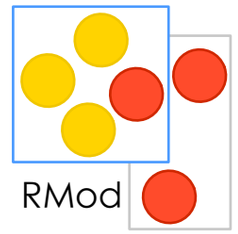


Where to get VM source code?

Smalltalk VMMaker code:
- http://squeaksource.com/VMMaker

Platform-specific code:
- http://squeakvm.org/svn/squeak/trunk
- ***http://gitorious.org/squeak-vm***
- ***http://gitorious.org/cogvm (for Cog)***

My current toolset



- cmake
- git
- pharo-core 1.3
- VirtualBox, shell, gcc , gdb ... and brain :)

To load platform sources, go to

http://gitorious.org/+squeak-vm-developers

and clone one of VMs, or if you prefer SVN, read how to do that at:

http://squeakvm.org/index.html

GITORIOUS

Activities Projects Teams

ducasse stef-squeakvm

squeak-vm ➡ stef-squeakvm

Clone & push urls | GIT | HTTP | SSH | git@gitorious.org:~ducasse/squeak-vm/stef-squeak ?

Adding this repository as a pushable origin:

```
git remote add origin git@gitorious.org:~ducasse/squeak-vm/stef-squeakvm.git
# to push the master branch to the origin remote we added above:
git push origin master
# after that you can just do:
git push
```

Cloning this repository:

```
git clone git://gitorious.org/~ducasse/squeak-vm/stef-squeakvm.git stef-
squeakvm
cd stef-squeakvm
```

Add this repository as a remote to an existing local repository:

```
git remote add stef-squeakvm git://gitorious.org/~ducasse/squeak-vm/stef-
squeakvm.git
git fetch stef-squeakvm
git checkout -b my-local-tracking-branch stef-
squeakvm/master_or_other_branch
```

Branches: master

Clone of: blessed

Commit log | Source tree | Merge requests (0) | Clone repository | Unwatch

Activities

Wednesday February 02 2011

REPOSITORY 15:45 **ducasse** cloned squeak-vm/blessed

New repository is in stef-squeakvm

Project: squeak-vm
Owner: ~ducasse
Clone of: squeak-vm/blessed.git
Created: 02 Feb 15:45

Clone repository
Request merge
Manage collaborators
Edit repository

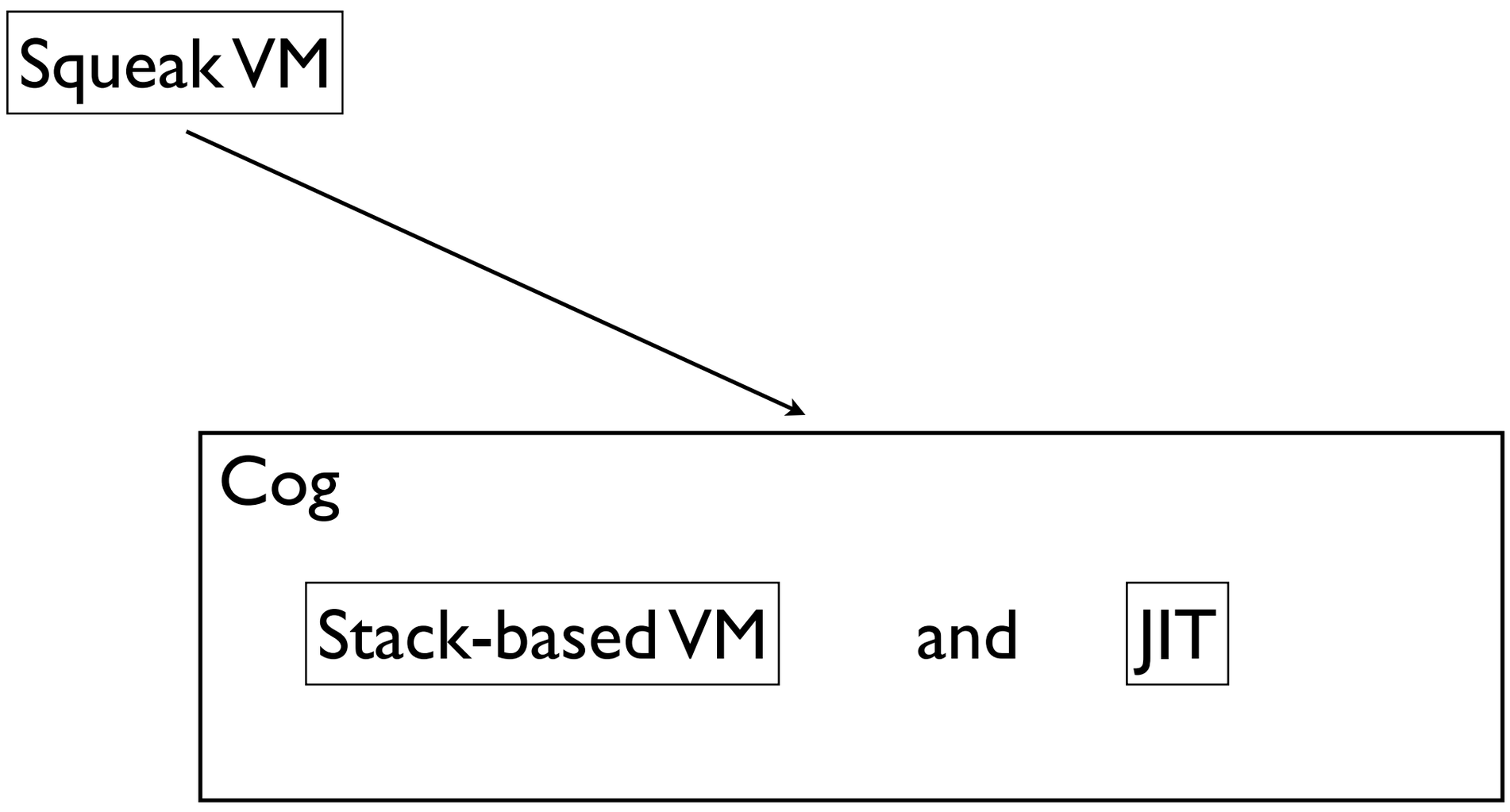
Committers

ducasse (creator)

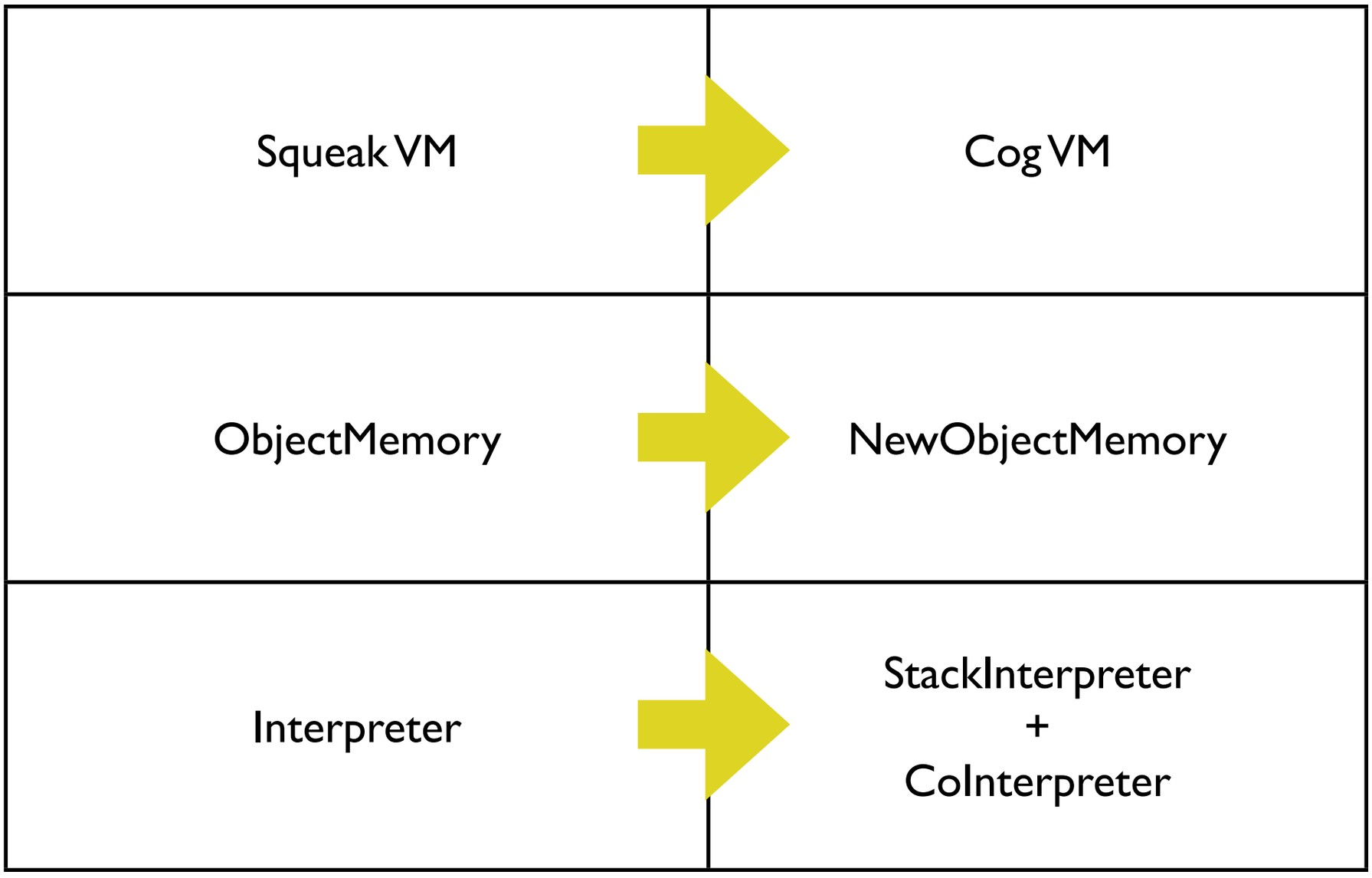
Repository clones

No clones on Gitorious yet of this repository

We are in transition

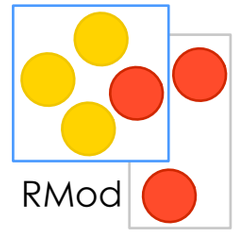


We are in transition



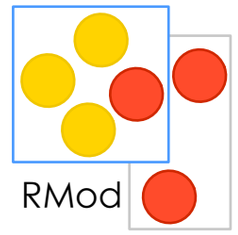
VM autopsy

Structure of VMMaker



- Interpreter implementation
  - ObjectMemory class
  - Interpreter class
  - plugins (IntepreterPlugin subclasses)
- Code generator (CCodeGenerator)
  - translating smalltalk code aka “slang” to C
- VMMaker , VMMakerTool
  - a front end for code generation
- InterpreterSimulator
  - yes, you can run VM emulated!
- Plugins

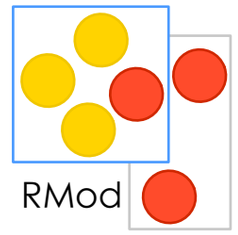
ObjectMemory class



Responsible for:

- object format
- memory management (heap, object enumeration, GC)
- special objects (nil, true, false, String, Character etc)

ObjectMemory>> object format



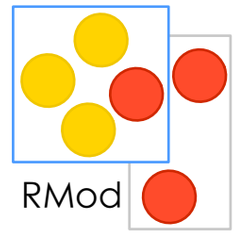
hashBitsOf: oop

```
^ ((self baseHeader: oop) >> HashBitsOffset)
  bitAnd: HashMaskUnshifted
```

isYoung: oop

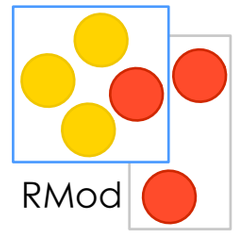
```
<api>
  ^(self isNonIntegerObject: oop)
  and: [self oop: oop isGreaterThanOrEqualTo: youngStart]
```

ObjectMemory>> enumeration




```
initialInstanceOf: classPointer
        "Support for instance enumeration. Return the first instance
        of the given class, or nilObj if it has no instances."
        | thisObj thisClass |
        thisObj := self firstAccessibleObject.
        [thisObj = nil]
                whileFalse: [thisClass := self fetchClassOf: thisObj.
                        thisClass = classPointer ifTrue: [^ thisObj].
                        thisObj := self accessibleObjectAfter: thisObj].
        ^ nilObj

firstAccessibleObject
        "Return the first accessible object in the heap."
        | obj |
        obj := self firstObject.
        [self oop: obj isLessThan: endOfMemory]
                whileTrue: [(self isFreeObject: obj) ifFalse: [^ obj].
                        obj := self objectAfter: obj].
```

ObjectMemory>>memory management



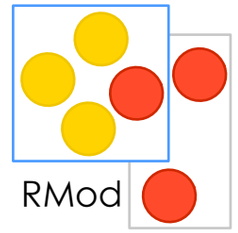
Instantiating new objects:

#instantiateClass: classPointer indexableSize: size
#instantiateSmallClass: classPointer sizeInBytes: sizeInBytes

Garbage collection:

#incrementalGC
#fullGC

ObjectMemory>> special objects



- there are single “special objects” array,

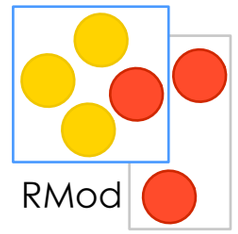
```
splObj: index
        "Return one of the objects in the SpecialObjectsArray"
        ^ self fetchPointer: index ofObject: specialObjectsOop
```

- the rest of special objects are seen through it:

```
classArray
        <api>
        ^self splObj: ClassArray  “ClassArray == 7”
```

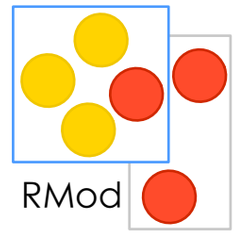
- see Smalltalk specialObjectsArray

Interpreter



- execution machinery
- basic primitives (#primitiveAdd etc)
- public VM API (used by plugins)
- services (scheduling, basic I/O)
- (Huge beast)

Interpreter>> Execution machinery



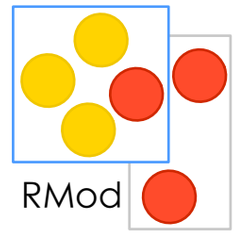
- interpreting bytecodes in #interpret loop

interpret

```
<inline: false> "should *not* be inlined into sendInvokeCallback:Stack:Registers:Jmpbuf:"
"record entry time when running as a browser plug-in"
self browserPluginInitialiseIfNeeded.
self internalizeIPandSP.
self fetchNextBytecode.
[true] whileTrue: [self dispatchOn: currentBytecode in: BytecodeTable].
localIP := localIP - 1.  "undo the pre-increment of IP before returning"
self externalizeIPandSP.
^nil
```

- drawbacks: you cannot ‘call’ interpreter, because it is an infinite loop.

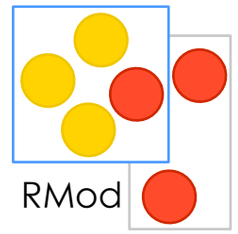
Interpreter standard prims



- All numbered primitives: integer, float ops, object creation, scheduling, controlling GC, save&quit etc..

```
primitiveNew
		"Allocate a new fixed-size instance. Fail if the allocation would leave less than lowSpaceThreshold bytes free. May cause a GC"

		| class spaceOkay |
		class := self stackTop.
		"The following may cause GC!"
		spaceOkay := self sufficientSpaceToInstantiate: class indexableSize: 0.
		self success: spaceOkay.
		successFlag ifTrue: [ self push: (self instantiateClass: self popStack indexableSize: 0) ]
```

Interpreter>>public VM API

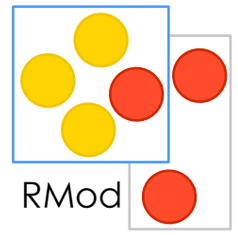


- InterpreterProxy - a “canonical” way to communicate with VM

  (interpreterProxy fooBar: x) ==> (Interpreter>>fooBar:x)
  (see platforms/Cross/vm/sqVirtualMachine.h / .c)

- Other methods, marked by <api> pragma, can be used by various C sources when linked statically

Interpreter basic I/O



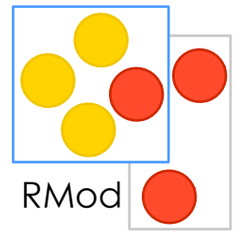
- Display (#primitiveSetDisplayMode, #primitiveDeferDisplayUpdates)

- Keyboard (#primitiveGetNextEvent)

- Mouse (#primitiveGetNextEvent)

- Beep, (yeah - #primitiveBeep)

VM is HUGE beast



- Interpreter methodDict size 580
- ObjectMemory methodDict size 274
- Interpreter allInstVarNames

  #('memory' 'youngStart' 'endOfMemory' 'memoryLimit' 'nilObj' 'falseObj' 'trueObj' 'specialObjectsOop' 'rootTable' 'rootTableCount' 'extraRoots' 'extraRootCount' 'weakRoots' 'weakRootCount' 'child' 'field' 'parentField' 'freeBlock' 'lastHash' 'allocationCount' 'lowSpaceThreshold' 'signalLowSpace' 'compStart' 'compEnd' 'fwdTableNext' 'fwdTableLast' 'remapBuffer' 'remapBufferCount' 'allocationsBetweenGCs' 'tenuringThreshold' 'gcBiasToGrow' 'gcBiasToGrowGCLimit' 'gcBiasToGrowThreshold' 'statFullGCs' 'statIncrGCs' 'statFullGCUsecs' 'statIncrGCUsecs' 'statGCEndTime' 'statIGCDeltaUsecs' 'statTenures' 'statRootTableOverflows' 'freeContexts' 'freeLargeContexts' 'interruptCheckCounter' 'totalObjectCount' 'shrinkThreshold' 'growHeadroom' 'headerTypeBytes' 'youngStartLocal' 'statMarkCount' 'statMarkCountLocal' 'statSweepCount' 'statMkFwdCount' 'statCompMoveCount' 'statGrowMemory' 'statShrinkMemory' 'statRootTableCount' 'statAllocationCount' 'statSurvivorCount' 'statSpecialMarkCount' 'forceTenureFlag' 'gcStartUsecs' 'activeContext' 'theHomeContext' 'method' 'receiver' 'instructionPointer' 'stackPointer' 'localIP' 'localSP' 'localHomeContext' 'localReturnContext' 'localReturnValue' 'messageSelector' 'argumentCount' 'newMethod' 'currentBytecode' 'successFlag' 'primitiveIndex' 'primitiveFunctionPointer' 'methodCache' 'atCache' 'lkupClass' 'reclaimableContextCount' 'nextPollTick' 'nextWakeupTick' 'lastTick' 'interruptKeycode' 'interruptPending' 'semaphoresToSignalA' 'semaphoresUseBufferA' 'semaphoresToSignalCountA' 'semaphoresToSignalB' 'semaphoresToSignalCountB' 'savedWindowSize' 'fullScreenFlag' 'deferDisplayUpdates' 'pendingFinalizationSignals' 'compilerInitialized' 'extraVMMemory' 'receiverClass' 'interpreterProxy' 'showSurfaceFn' 'interruptCheckCounterFeedBackReset' 'interruptChecksEveryNms' 'externalPrimitiveTable' 'primitiveTable' 'globalSessionID' 'jmpBuf' 'jmpDepth' 'jmpMax' 'suspendedCallbacks' 'suspendedMethods' 'profileProcess' 'profileMethod' 'profileSemaphore' 'nextProfileTick' 'metaclassSizeBytes' 'statIOProcessEvents' 'statCheckForEvents' 'statQuickCheckForEvents' 'statProcessSwitch' 'statPendingFinalizationSignals' 'gcSemaphoreIndex')

- Still much better than writing everything in C!

But we armed well!

Object format

- Object pointer structure

- Object header structure

- Object formats

Object pointer structure

Object pointers are tagged:

First bit is 1 (one)

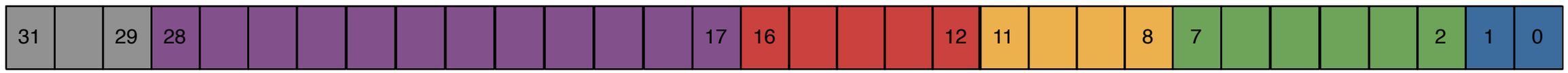
a 31-bit SmallInteger instance (2147483648 possible Values)

First bit is 0

direct pointer to object in object memory. Always points to object header word.

```
isIntegerObject: objectPointer
        ^ (objectPointer bitAnd: 1) > 0
```

Object Header structure



ObjectMemory>>#baseHeader:

Header type
Compact class index
Object size
Object hash
Object Format
Garbage collector

Garbage Collector (#isMarked:)
Mark : used to mark objects during mark phase
Root : set if object is a root object.
- the third bit is unused

Compact Class index (#compactClassIndexOf:)
non-zero value if the class of the object pointed is a compact, represent the index in the compactClass array

Object Size (#sizeBitsOf:)
The object size in 32-bit machine words.
For objects with size which don't fit in this field, we put all ones,
and store object size in separate 32-bit word of object header (header type == 0)

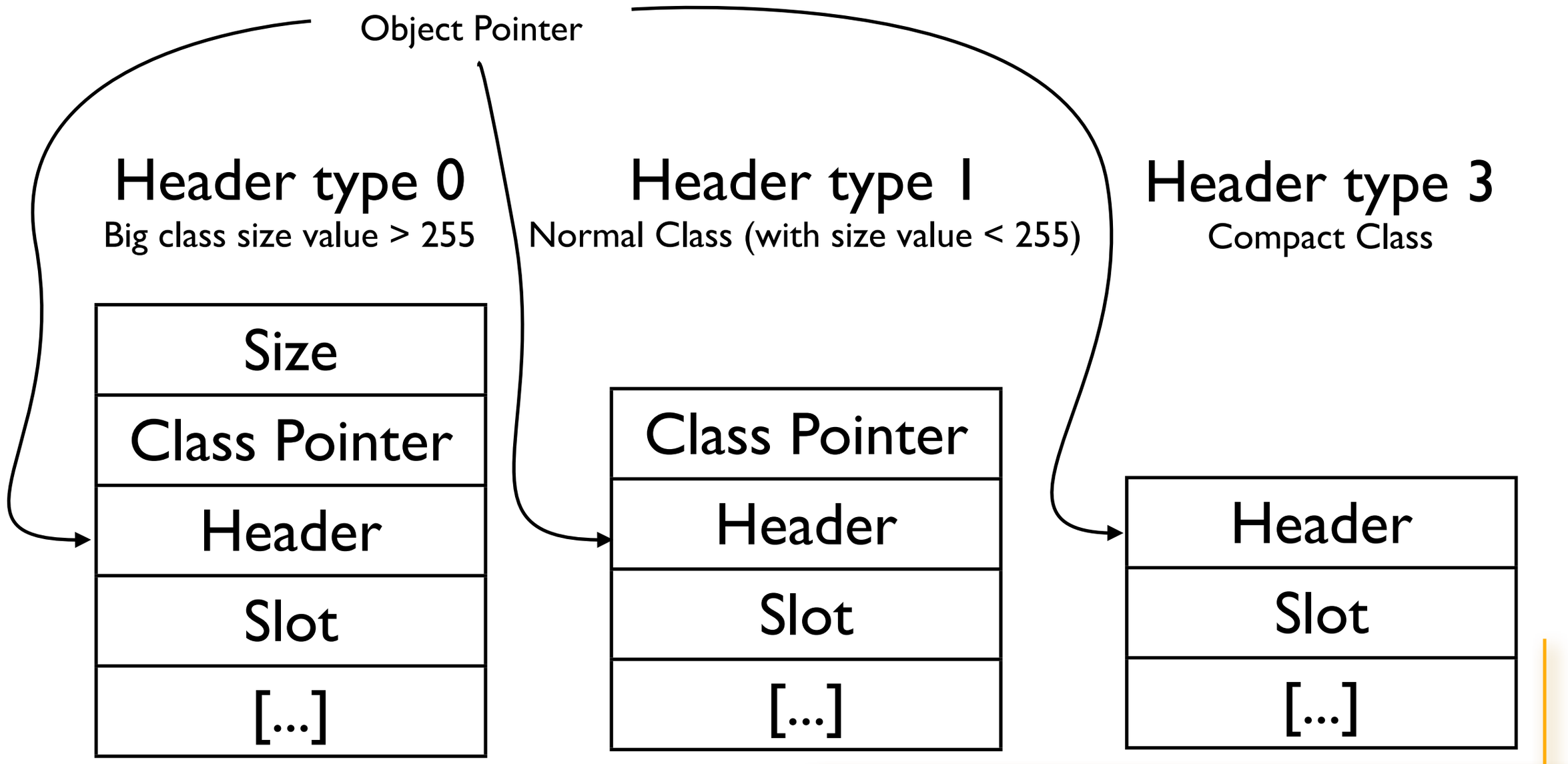
Object Hash (#hashBitsOf:)
12 bits with some random value, assigned at object creation. See #identityHash

Object Format (#formatOf:)
16 possible values describing the type of the object.
An exhaustive list in a dedicated slide.

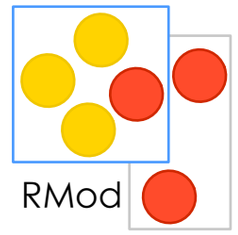
Header Type (#headerType:)
0 : 3 Words Objects (Size and Class)
1 : 2 Words Objects (usually class, which not compact class)
2 : Word free
3 : 1 Work object (usually compact class)

Header Types

#headerType:



Object Format : different type of Fields



fixed fields:

used for instance variables.

```
Object subclass: #Point
        instanceVariableNames: 'x y'
```

- Point has two fixed fields.

indexable fields:

accessed by index

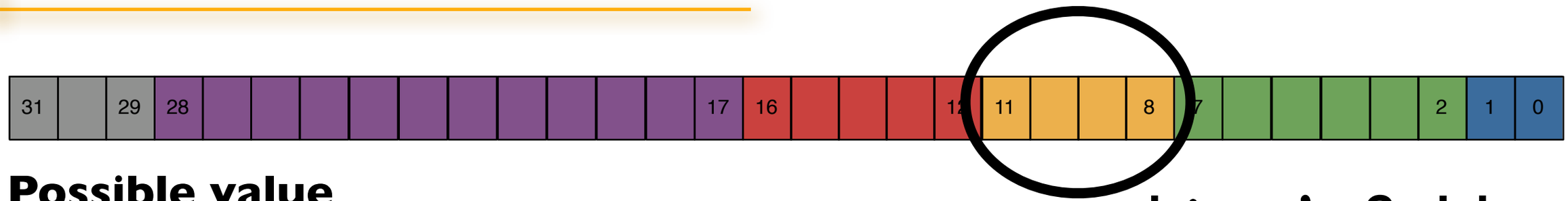
```
arr := Array new: 50.
arr at: 32.
```

indexable fields are accessed via primitives (#basicAt: #basicAt:put:)

ArrayedCollection **variableSubclass:** #Array

....

Object Format :



object's fields format

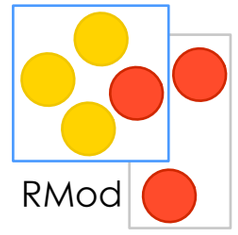
Possible value

0 : the object have no fields
1 : fixed fields only
2 : indexable fields only
3 : both fixed and indexable fields
4 : both fixed and indexable weak fields
5 : Unused
6 : indexable word (32-bit) fields only
7 : indexable long (64-bit) fields (only in 64-bit images)
8-11 : indexable byte fields only (low 2 bits are low 2 bits of size)

Additional possible values for CompiledMethod

12-15 : # of literal oops specified in method header,
followed by indexable bytes(low 2 bits are low 2 bits of size)

Structure Navigation

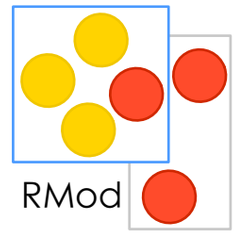


- fetching object's class

- inst vars (aka fixed slots)

- indexed vars aka variable slots

Fetching an instance variable



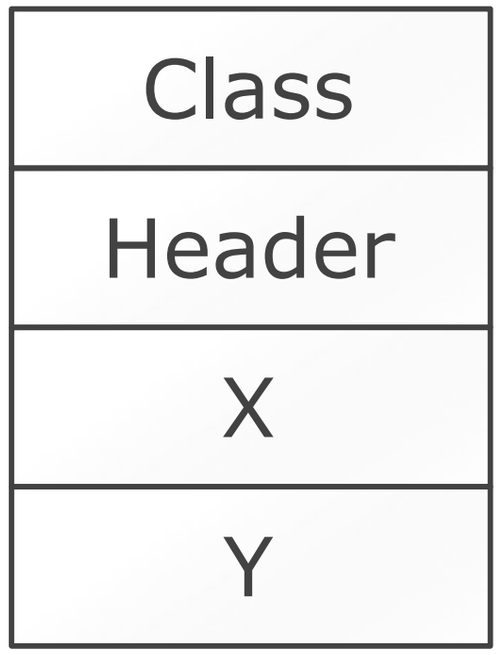
Instance Variable access is Indexed in VM

Point allInsVarNames
 #('x' 'y')

x := self ***fetchPointer:*** 0
  ofObject: aPointInstance

y := self fetchPointer: 1
  ofObject: aPointInstance

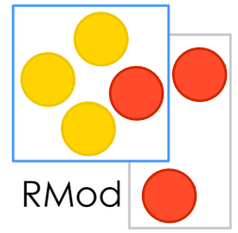
Instance of Point



oop + BaseHeaderSize + (fieldIndex << ShiftForWord)

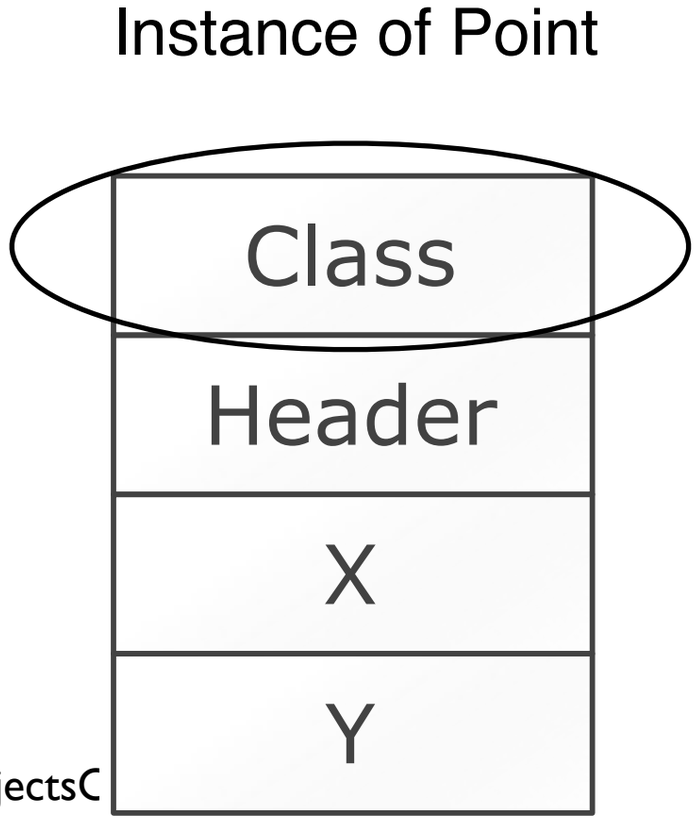
Storing:
self ***storePointer:*** 0 ***ofObject:*** oop ***withValue:*** someOop

Fetching object's class

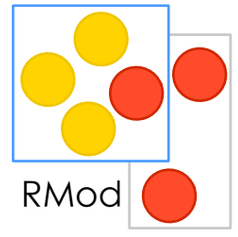


fetchClassOf: oop.

```
fetchClassOf: oop
                    | ccIndex |
                    <inline: true>
                    <asmLabel: false>
                    (self isIntegerObject: oop) ifTrue: [^ self splObj: ClassInteger].

                    (ccIndex := (self compactClassIndexOf: oop)) = 0
                            ifTrue: [^(self classHeader: oop) bitAnd: AllButTypeMask]
                            ifFalse: "look up compact class"
                                    [^self fetchPointer: ccIndex - 1
                                            ofObject: (self fetchPointer: CompactClasses
                                                                    ofObject: specialObjectsC
```



indexed variable access




```
primitiveSetWindowLabel
        "Primitive. Set the OS window's label"
        | labelOop sz |
        <export: true>
        labelOop := self stackTop.
        (self isBytes: labelOop) ifFalse:[^self success: false].
        sz := self byteSizeOf: labelOop.
        self ioSetWindowLabel: (self firstIndexableField: labelOop) OfSize: sz.
        successFlag ifTrue:[self pop: self methodArgumentCount]
```

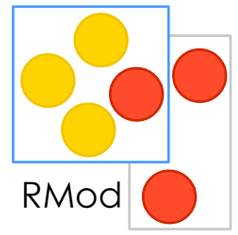
self isBytes: stringOop
- checks that object format is variable bytes format (ByteString, ByteArray etc)

ArrayedCollection **variableByteSubclass:** #ByteArray

self **byteSizeOf:** labelOop
- answers the size of object in bytes (minus header etc)

self **firstIndexableField:** labelOop
- answers the pointer to first indexable field

Fetching an instance variable(other)



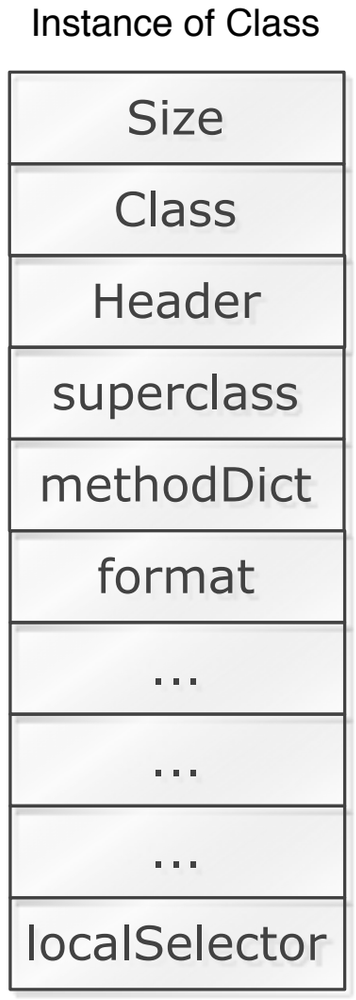
Class allInsVarNames

#('superclass' 'methodDict' 'format' 'instanceVariables' 'organization' 'subclasses' 'name' 'classPool' 'sharedPools' 'environment' 'category' 'traitComposition' 'localSelectors')

Interpreter class>>#initializeClassIndices

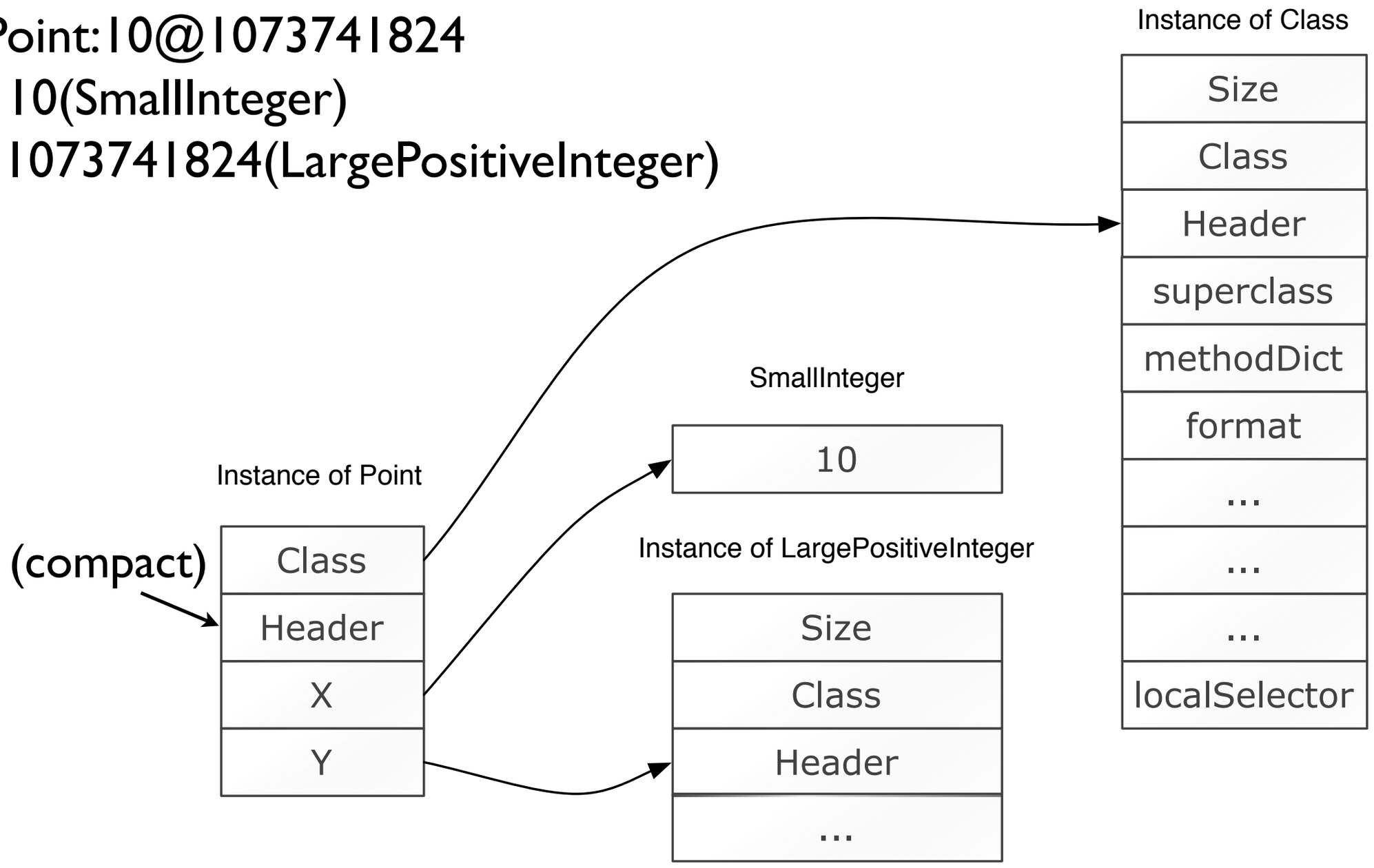
```
SuperclassIndex := 0.
MessageDictionaryIndex := 1.
InstanceSpecificationIndex := 2.
```

```
superClass := self fetchPointer: SuperclassIndex ofObject: aClassInstance
methodDict := self fetchPointer: MessageDictionaryIndex ofObject: aClassInstance
format := self fetchPointer: InstanceSpecificationIndex ofObject: aClassInstance
```



How all is connected ?

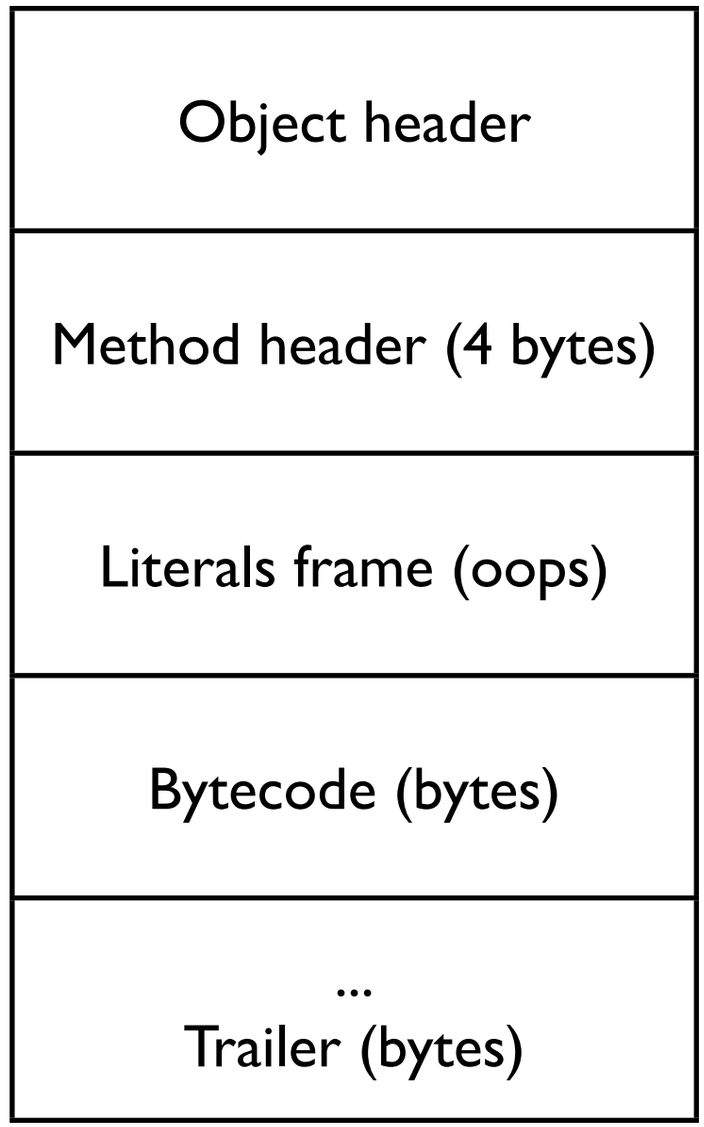
aPoint:10@1073741824
x: 10(SmallInteger)
y: 1073741824(LargePositiveInteger)



CompiledMethod

A bit exotic object format
 - variable oop slots (literals)
 - variable byte slots (bytecode & rest)

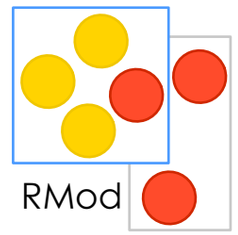
VM "knows" how to deal with it.



Method Header

(index 0)	9 bits:	main part of primitive number (#primitive)
(index 9)	8 bits:	number of literals (#numLiterals)
(index 17)	1 bit:	whether a large frame size is needed (#frameSize)
(index 18)	6 bits:	number of temporary variables (#numTemps)
(index 24)	4 bits:	number of arguments to the method (#numArgs)
(index 28)	1 bit:	high-bit of primitive number (#primitive)
(index 29)	1 bit:	flag bit, ignored by the VM (#flag)

Literals Frame



Array of all literals used in the method.

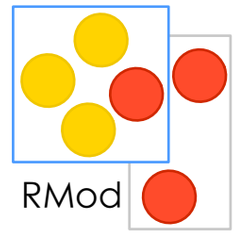
```
corner: aPoint
                "Answer a Rectangle whose origin is the receiver and whose corner is
                aPoint. This is one of the infix ways of expressing the creation of a
                rectangle."

                ^Rectangle origin: self corner: aPoint
```

(Point>>#corner:) **literals**

{#origin:corner:. (#Rectangle->Rectangle). #corner:. (#Point->Point)}

(see Interpreter>>**#literalCountOf: , #literal:ofMethod: , #primitiveObjectAt , #primitiveObjectAtPut**)

ByteCode

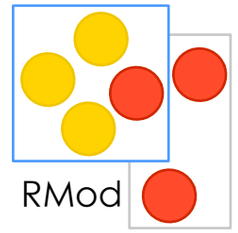


Bytecode is compact and platform neutral representation of machine code instructions, interpreted by VM.

Operations can be reduced to:

- Stack manipulation, pop/push
- Message sending
- Variable manipulation : store
- Closure management
- Return

ByteCode




```
CompiledMethod>>initialPC
		"Answer the program counter for the receiver's first bytecode."

		^ (self numLiterals + 1) * Smalltalk wordSize + 1
```

See Interpreter class>>#initializeBytecodeTable

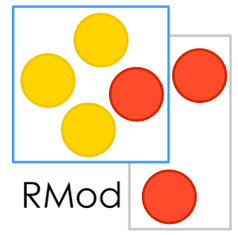
....

```
( 0  15 pushReceiverVariableBytecode)
( 16  31 pushTemporaryVariableBytecode)
( 32  63 pushLiteralConstantBytecode)
( 64  95 pushLiteralVariableBytecode)
( 96 103 storeAndPopReceiverVariableBytecode)
(104 111 storeAndPopTemporaryVariableBytecode)
(112 pushReceiverBytecode)
(113 pushConstantTrueBytecode)
(114 pushConstantFalseBytecode)
(115 pushConstantNilBytecode)
(116 pushConstantMinusOneBytecode)
```

....

Object format: Questions?

CCodeGenerator



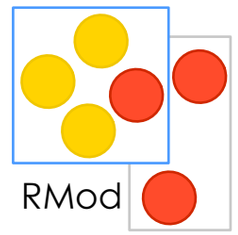
Translating smalltalk “slang” to C.

CCodeGenerator

VMPluginCodeGenerator

SmartSyntaxPluginCodeGenerator

CCodeGenerator in a nutshell

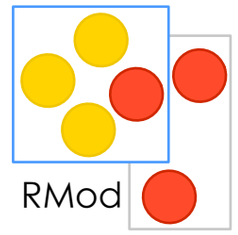



```
Object subclass: #ExampleSlangClass
        instanceVariableNames: 'value'
        classVariableNames: ''
        poolDictionaries: ''
        category: 'JourneyInTheVM'
```

```
foo
        ^ self malloc: 100
```

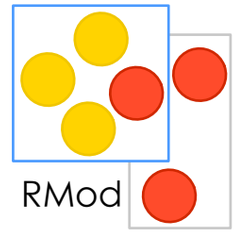
```
value
        ^ value + 1
```

CCodeGenerator in a nutshell



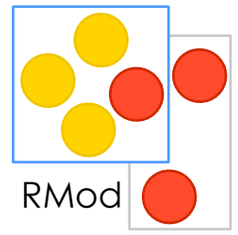

```
CCodeGenerator new
        vmClass: ExampleSlangClass;
        addClass: ExampleSlangClass;
storeCodeOnFile: 'foo.c' doInlining: true
```

Generated code




```
/* Automatically generated from Squeak on an Array(2 March 2011 4:21:01 pm) */

static char __buildInfo[] = "Generated on an Array(2 March 2011 4:21:01 pm). Compiled on "__DATE__ ;


#include "sq.h"

/*** Constants ***/

/*** Function Prototypes ***/
static sqInt foo(void);
static sqInt value(void);

/*** Variables ***/
sqInt value;


static sqInt
foo(void)
{
    return malloc(100);
}

static sqInt
value(void)
{
    return value + 1;
}
```

Code translation basics



- converts message sends to function calls:

```
z := self foo: x bar: y
```

→

```
z = foobar(x,y);
```

- binary & arithmetic expressions converted to their C equivalent:

```
^ a+b.
```

→

```
return a+b;
```

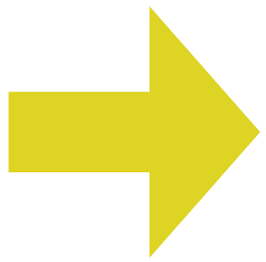
```
a bitShift: 4
```

→

```
return a << 4;
```

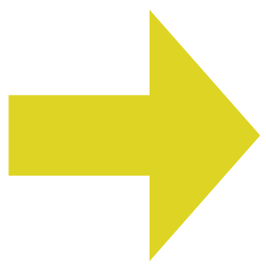
Code translation: control flow

```
^ self flag ifTrue: [ 5 ] ifFalse: [ 10 ].
```



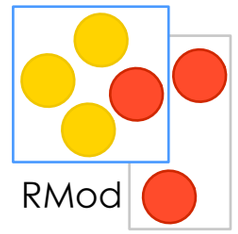
```
return (flag()
        ? 5
        : 10);
```

```
[ value - 1 > 0 ] whileTrue: [ value := value + 1 ]
```



```
while ((value - 1) > 0) {
        value += 1;
   }
```

CodeGen: instance variables



- placed at global module scope

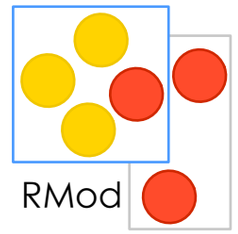
```
Object subclass: #ExampleSlangClass
        instanceVariableNames: 'value'
...
```

declaration

```
…
/*** Variables ***/
sqInt value;
…
```

C code

CodeGen: class & pool vars

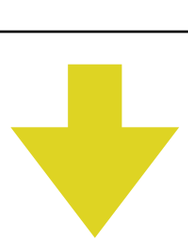



```
...
classVariableNames: 'ClassVar'
...
```

declaration

```
initialize
    ClassVar := 10.
```

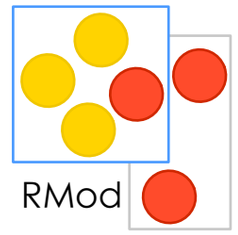
don't forget to initialize before translating :)



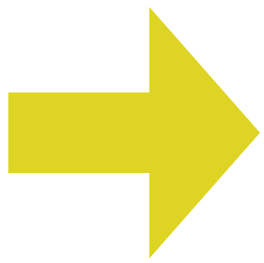
```c
/*** Constants ***/
...
#define ClassVar 10
...
```

C code

CodeGen: types

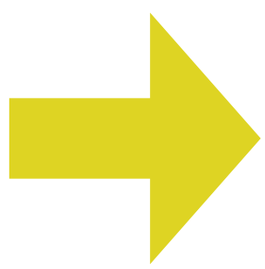



```
xtype
        | x |
        <var: #x type: 'void*'>
        ^ x at: 5
```



```
static sqInt
xtype(void)
{
    void*x;

  return x[5];
}
```

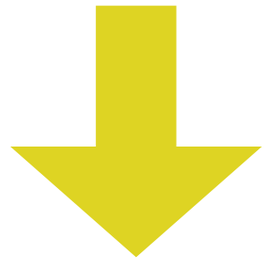
```
returnType

        <returnTypeC: 'char*'>

        ^ 'aaaaa'
```



```
static char*
returnType(void)
{
  return "aaaaa";
}
```

CodeGen: declarations

```
xWithDecl
        | x |
        <var: #x declareC: 'void* x = 0'>
        ^ x
```

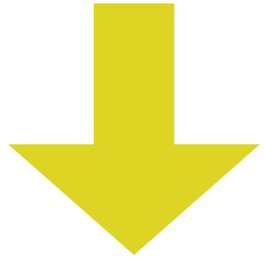


```
static sqInt
xWithDecl(void)
{
    void* x = 0;

  return x;
}
```

CodeGen: declarations for ivars

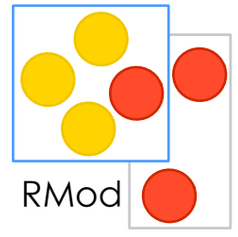
```
InterpreterPlugin subclass: #MyPlugin
        instanceVariableNames: 'typedVar'
```

```
MyPlugin class>>declareCVarsIn: aCCodeGenerator
        aCCodeGenerator
                var: #typedVar type: 'int []'.
```



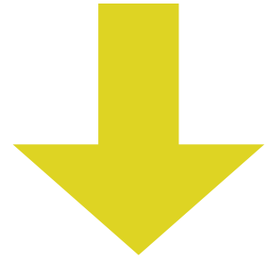
```
/*** Variables ***/
int [] typedVar;
```

CodeGen: inlining



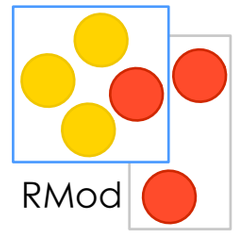

```
inlinedBuggy: x
        <inline: true >
        ^ x + 5
```

```
methodWithInlinedBuggy
        <inline: false>
    ^ self inlinedBuggy: 10
```



```
static sqInt
methodWithInlinedBuggy(void)
{
  return 10 + 5;
}
```

Code generator summary

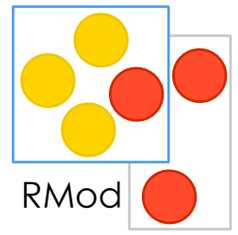


- once you get used to it, it won't cause you much trouble
- you can always extend or improve it

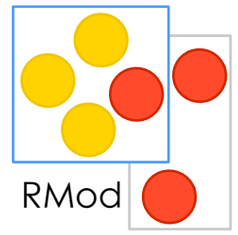
- C compiler will help you to verify generated code :)
- a run-time bugs are much more harder to fix than compile time errors. But what you expected from C?

Code Generation: Questions?

Let **you** compile the VM



<Git's Ideology rant>



with svn, you do something
then to make it public you commit
if this is wrong, you looked like an idiot

with git, you fork
then this is public and people can choose to use it or not
You are not an idiot because you do not damage others work

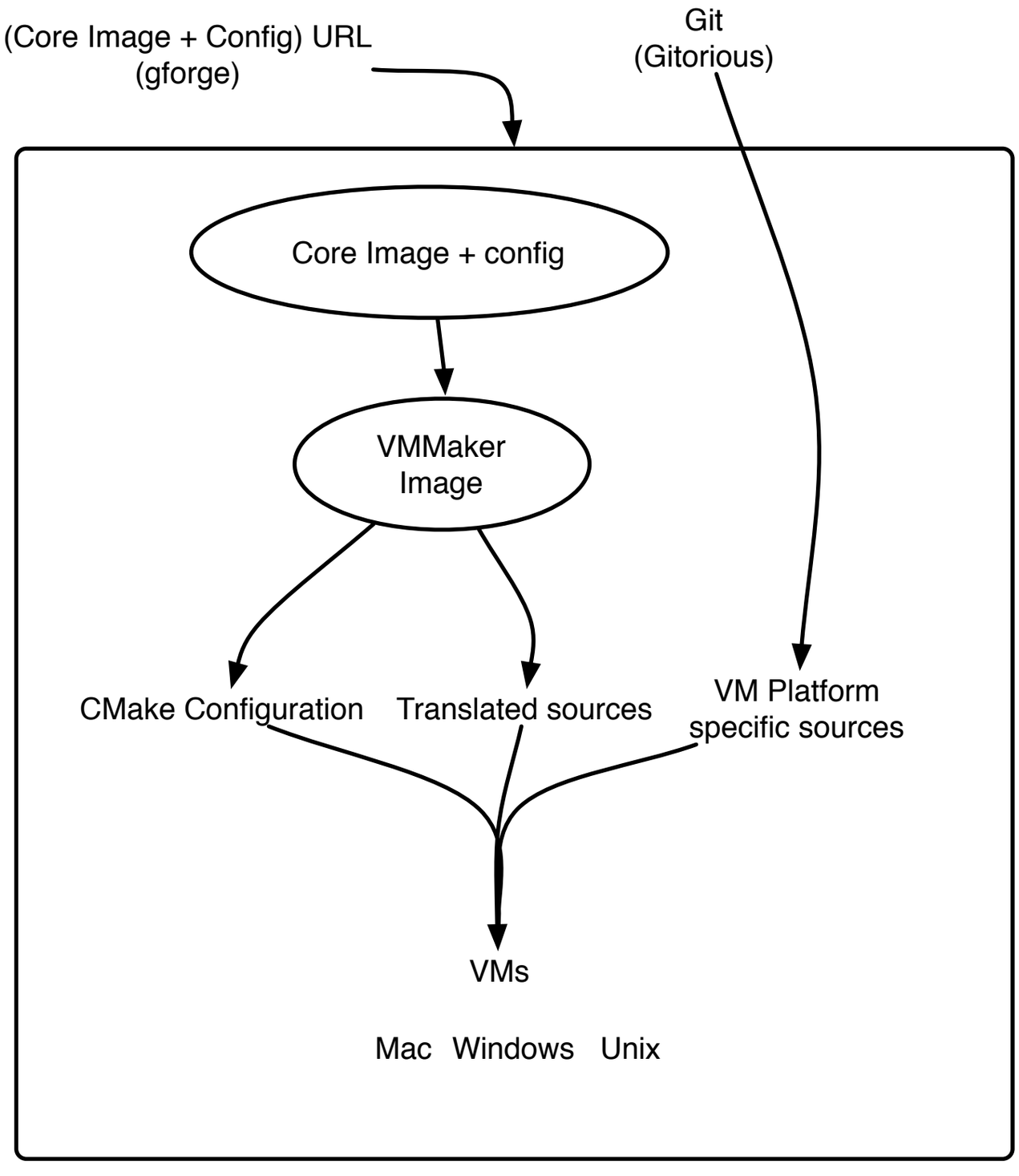
Join gitorious!

Image + VMMaker
Git
CMake

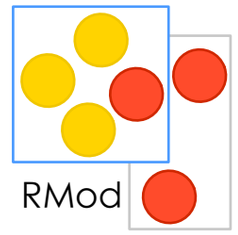
xcode / gcc

git clone
  --depth 1
  http://gitorious.org/~abrabapupa/cogvm/sig-cog.git
  cogvm

Automated build system

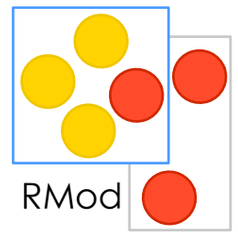


Preparing VMMaker image



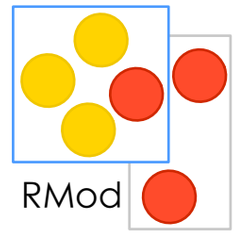

```
Gofer new
        squeaksource: 'MetacelloRepository';
        package: 'ConfigurationOfCog';
        load.
(ConfigurationOfCog project version: '1.3') load
```

Loading platform-specific source code




```
git clone
  --depth 1
  git://gitorious.org/~abrabapupa/cogvm/sig-cog.git
  cogvm
```

Using gitorious



Create an account
Add ssh key
Clone cogvm blessed
Clone your clone to your machine:

```
git clone
  --depth 1
  git://gitorious.org/~johndoe/cogvm/johndoe-cogvm.git
  johndoe-cogvm
```

GITORIOUS

Activities Projects Teams

ducasse stef-squeakvm

squeak-vm ⇒ stef-squeakvm

Clone & push urls | GIT | HTTP | SSH | git@gitorious.org:~ducasse/squeak-vm/stef-squeak ?

Adding this repository as a pushable origin:

```
git remote add origin git@gitorious.org:~ducasse/squeak-vm/stef-squeakvm.git
# to push the master branch to the origin remote we added above:
git push origin master
# after that you can just do:
git push
```

Cloning this repository:

```
git clone git://gitorious.org/~ducasse/squeak-vm/stef-squeakvm.git stef-
squeakvm
cd stef-squeakvm
```

Add this repository as a remote to an existing local repository:

```
git remote add stef-squeakvm git://gitorious.org/~ducasse/squeak-vm/stef-
squeakvm.git
git fetch stef-squeakvm
git checkout -b my-local-tracking-branch stef-
squeakvm/master_or_other_branch
```

Branches: master

Clone of: blessed

Commit log | Source tree | Merge requests (0) | Clone repository | Unwatch

Project: squeak-vm
Owner: ~ducasse
Clone of: squeak-vm/blessed.git
Created: 02 Feb 15:45

Clone repository

Request merge

Manage collaborators

Edit repository

Committers

ducasse (creator)

Repository clones

No clones on Gitorious yet of this repository

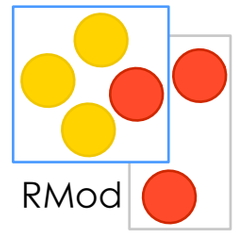
Activities

Wednesday February 02 2011

REPOSITORY 15:45 **ducasse** cloned squeak-vm/blessed

New repository is in stef-squeakvm

Source code directory structure



"platforms root dir"

platforms/

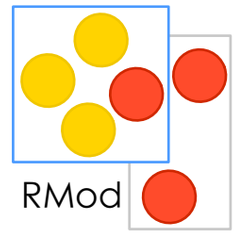
- Cross/ — common code for all platforms
- Mac OS/
- RiscOS/
- unix/
- win32/

— platform-specific source code & configuration files/tools

src/

— VMMaker generated code

Loading CMakeVMMaker




```
Gofer new
        squeaksource: 'VMMaker';
        package: 'CMakeVMMaker';
        load.
```

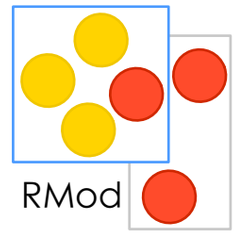
Generating source code + makefiles

<MyConfiguration> generateWithSources

Depends what you building:

- CPlatformConfig
  - CUnixConfig
    - CogUnixConfig
      - CogDebugUnixConfig
      - CogUnixNoGLConfig
      - StackInterpreterUnixConfig
        - StackInterpreterDebugUnixConfig
          - FixedVerSIDebugUnixConfig
- MacOSConfig
  - CocoaIOSConfig
    - CocoaIOSCogConfig
      - CocoaIOSCogJitConfig
      - CocoaIOSCogStackConfig
  - CogMacOSConfig
    - StackInterpreterMacOSConfig

Automatic Jenkins build



Jenkins does that:

export SQUEAKVM=<path to your existing vm>
sh ./buildImage.sh -headless
sh ./generate.sh -headless <ConfigurationName>

Building VM

cd build

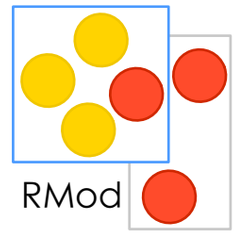
cmake .

make

.. run ..

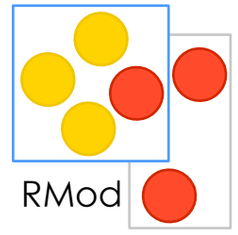
```
/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/video/slice.c: In function 'mpeg3slice_getbit':
/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/video/slice.h:87: warning: control reaches end of non-void functio
[ 99%] Building C object Mpeg3Plugin/CMakeFiles/Mpeg3Plugin.dir/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/aud
/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/audio/synthesizers.c: In function 'mpeg3audio_synth_stereo':
/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/audio/synthesizers.c:63: warning: implicit declaration of function
[ 99%] Building C object Mpeg3Plugin/CMakeFiles/Mpeg3Plugin.dir/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/aud
/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/audio/tables.c: In function 'mpeg3audio_new_decode_tables':
/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/audio/tables.c:504: warning: implicit declaration of function 'mpe
[100%] Building C object Mpeg3Plugin/CMakeFiles/Mpeg3Plugin.dir/Users/sig/projects/cog/tests/cogvm/platforms/Cross/plugins/Mpeg3Plugin/libmpeg/vid
[100%] Building C object Mpeg3Plugin/CMakeFiles/Mpeg3Plugin.dir/Users/sig/projects/cog/tests/cogvm/platforms/Mac_OS/plugins/Mpeg3Plugin/sqMacFileB
In file included from /Users/sig/projects/cog/tests/cogvm/platforms/Mac OS/plugins/Mpeg3Plugin/sqMacFileBits.c:10:
/Users/sig/projects/cog/tests/cogvm/platforms/Mac OS/plugins/Mpeg3Plugin/sqMacFileBits.h: In function 'sqFilenameFromStringOpen':
/Users/sig/projects/cog/tests/cogvm/platforms/Mac OS/plugins/Mpeg3Plugin/sqMacFileBits.h:14: warning: passing argument 2 of 'interpreterProxy->ioF
nter from integer without a cast
Linking C shared library ../StackVM.app/Contents/Resources/libMpeg3Plugin.dylib
[100%] Built target Mpeg3Plugin
sig@lokernaere: ~/projects/cog/tests/cogvm/build $
```

Primitive: what is a primitive?



- it is a magic we do, when smalltalk is not enough :)

whatIsAPrimitive



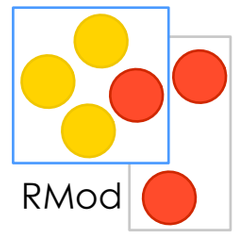
"Some messages in the system are responded to primitively. A primitive response is performed directly by the interpreter rather than by evaluating expressions in a method. The methods for these messages indicate the presence of a primitive response by including <primitive: xx> before the first expression in the method.

Primitives exist for several reasons. Certain basic or 'primitive' operations cannot be performed in any other way. Smalltalk without primitives can move values from one variable to another, but cannot add two SmallIntegers together. Many methods for arithmetic and comparison between numbers are primitives. Some primitives allow Smalltalk to communicate with I/O devices such as the disk, the display, and the keyboard. Some primitives exist only to make the system run faster; each does the same thing as a certain Smalltalk method, and its implementation as a primitive is optional.

When the Smalltalk interpreter begins to execute a method which specifies a primitive response, it tries to perform the primitive action and to return a result. If the routine in the interpreter for this primitive is successful, it will return a value and the expressions in the method will not be evaluated. If the primitive routine is not successful, the primitive 'fails', and the Smalltalk expressions in the method are executed instead. These expressions are evaluated as though the primitive routine had not been called.

The Smalltalk code that is evaluated when a primitive fails usually anticipates why that primitive might fail. If the primitive is optional, the expressions in the method do exactly what the primitive would have done (See Number @). If the primitive only works on certain classes of arguments, the Smalltalk code tries to coerce the argument or appeals to a superclass to find a more general way of doing the operation (see SmallInteger +). If the primitive is never supposed to fail, the expressions signal an error (see SmallInteger asFloat).

Primitives: Defining primitive



To declare a primitive, use export directive:

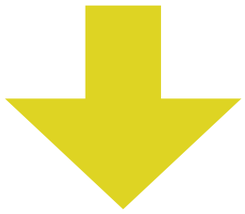
```
primitiveFoo

            <export: true>

...
```

```
EXPORT(sqInt) primitiveFoo(void)
{
...
```

Primitives: declare and use

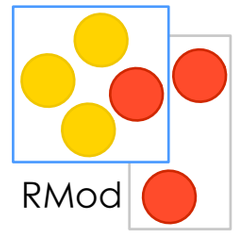
- to use a primitive, use name of an exported method and a module name where it is defined:

```
MyPlugin>>primitiveFoo
 <export: true>
```



```
MyClass>> callPrimitiveFoo: param
  <primitive:'primitiveFoo' module:'MyPlugin'>
self primitiveFailed
```

Primitives



- a primitive function takes no arguments
- a return value of function is not used

So, how to access the arguments passed to primitive?
And how to pass a resulting value?

```
MyClass >> primitiveFoo: bar
  <primitive: ‘primitiveFoo’ module: ‘MyPlugin’>
  self primitiveFailed
```

Primitives: accessing arguments

```
MyClass >> primitiveFoo: bar
  <primitive: ‘primitiveFoo’ module: ‘MyPlugin’>
  self primitiveFailed
```

```
primitiveFoo
<export: true>
 | bar receiver |

bar := self stackValue: 0.
receiver := self stackValue: 1.
```

The top of the stack is last method’s argument, or receiver if none.

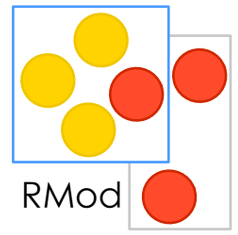
Primitives: answering result

```
MyClass >> primitiveFoo: bar
  <primitive: ‘primitiveFoo’ module: ‘MyPlugin’>
  self primitiveFailed
```

```
primitiveFoo
<export: true>
 | result |

result := ....
self pop: 2 thenPush: result
```

MAKE SURE STACK IS BALANCED
(count receiver too)!!!

Primitive: getting number of args



In Interpreter:

argumentCount

In plugin:

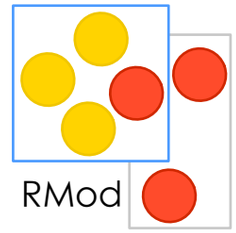
argCnt := interpreterProxy argumentCountOf:
interpreterProxy primitiveMethod.

Failing a primitive

```
MyClass >> primitiveFoo: integer
  <primitive: ‘primitiveFoo’ module: ‘MyPlugin’>
  self primitiveFailed
```

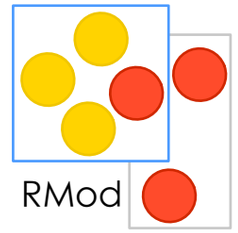
```
primitiveFoo
<export: true>
 | integerOop |

integerOop := self stackValue: 0.
(self isIntegerObject: integerOop) ifFalse: [ ^ self
primitiveFail ]
```

Failing a primitive: a new way




```
primitiveFoo
<export: true>
....
self somethingWrong ifTrue: [
^ self primitiveFailFor: anErrorCode ]
...
```

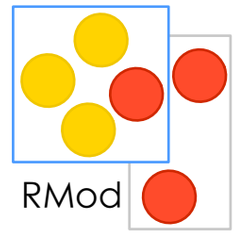
Why error codes is better?




```
basicAt: index
		"Primitive. Assumes receiver is indexable. Answer the value of an
		indexable element in the receiver. Fail if the argument index is not an
		Integer or is out of bounds. Essential. Do not override in a subclass. See
		Object documentation whatIsAPrimitive."

		<primitive: 60>
		index isInteger ifTrue: [self errorSubscriptBounds: index].
		index isNumber
			ifTrue: [^self basicAt: index asInteger]
			ifFalse: [self errorNonIntegerIndex]
```

We should stop guessing what is gone wrong, because primitive could tells us directly what happen.

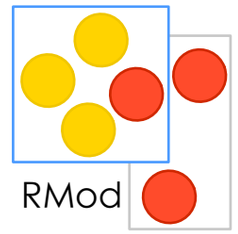
What happens when prim fails?



The interpreter activates the method which contains a primitive and start interpreting it as normal method.

```
MyClass >> primitiveFoo: integer
  <primitive: ‘primitiveFoo’ module: ‘MyPlugin’>


  self primitiveFailed
```

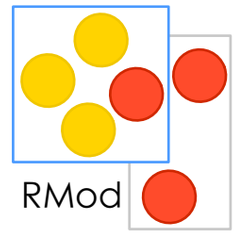
What happens when prim NOT fails?



The interpreter continues with evaluating the caller context.

```
MyClass >> primitiveFoo: integer
  <primitive: ‘primitiveFoo’ module: ‘MyPlugin’>


  self primitiveFailed
```

Primitives: numbered vs named

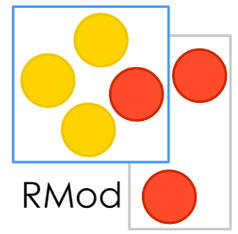


numberedOne
  <primitive: 100>

namedOne
  <primitive: ‘name’ module: ‘moduleName’>

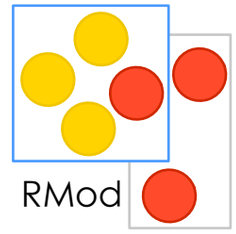
- Numbered primitives can reside only in Interpreter.
- To find implementation of a numbered prim, refer to Interpreter class>>***initializePrimitiveTable***

PrimitiveSomeInstance




```
Interpreter>>primitiveSomeInstance
        | class instance |
        class := self stackTop.
        instance := self initialInstanceOf: class.
        instance = nilObj
            ifTrue: [self primitiveFail]
            ifFalse: [self pop: argumentCount+1 thenPush: instance]
```

Primitives: What, when and why?



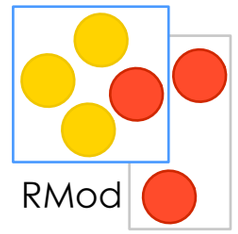
Rule #1: Primitive should be kept primitive!

I.E. Don't make primitives too clever.

Leave complex logic at language side

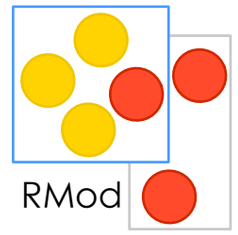
```
primSameClassAbsoluteStrokeDistanceMyPoints: myPointsOop otherPoints:
otherPointsOop myVectors: myVectorsOop otherVectors: otherVectorsOop
mySquaredLengths: mySquaredLengthsOop otherSquaredLengths:
otherSquaredLengthsOop myAngles: myAnglesOop otherAngles: otherAnglesOop
maxSizeAndReferenceFlag: maxSizeAndRefFlag rowBase: rowBaseOop rowInsertRemove:
rowInsertRemoveOop rowInsertRemoveCount: rowInsertRemoveCountOop
                | base insertRemove jLimiT substBase insert remove subst removeBase
insertBase insertRemoveCount additionalMultiInsertRemoveCost myPoints otherPoints
myVectors otherVectors rowInsertRemoveCount mySquaredLengths
otherSquaredLengths myAngles otherAngles rowBase rowInsertRemove otherPointsSize
myVectorsSize otherVectorsSize otherSquaredLengthsSize rowBaseSize maxDist maxSize
forReference jM1 iM1 iM1T2 jM1T2 |
                self var: #myPoints type: 'int *  '.
                self var: #otherPoints type: 'int *  '.
                self var: #myVectors type: 'int *  '.
                self var: #otherVectors type: 'int *  '.
                self var: #mySquaredLengths type: 'int *  '.
                self var: #otherSquaredLengths type: 'int *  '.
                self var: #myAngles type: 'int *  '.
                self var: #otherAngles type: 'int *  '.
                self var: #rowBase type: 'int *  '.
                self var: #rowInsertRemove type: 'int *  '.
                self var: #rowInsertRemoveCount type: 'int *  '.
```

Genie plugin continued..



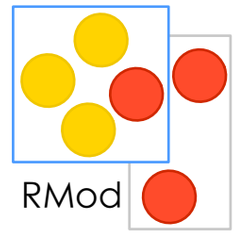

```
		self
			primitive:
'primSameClassAbsoluteStrokeDistanceMyPoints_otherPoints_myVectors_otherVectors_mySquaredLengths_otherSquaredLengths_myAngles_otherAngle
s_maxSizeAndReferenceFlag_rowBase_rowInsertRemove_rowInsertRemoveCount'
			parameters: #(#Oop #Oop #Oop #Oop #Oop #Oop #Oop #Oop #SmallInteger #Oop #Oop #Oop)
			receiver: #Oop.
		interpreterProxy failed
			ifTrue: [self msg: 'failed 1'.
				^ nil].

		interpreterProxy success: (interpreterProxy isWords: myPointsOop)
				& (interpreterProxy isWords: otherPointsOop)
				& (interpreterProxy isWords: myVectorsOop)
				& (interpreterProxy isWords: otherVectorsOop)
				& (interpreterProxy isWords: mySquaredLengthsOop)
				& (interpreterProxy isWords: otherSquaredLengthsOop)
				& (interpreterProxy isWords: myAnglesOop)
				& (interpreterProxy isWords: otherAnglesOop)
				& (interpreterProxy isWords: rowBaseOop)
				& (interpreterProxy isWords: rowInsertRemoveOop)
				& (interpreterProxy isWords: rowInsertRemoveCountOop).
	interpreterProxy failed
		ifTrue: [self msg: 'failed 2'.
				^ nil].
	interpreterProxy success: (interpreterProxy is: myPointsOop MemberOf: 'PointArray')
				& (interpreterProxy is: otherPointsOop MemberOf: 'PointArray').
	interpreterProxy failed
		ifTrue: [self msg: 'failed 3'.
				^ nil].
	myPoints := interpreterProxy firstIndexableField: myPointsOop.
	otherPoints := interpreterProxy firstIndexableField: otherPointsOop.
	myVectors := interpreterProxy firstIndexableField: myVectorsOop.
	otherVectors := interpreterProxy firstIndexableField: otherVectorsOop.
	mySquaredLengths := interpreterProxy firstIndexableField: mySquaredLengthsOop.
	otherSquaredLengths := interpreterProxy firstIndexableField: otherSquaredLengthsOop.
	myAngles := interpreterProxy firstIndexableField: myAnglesOop.
	otherAngles := interpreterProxy firstIndexableField: otherAnglesOop.
	rowBase := interpreterProxy firstIndexableField: rowBaseOop.
	rowInsertRemove := interpreterProxy firstIndexableField: rowInsertRemoveOop.
```

Genie plugin



and it was only the initialization and argument checking....

Primitives: What, when and why?



Q: Use for heavy byte crunching?
A: less and less relevant. With JIT (and NativeBoost), you can achieve same without touching VM.

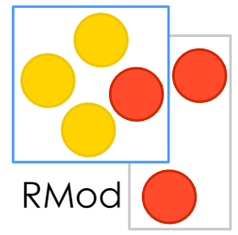
Q: to improve VM?
A: yes. But don't violate rule #1.

Q: to communicate with some 3rd party library to access some missing/necessary functionality?
A: yes. This is the right choice.

Primitives: Questions?

RMod

Plugin

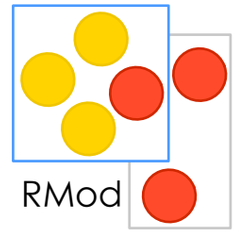


- a way to extend VM

- can be made external or internal

- implement by creating a subclass of InterpreterPlugin or SmartSyntaxInterpreterPlugin

Plugin: important details



moduleName

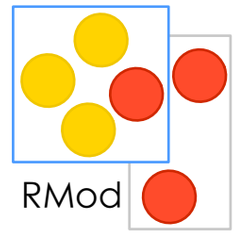
- answer a string to use for plugin name
- default is just class name of plugin

- used by code generation:
  src/plugins/<ModuleName>/<ModuleName>.c

- used to identify a plugin at run-time:
  InflatePlugin>>primitiveInflateDecompressBlock

  <primitive: 'primitiveInflateDecompressBlock' module: '**ZipPlugin**'>

InterpreterProxy: Plugin vs Core



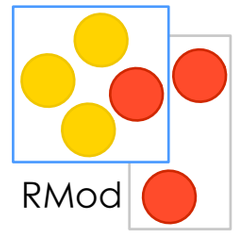
Primitive written for Interpreter

```
primitiveByteSize
    <export: true>
    | oop |
    oop := self stackValue: 0.

    self pop: 1 thenPush:
    (self integerObjectOf:(self byteSizeOf: oop))
```

Same primitive, but in plugin

```
primitiveByteSize
    <export: true>
    | oop |
    oop := interpreterProxy stackValue: 0.

    interpreterProxy pop: 1 thenPush:
    (interpreterProxy integerObjectOf:(interpreterProxy byteSizeOf: oop))
```

Interpreter vs Plugin



	Interpreter	**Plugin**
receiver for sending message to interpreter	self	interpreterProxy
can access interpreter global state?	yes	no
can use any interpreter method?	yes	only methods defined in InterpreterProxy *

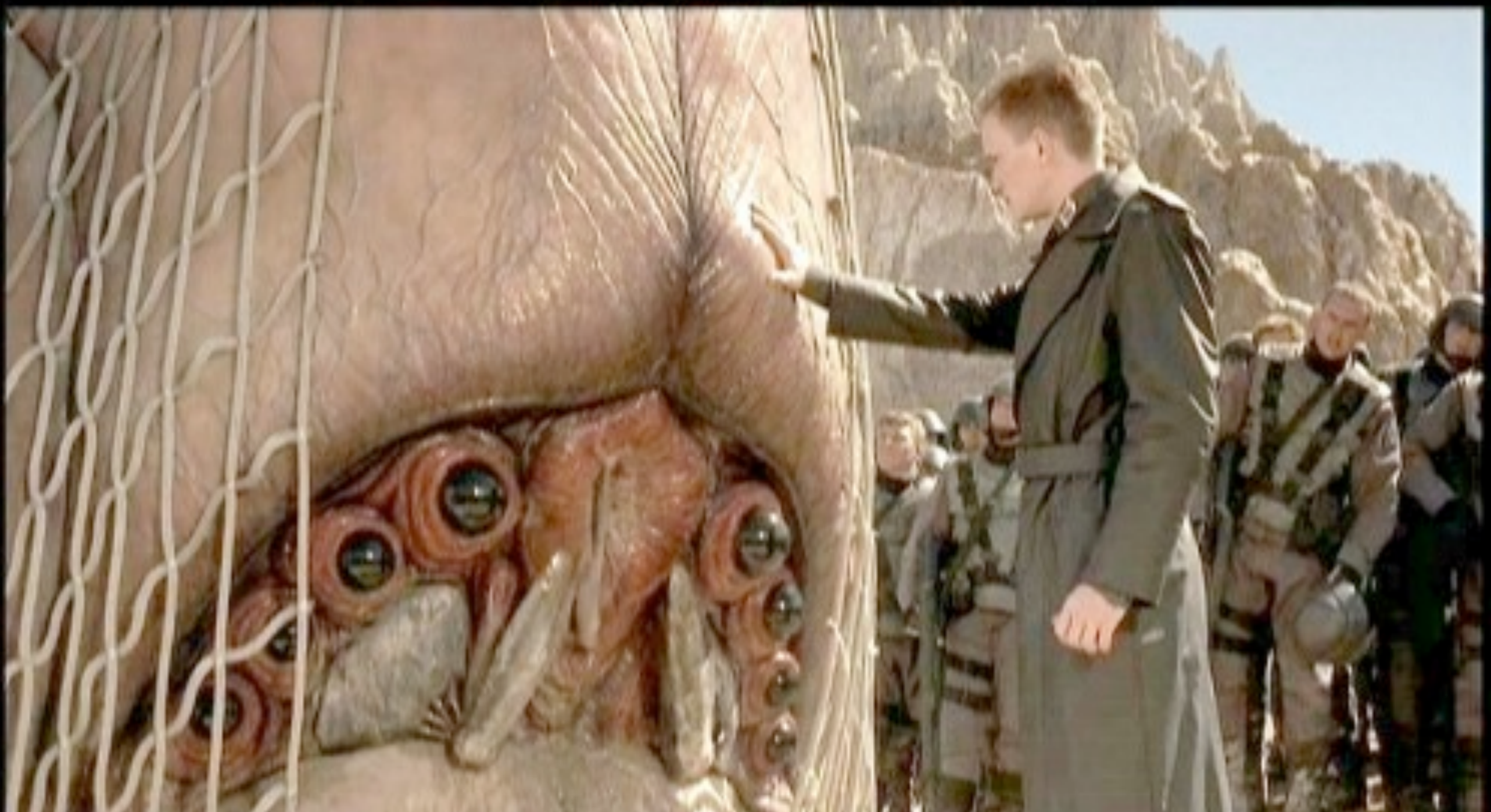
* can use <api> methods, but only by internal plugins (not recommended for use, but we're not living in perfect world ;)

So, lets make our first plugin?

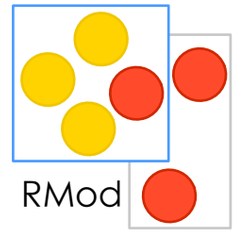
Not so fast. First we need to build VM :)

RMod

Face your enemy!



Building VM: preparing sources




```
git clone --depth 1 git://gitorious.org:~abrabapupa/cogvm/sig-cog.git cogvm
cd cogvm/codegen-scripts
```

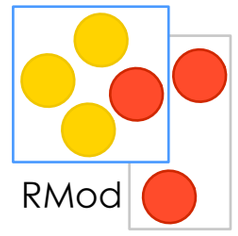
Doing automatically:

```
export SQUEAKVM=<path to your existing vm>
sh ./buildImage.sh -headless
sh ./generate.sh -headless CogUnixConfig
```

Doing manually:
download image from image.url

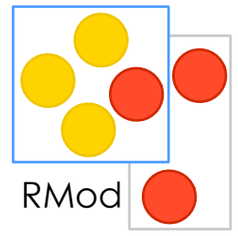
```
Gofer new
        squeaksource: 'MetacelloRepository';
        package: 'ConfigurationOfCog';
        load.

((Smalltalk at: #ConfigurationOfCog) project
        version: '1.3') load.

Gofer new
        squeaksource: 'VMMaker';
        package: 'CMakeVMMaker';
        load.
```

TheUniversalAnswer



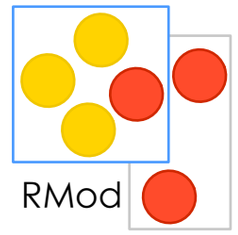

```
InterpreterPlugin subclass: #TheUniversalAnswer
        instanceVariableNames: ''
        classVariableNames: ''
        poolDictionaries: ''
        category: 'JourneyInTheVM'
```

TheUniversalAnswer



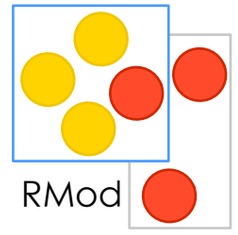

```
primitiveAnswer
        <export: true>

        interpreterProxy pop: 1 thenPush: (
            interpreterProxy integerObjectOf: 42
)
```

Adding plugin to build




```
| config |
config := StackInterpreterMacOSConfig new.

config internalPlugins: (config internalPlugins
copyWith: #TheUniversalAnswer).

config generateSourceFiles.
CMakeVMGenerator generate: config.
```

Building VM



cd build

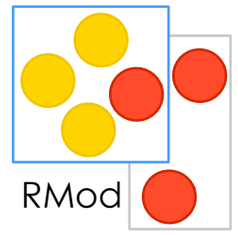
cmake .

grep TheUniversalAnswer CMakeLists.txt

make

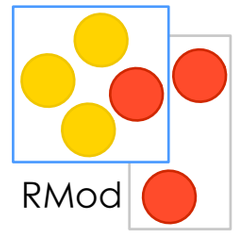
.. run ..

Using the primitive




```
askTheQuestionOfUniverse
        <primitive: 'primitiveAnswer' module: 'TheUniversalAnswer'>
        self primitiveFailed
```

Making your contribution accessible



.. and visible, and easy to follow by others?

Use gitorious!

http://gitorious.org/+squeak-vm-developers