Robustification through introspection and analysis tools.

(Avoiding developer taxes)

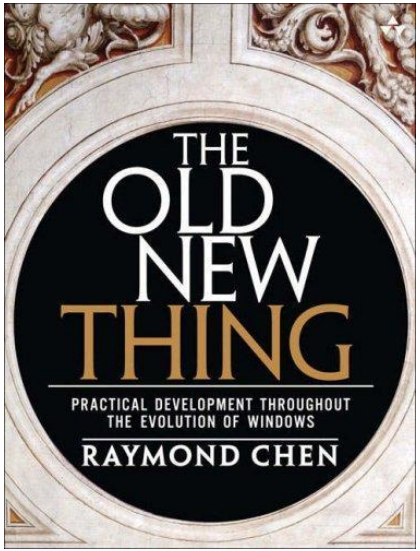
Stephen.Kennedy @ havok.com

Principal Engineer

GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 5-9, 2012
EXPO DATES: MARCH 7-9
2012

Developer Taxes

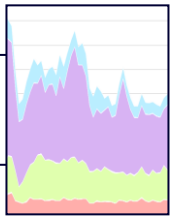
"It's something you do, not because it actually benefits you specifically, but because it benefits the software landscape as a whole." (Raymond Chen)



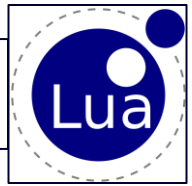
The Classic Game Taxes

Serialization

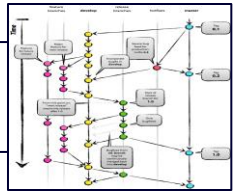
Memory Reporting



Script Binding



Versioning



Memory Lane

10 Years of taxes

Why we changed

Automate

Correctness

Spread the workload

Structure

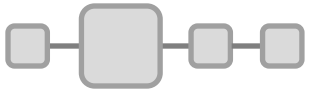
Serialization

Reflection

Memory Reporting

Script Binding

Versioning

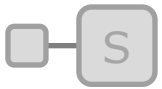


Manual serialization

```
xtream.TokenMustBe("POINT_A"); xtream>>point_A;
bool has_two_bodies = true;
if (xtream.GetVersion() >= 1350)
{
    xtream.TokenMustBe("HAS_TWO_BODIES"); xtream>>has_two_bodies;
}
if (has_two_bodies)
{
    xtream.TokenMustBe("BODY_B"); xtream>>prot_buffer;
    priv_rigid_body_B = prot_subspace->GetRigidBody(prot_buffer);
    if (!priv_rigid_body_B) throw_exception("Rigidbody unknown in Spring");
}
xtream.TokenMustBe("POINT_B"); xtream>>point_B;
//…
```

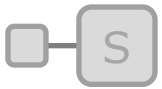
Reflection Recap (1)

```
struct A0 { float x; byte y; }
struct A1 { float z; byte p; byte q };
struct A2 { byte i[3]; }
…
struct A1023 { };
```



Reflection Recap (2)

```
struct A0 { float x; byte y; }
char A0_type[]={ “FB” };
void save_any( void* obj, char* type )
   while(*type) { switch( *type++ ) {
      case ‘F’:// write(); obj += sizeof(float);
      case ‘B’:// write(); obj += sizeof(bool);
      case 0: return; }
```



Serialization with Reflection

Straightforward. (mostly)

The problem now is …

Get Reflected

How to reflect our data?

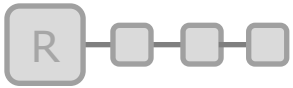
How to keep it up to date?

Robustness

Manual Reflection

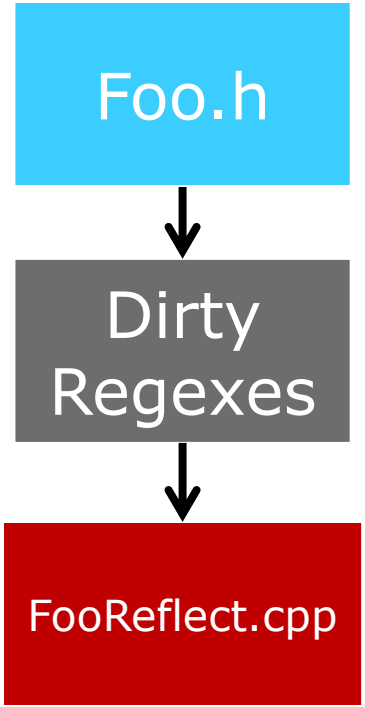
```
class Foo {
  public:
    enum Flags {…};
    int i;
    char* pc;
    double d;
    Flags flags;
  protected:
    long larr[10];
    A* pa;
};
```

```
RTTI_DESCRIBE_CLASS( Foo, (
   RTTI_FIELD(i, RTTI_FLD_PUBLIC),
   RTTI_PTR(pc, RTTI_FLD_PUBLIC),
   RTTI_FIELD(d, RTTI_FLD_PUBLIC),
   RTTI_FIELD(f, RTTI_FLD_PUBLIC),
   RTTI_ARRAY(larr, RTTI_FLD_PROTECTED),
   RTTI_PTR(pa, RTTI_FLD_PROTECTED)
) );
```

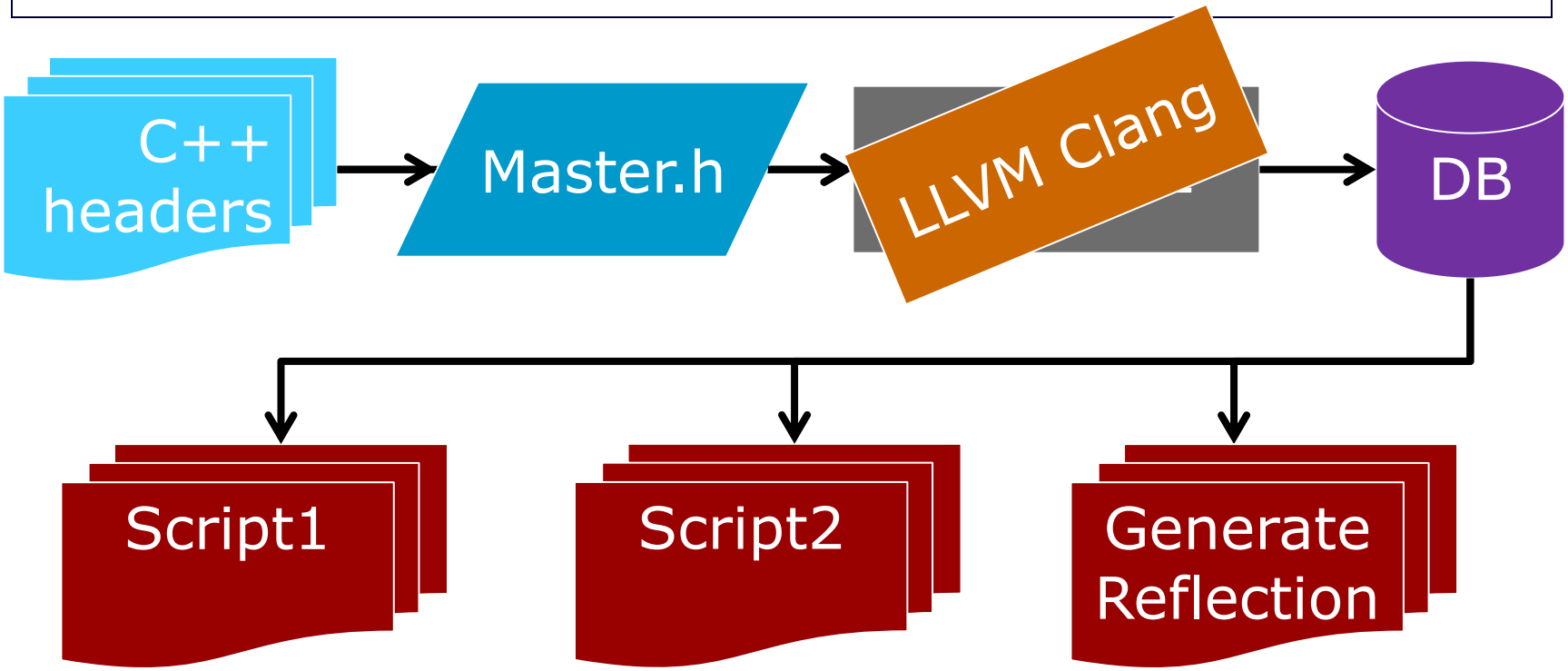


Parsing Headers

```
class Foo {
  public:
    int i;
    char* pc;
    double d;
    enum Flags flags;
  protected:
    long larr[10];
    class B* pb;
    #ifdef PLATFORM_Y
       special* s;
    #endif
};
```



We Went Full Auto



Clang AST Consumer

```
class RawDumpASTConsumer : public ASTConsumer
{
    virtual void Initialize(ASTContext& Context);
    virtual void HandleTopLevelDecl(DeclGroupRef DG)
    {
            // ...
        if( const FieldDecl* fd = dyn_cast<FieldDecl>(declIn) )
            // ...
        else if( const CXXMethodDecl* md = dyn_cast<CXXMethodDecl>(declIn) )
            // ...
        else if( const EnumConstantDecl* ed = dyn_cast<EnumConstantDecl>(declIn) )
            // ...
    }
};
```

Clang Custom Output

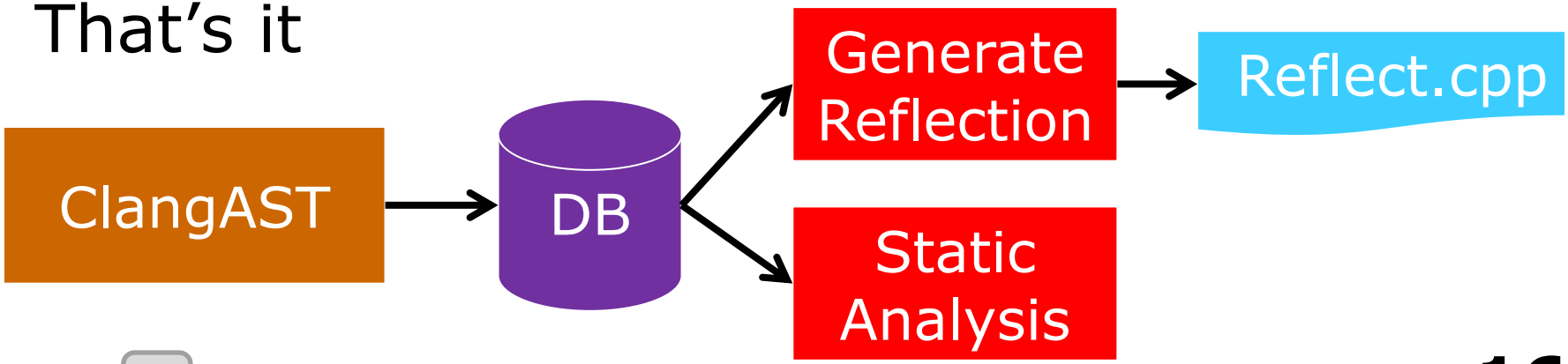
```
File( id=20270, location='Base/Types/Geometry/hkGeometry.h' )

RecordType( id=20271, name='hkGeometry', polymorphic=False,
abstract=False, scopeid=20270 )

Method( id=20317, recordid=20271, typeid=20316,
name='getTriangle' )

Field( id=20320, recordid=20271, typeid=9089,
name='m_vertices' )
```

Build Integration

Prebuild step runs Clang if necessary

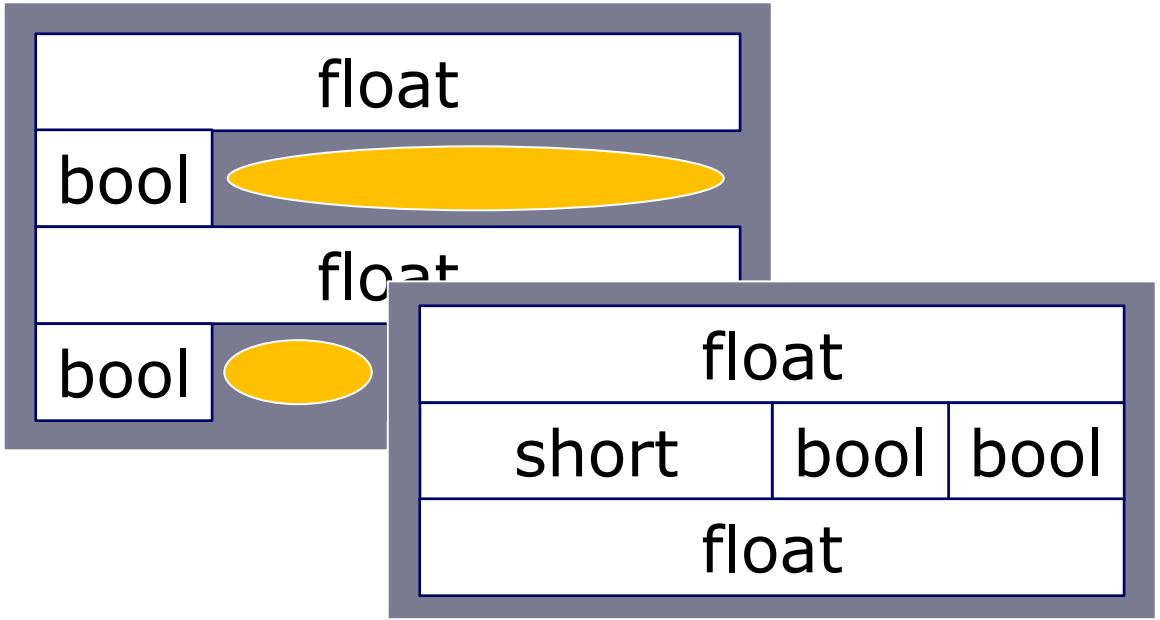
Plugins run on the database

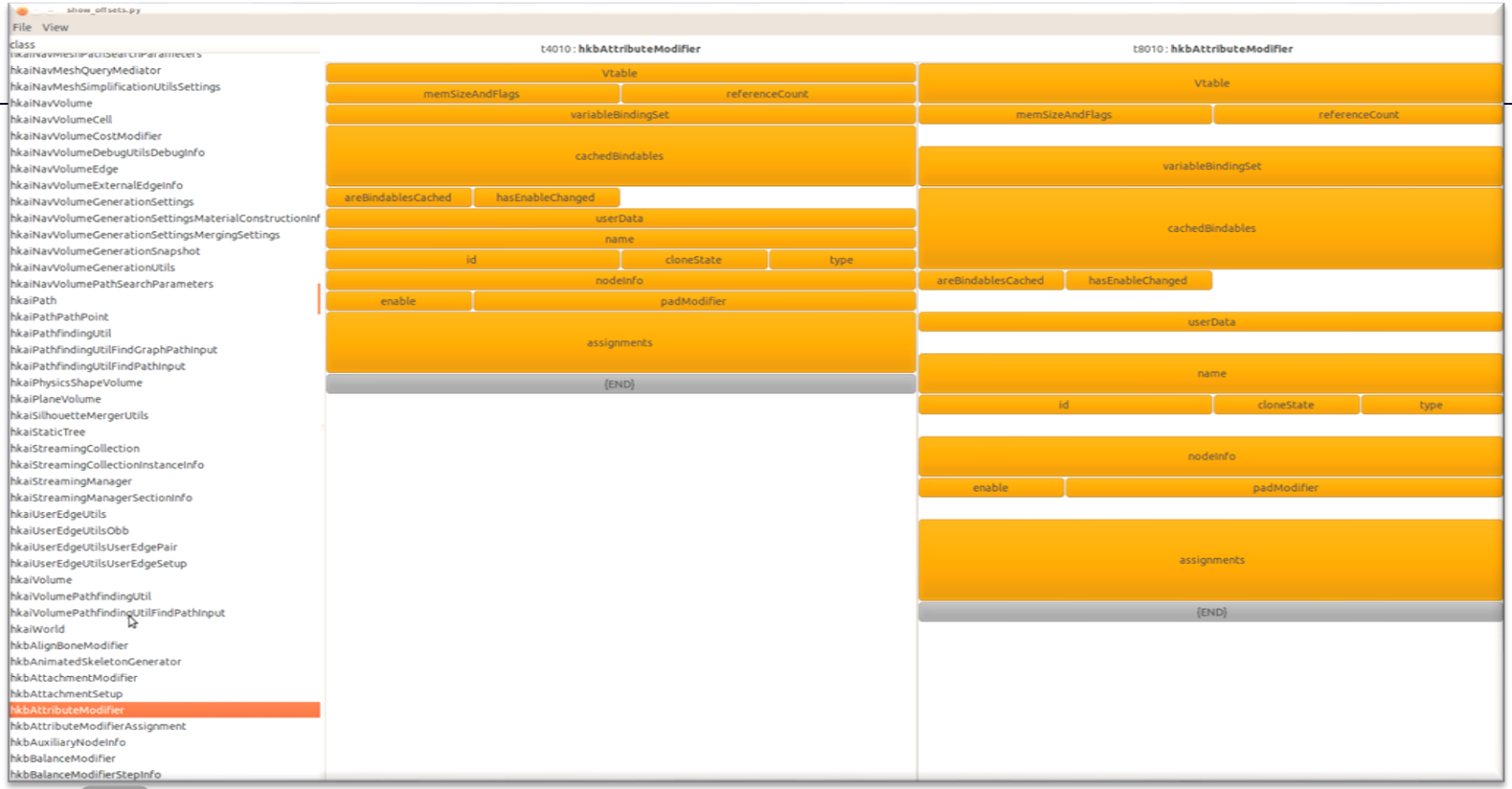
That's it



Runtime Introspection

```
struct X
{
   float time;
   bool used;
   float pos;
   bool canmove;
   short flags;
};
```




File View
class
hkaiNavMeshQueryMediator
hkaiNavMeshSimplificationUtilsSettings
hkaiNavVolume
hkaiNavVolumeCell
hkaiNavVolumeCostModifier
hkaiNavVolumeDebugUtilsDebugInfo
hkaiNavVolumeEdge
hkaiNavVolumeExternalEdgeInfo
hkaiNavVolumeGenerationSettings
hkaiNavVolumeGenerationSettingsMaterialConstructionInf
hkaiNavVolumeGenerationSettingsMergingSettings
hkaiNavVolumeGenerationSnapshot
hkaiNavVolumeGenerationUtils
hkaiNavVolumePathSearchParameters
hkaiPath
hkaiPathPathPoint
hkaiPathfindingUtil
hkaiPathfindingUtilFindGraphPathInput
hkaiPathfindingUtilFindPathInput
hkaiPhysicsShapeVolume
hkaiPlaneVolume
hkaiSilhouetteMergerUtils
hkaiStaticTree
hkaiStreamingCollection
hkaiStreamingCollectionInstanceInfo
hkaiStreamingManager
hkaiStreamingManagerSectionInfo
hkaiUserEdgeUtils
hkaiUserEdgeUtilsObb
hkaiUserEdgeUtilsUserEdgePair
hkaiUserEdgeUtilsUserEdgeSetup
hkaiVolume
hkaiVolumePathfindingUtil
hkaiVolumePathfindingUtilFindPathInput
hkaiWorld
hkbAlignBoneModifier
hkbAnimatedSkeletonGenerator
hkbAttachmentModifier
hkbAttachmentSetup
hkbAttributeModifier
hkbAttributeModifierAssignment
hkbAuxiliaryNodeInfo
hkbBalanceModifier
hkbBalanceModifierStepInfo
t4010 : hkbAttributeModifier
Vtable
memSizeAndFlags
referenceCount
variableBindingSet
cachedBindables
areBindablesCached
hasEnableChanged
userData
name
id
cloneState
type
nodeInfo
enable
padModifier
assignments
{END}
t8010 : hkbAttributeModifier
Vtable
memSizeAndFlags
referenceCount
variableBindingSet
cachedBindables
areBindablesCached
hasEnableChanged
userData
name
id
cloneState
type
nodeInfo
enable
padModifier
assignments
{END}

Reflection Conclusion

LLVM Clang pass

- Robust
- Eliminates out-of-sync errors

DB Consumer pass

- Pre-compile logic checks
- Runtime errors → compile errors

Unit Tests

- Examine reflection data

Language Binding

Expose C++ to script

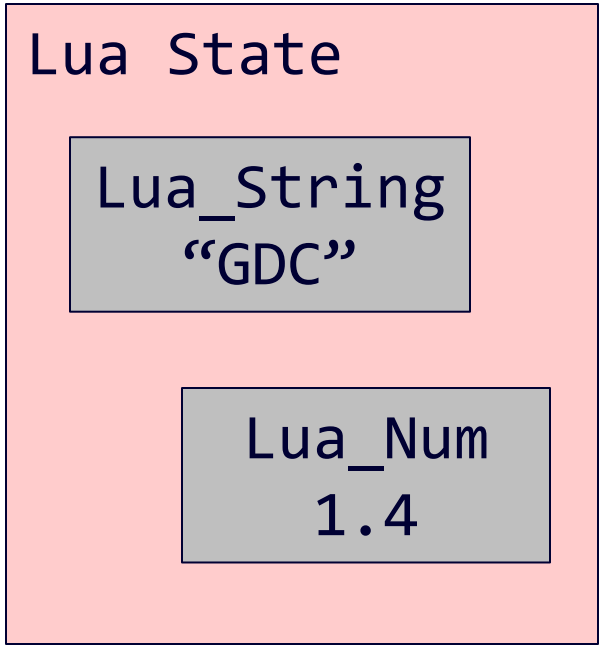
Data: Natural

Callables: Harder

Sample Interface

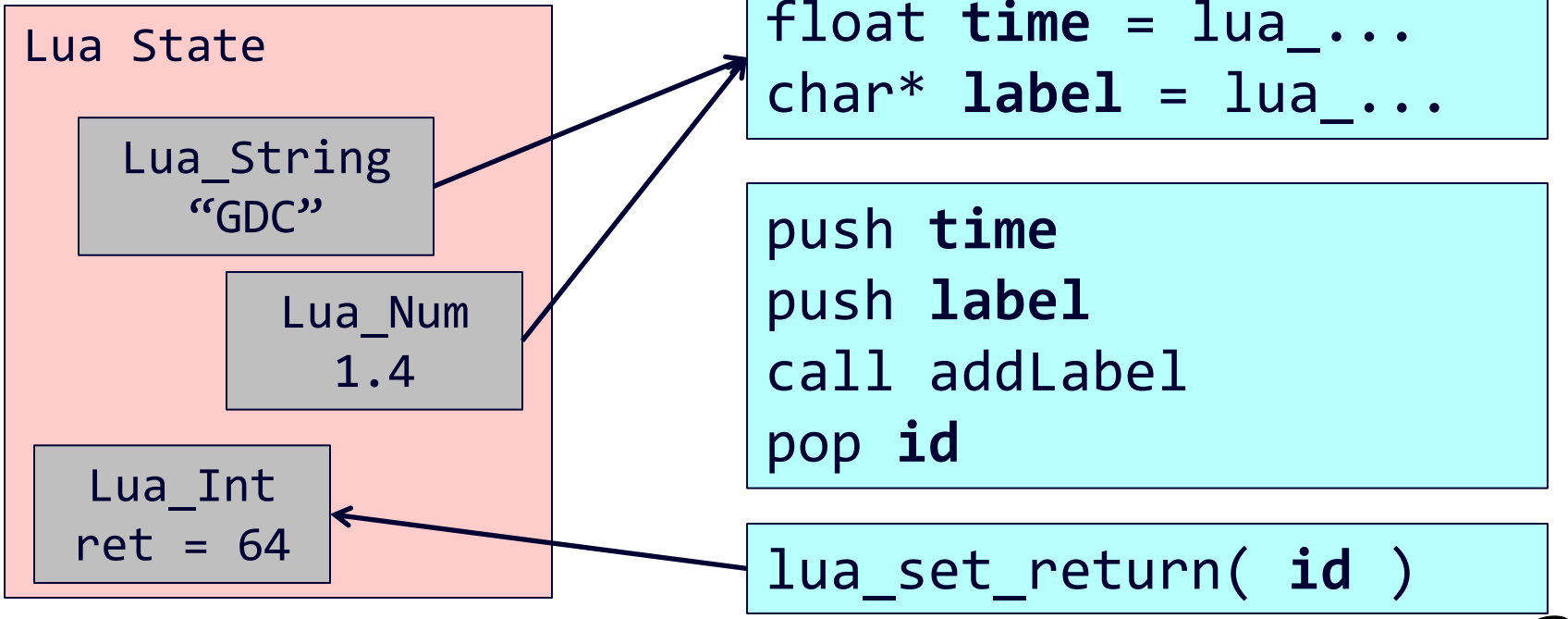
```
class Timeline
{
    /// Adds a label at the given time and
    /// returns an id for that label
  int addLabel( float time, const char* label );
};
```

What is a Binding?




```
int addLabel(
    float time,
    const char* label)
{
    // ...
}
```

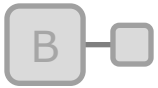
Three Parts of a Binding



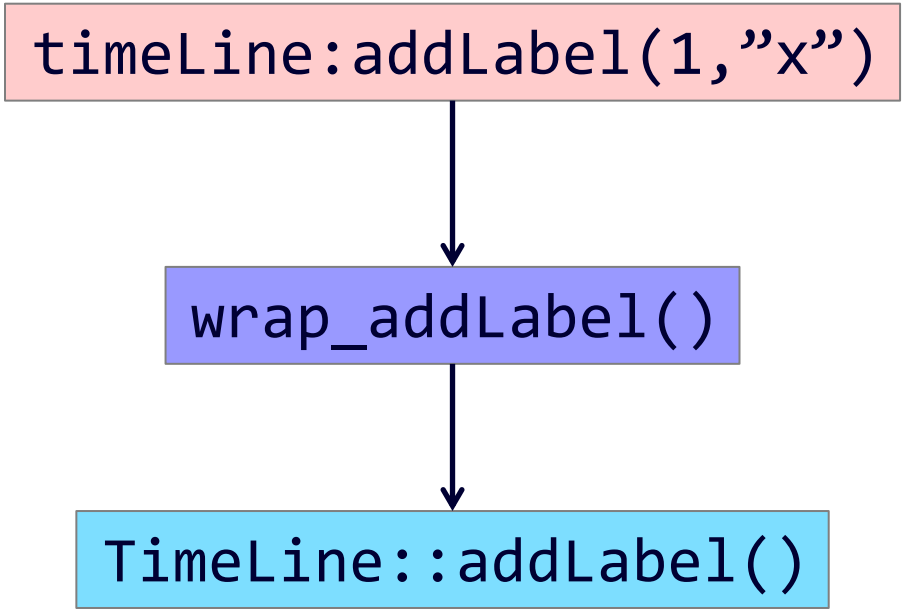
Manual Bindings

```
int wrap_timeLine_addLabel(lua_state* s) {
    TimeLine* tline = lua_checkuserdata(s,1);
    int arg0 = lua_checkint(s,2);
    const char* arg1 = lua_checkstring(s,3);
    int id = tline->addLabel( arg0, arg1 );
    lua_pushint(id);
    return 1;
}
```

```
timeLine:addLabel(1,”x”)
```



“Fat” Bindings

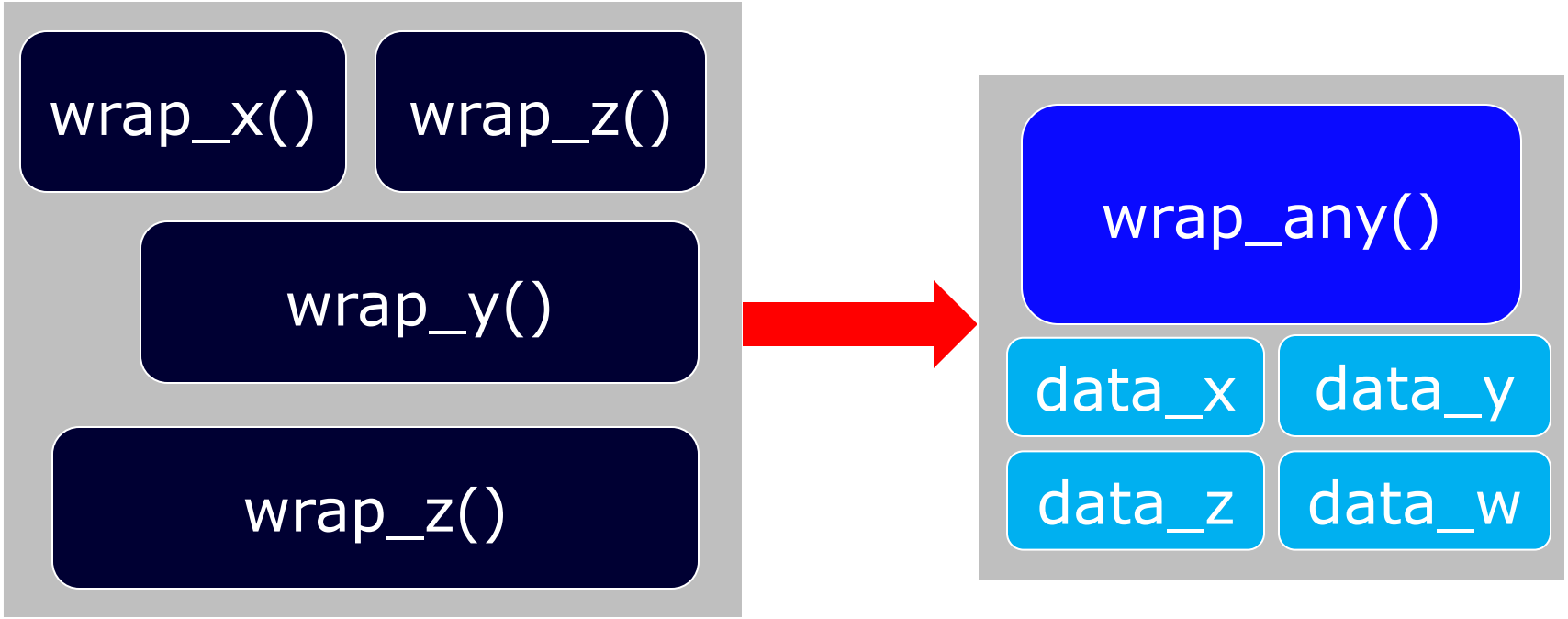


1:1 wrapper:native

Manual or Generated

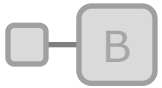
~400b per wrapper

Slimmer Bindings?



Reflected Function

```
struct FunctionReflection {
  const char* name;
  Callable callable; // “function pointer”
  TypeReflection* argTypes;
  int numArgs;  // Including return type
};
```

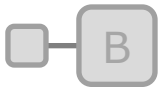


Slimmer Binding

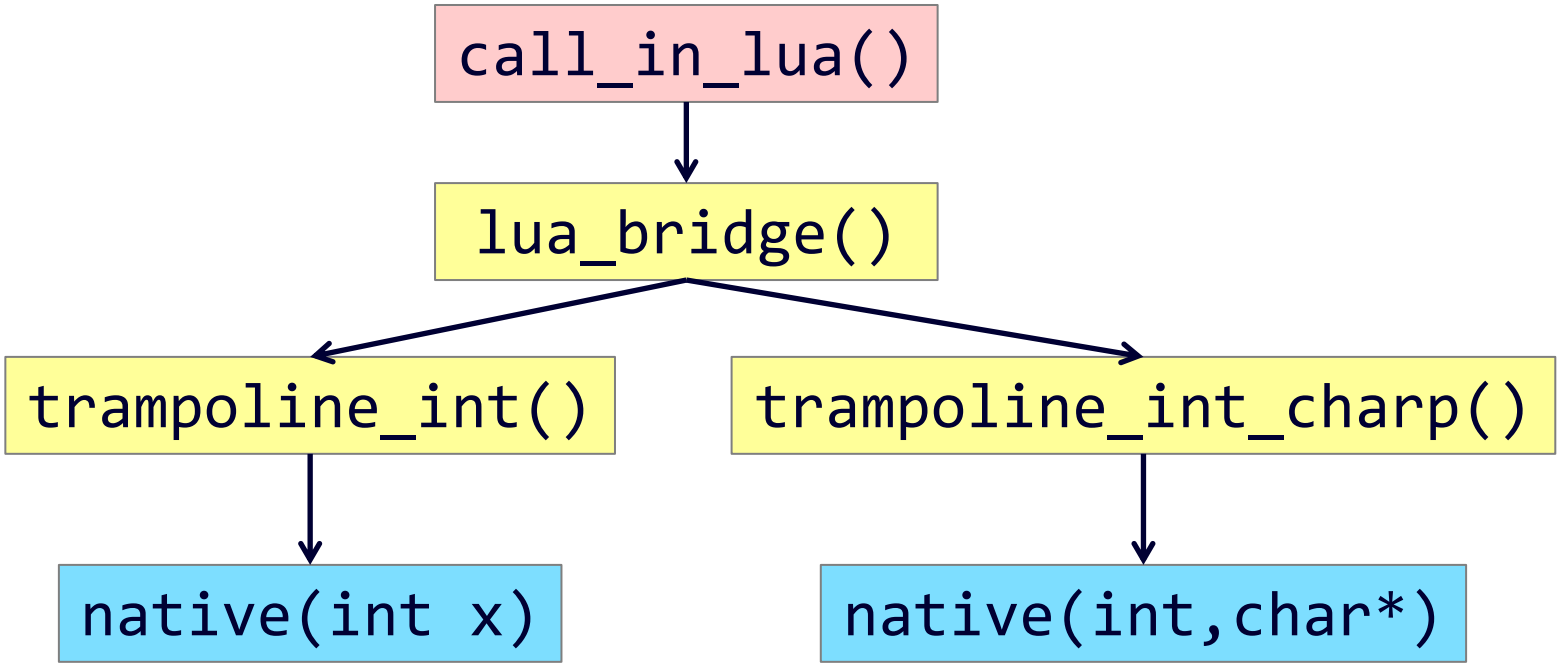
```
int wrap_anything(lua_state* s) {
    FunctionReflection* fr = func_get(s);
    Buffer buf;
    unpack_args(s, fr->argTypes, buf);
    (*fr->callable)(buf);
    return pack_args(s, fr->argTypes, buf);
}
```

Function Trampoline

```
typedef int (*Func_int__int_charp)(int i, char* c);

void trampoline(void** buf, Func_int__int_charp funcptr)
{
    int& ret = *(int*)buf[0];
    int& a0 = *(int*)buf[1];
    char* a1 = *(char**)buf[2];
    ret = (*funcptr)(a0,a1);
}
```

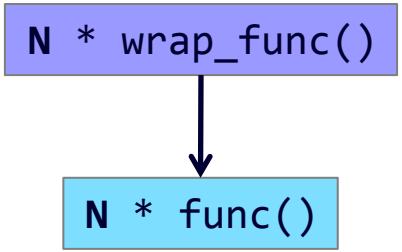


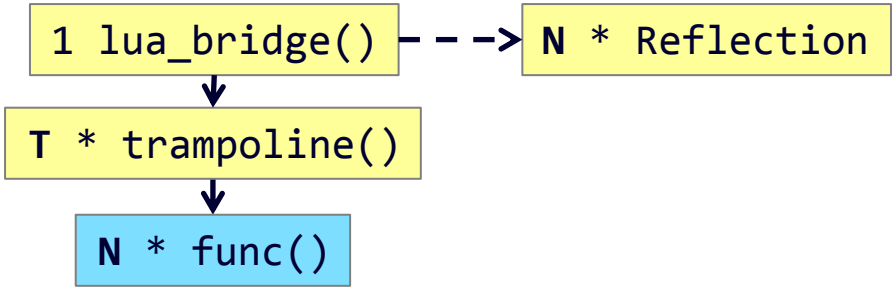
Trampolines



Fat vs Slim Memory Cost

N functions
T distinct trampolines (T≤N)

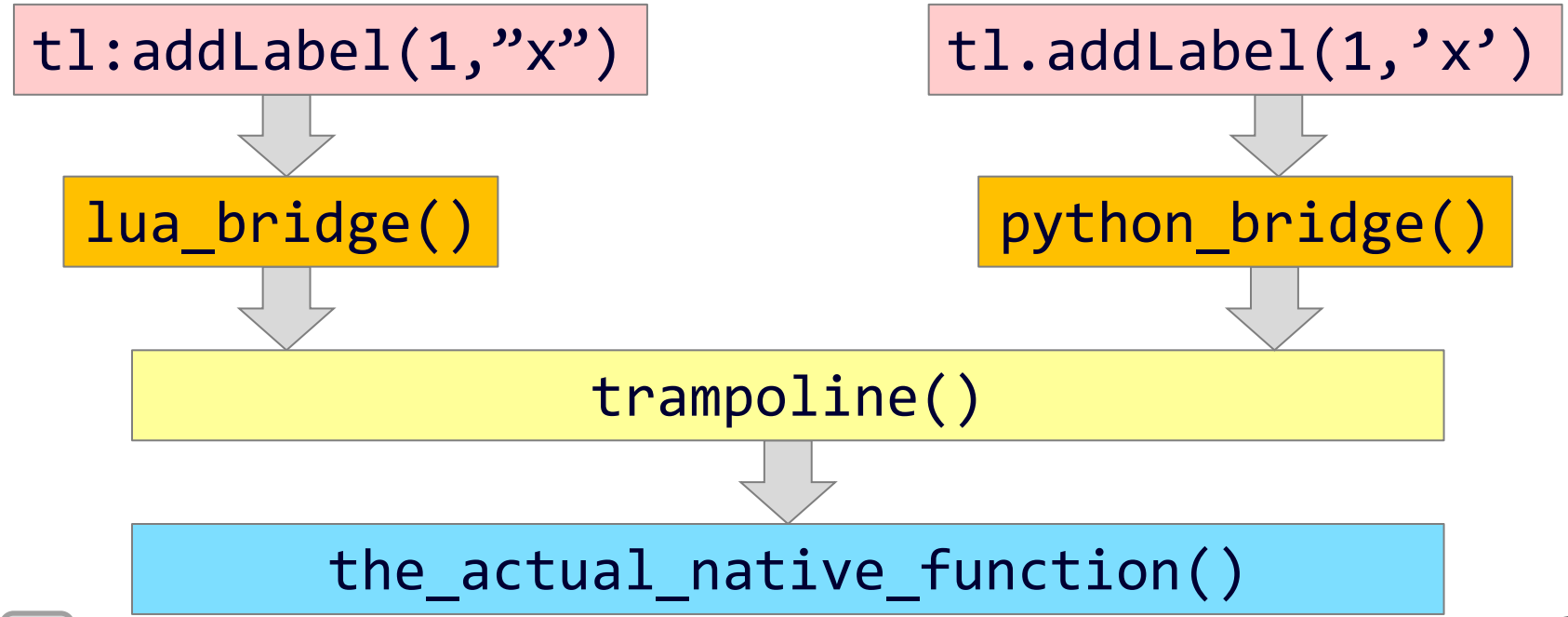




~400*N

~40*N + ~64*T

Sharing the Trampolines



Bindings Conclusion

- Generated “Slim” Bindings
  - Crossplatform & Multilanguage!
- Marginally slower
  - Extra indirection
- Considerably smaller

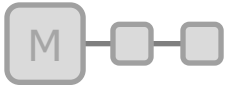
Memory Reporting Goals

More than just block sizes

Account for every byte

Low maintenance burden

Customizable output



Memory Reporting

Manual getMemoryStatistics()

Buggy

Tedious

```
void hkpMeshShape::calcContentStatistics
          ( hkStatisticsCollector* collector ) const
{
  collector->addArray( "SubParts", this->m_subparts );
  for( int i = 0; i < this->m_childInfo.getSize(); i++ )
  {
    collector->addReferencedObject( "Child", m_childInfo[i].m_shape );
  }
  hkpShapeCollection::calcContentStatistics(collector);
}
```

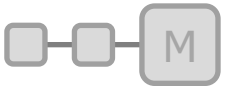
Automatic Reports

Aim for 100% automatic coverage

Provide debugging for missing

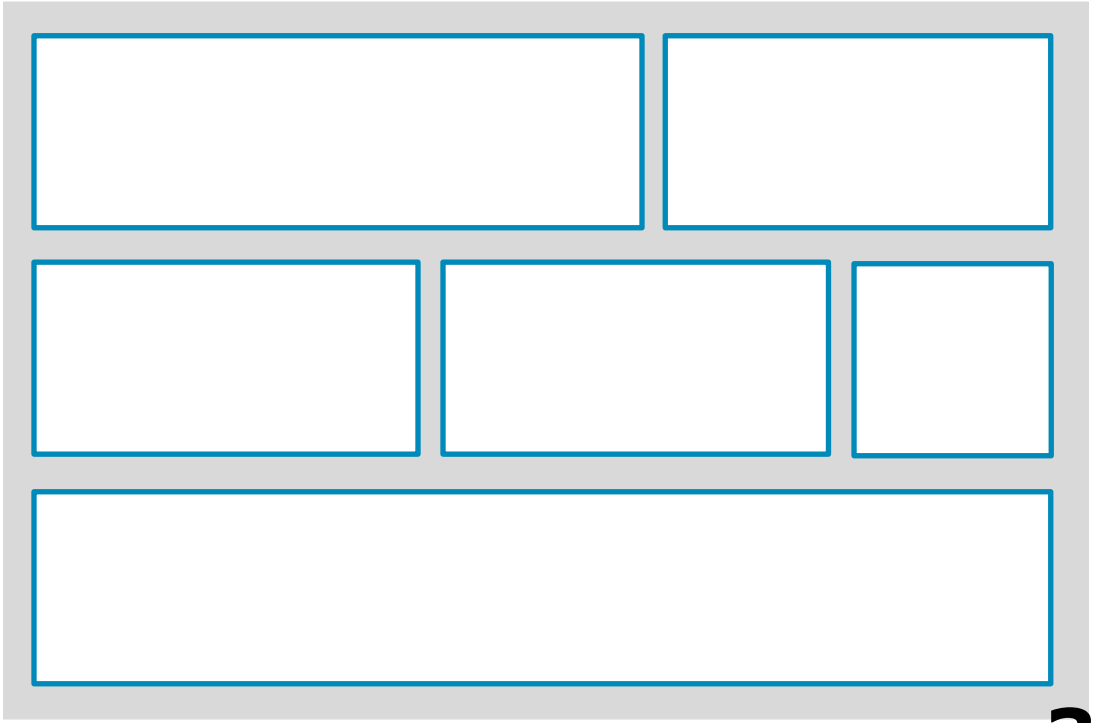
Leapfrog Technique:

- Remember types of (some) allocations
- Know offsets of pointers in each type



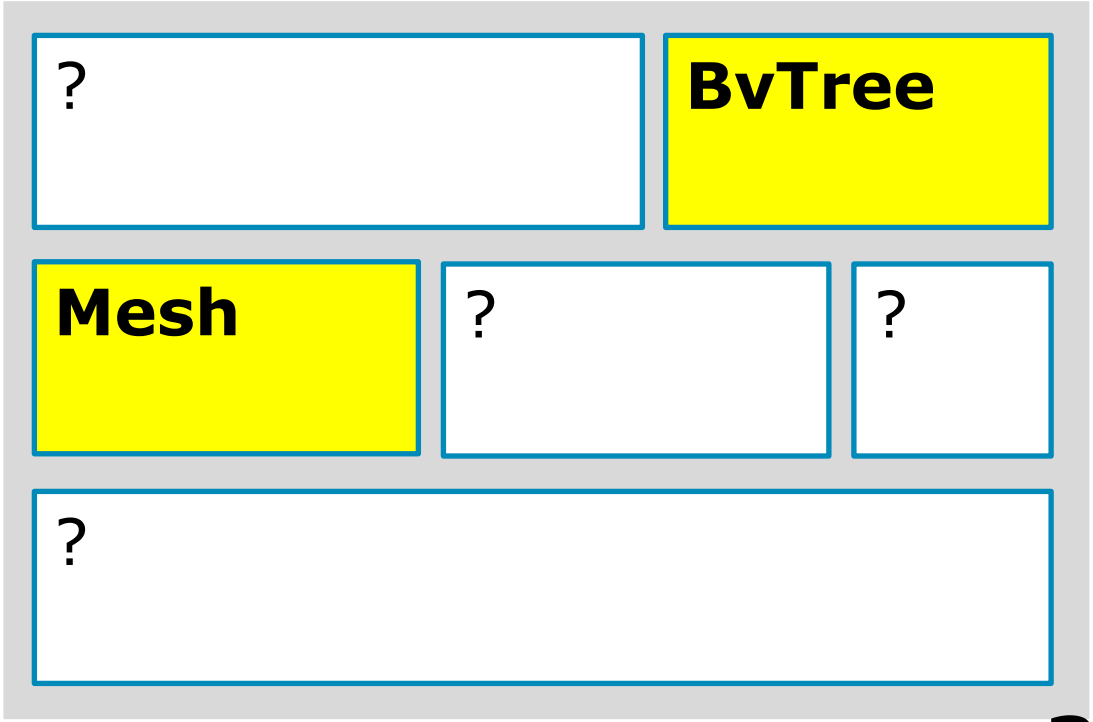
Raw Blocks

Raw pairs of (address,size)



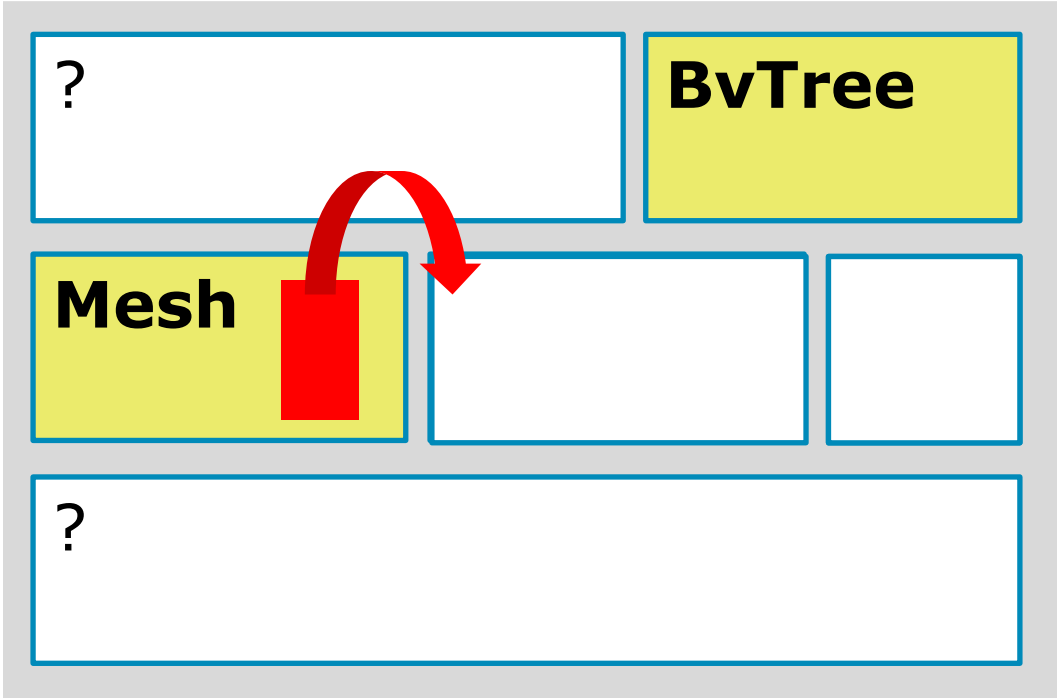
Type Roots

Hooked **class** operator new/delete



Reflection Walk

```
Mesh {
  Section [];
};
Section {
  Vector4[];
  Indices[];
};
```

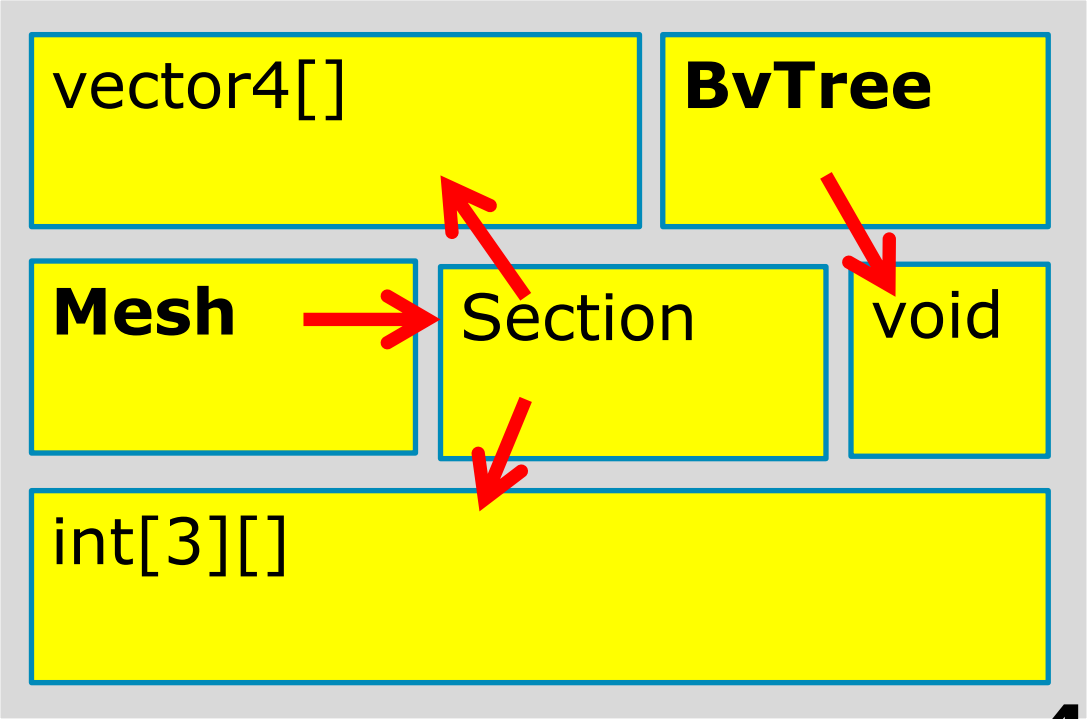


Reflection Walk

Finds all pointer-to relationships

Debug – Time & Stacktrace

Verify everything reached



Memory Implementation Details

Custom Type Handlers

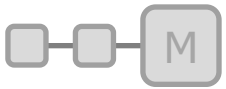
- slack space – false positives

Obfuscated Pointers

- `ptr ^= 1;`

Untyped Data

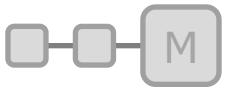
- `void*`

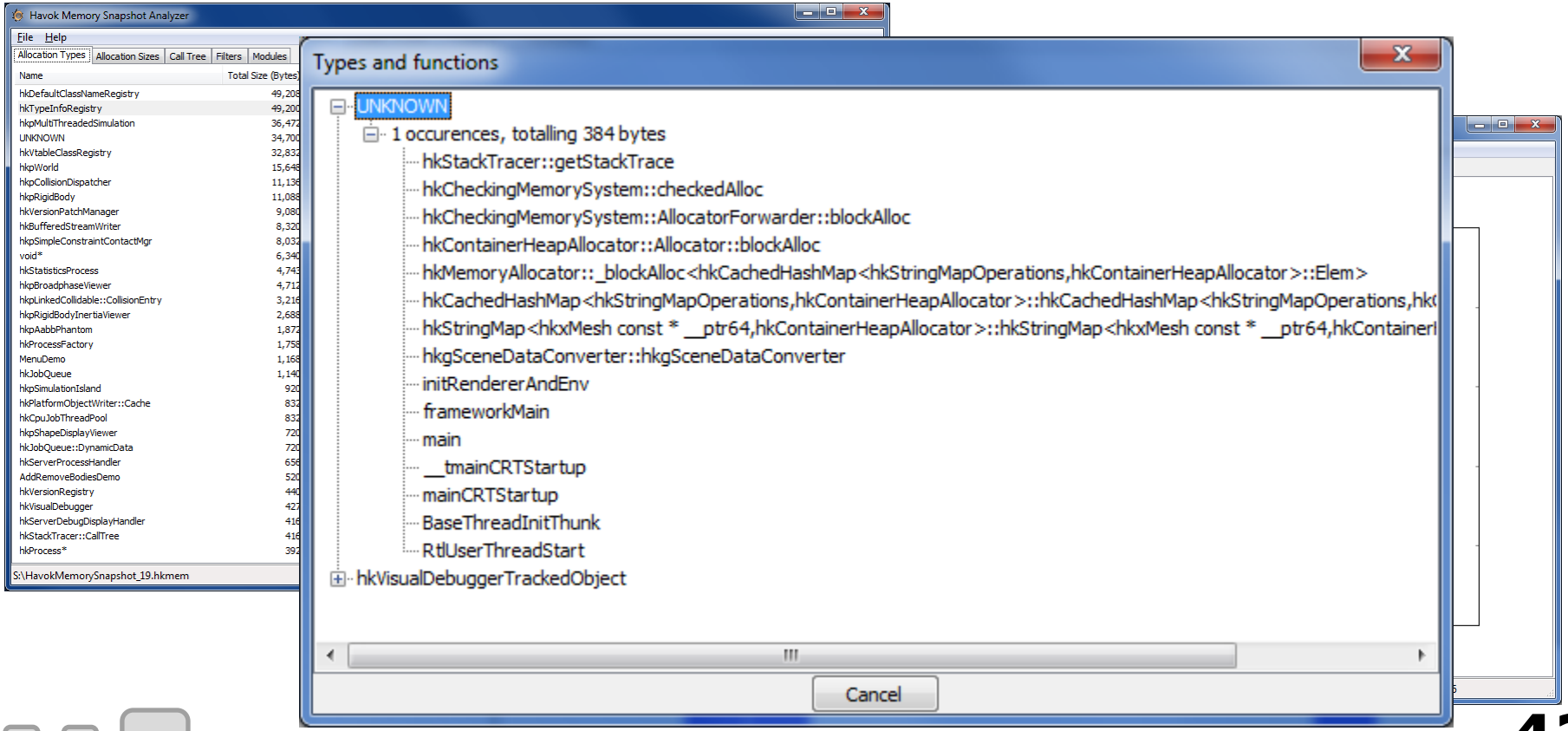


Annotated Memory Dump

```
D 1287169594
M Win32 “demos.exe"
T 0 UNKNOWN
T 1 hkNativeFileSystem
C 13 0x29656826 0x29828312 …
a 0 8
t 0 1
c 0 13
…
o 1 113 9
o 2 193
…
L 0x29656826 source\common\base\system\io\filesystem\hknativefilesystem.h(90):'hkNativeFileSystem::operator new'
```

```
#L(ocation) <address> <string name>?
#C(allstack) <callstack id> <address>+
#T(ype) <type id> <type name>
#a(llocation) <alloc id> <size>
#t(ype instance) <alloc id> <type id>
#c(allstack instance) <alloc id> <callstack id>
#o(wns) <alloc id> <'owned' alloc id>+
#M(odule information) <platform-dependent>
#D(ate of capture) <timestamp>
```




Havok Memory Snapshot Analyzer
File Help
Allocation Types
Allocation Sizes
Call Tree
Filters
Modules
Name
Total Size (Bytes)
hkDefaultClassNameRegistry 49,206
hkTypeInfoRegistry 49,200
hkpMultiThreadedSimulation 36,472
UNKNOWN 34,700
hkVtableClassRegistry 32,832
hkpWorld 15,648
hkpCollisionDispatcher 11,136
hkpRigidBody 11,088
hkVersionPatchManager 9,080
hkBufferedStreamWriter 8,320
hkpSimpleConstraintContactMgr 8,032
void* 6,340
hkStatisticsProcess 4,743
hkpBroadphaseViewer 4,712
hkpLinkedCollidable::CollisionEntry 3,216
hkpRigidBodyInertiaViewer 2,688
hkpAabbPhantom 1,872
hkProcessFactory 1,758
MenuDemo 1,168
hkJobQueue 1,140
hkpSimulationIsland 920
hkPlatformObjectWriter::Cache 832
hkCpuJobThreadPool 832
hkpShapeDisplayViewer 720
hkJobQueue::DynamicData 720
hkServerProcessHandler 656
AddRemoveBodiesDemo 520
hkVersionRegistry 440
hkVisualDebugger 427
hkServerDebugDisplayHandler 416
hkStackTracer::CallTree 416
hkProcess* 392
S:\HavokMemorySnapshot_19.hkmem
Types and functions
UNKNOWN
1 occurences, totalling 384 bytes
hkStackTracer::getStackTrace
hkCheckingMemorySystem::checkedAlloc
hkCheckingMemorySystem::AllocatorForwarder::blockAlloc
hkContainerHeapAllocator::Allocator::blockAlloc
hkMemoryAllocator::_blockAlloc<hkCachedHashMap<hkStringMapOperations,hkContainerHeapAllocator>::Elem>
hkCachedHashMap<hkStringMapOperations,hkContainerHeapAllocator>::hkCachedHashMap<hkStringMapOperations,hk
hkStringMap<hkxMesh const * __ptr64,hkContainerHeapAllocator>::hkStringMap<hkxMesh const * __ptr64,hkContainerH
hkgSceneDataConverter::hkgSceneDataConverter
initRendererAndEnv
frameworkMain
main
__tmainCRTStartup
mainCRTStartup
BaseThreadInitThunk
RtlUserThreadStart
hkVisualDebuggerTrackedObject
Cancel

Memory Report Conclusion

Label root blocks & grow with reflection

We found offline analysis useful

Low maintenance

Accounts for all reachable memory

Or tells you where it came from

Versioning

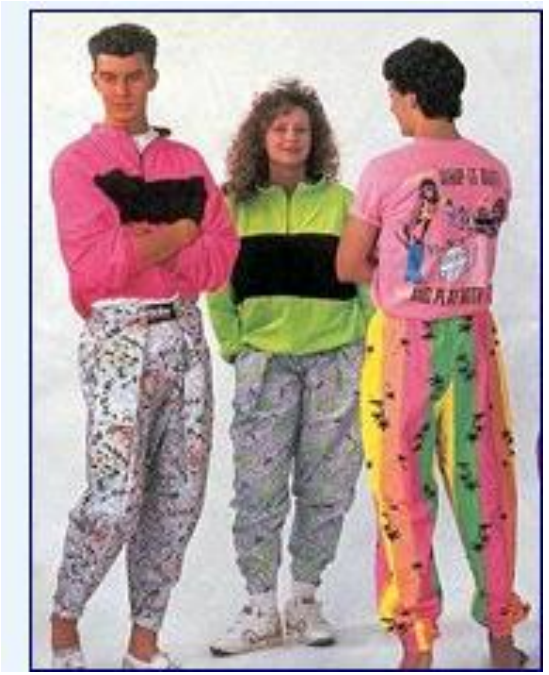
Change Happens

Hitting a moving target

Not much in literature

Fidelity

Separate to Serialization



Manual Conditionals

Version number

Conditionals

Inter-object changes?

Large refactor?

```
xtream.TokenMustBe("POINT_A"); xtream>>point_A;
bool has_two_bodies = true;
if (xtream.GetVersion()>=1350)
{
 xtream.TokenMustBe("HAS_TWO_BODIES");
 xtream>>has_two_bodies;
}
if (has_two_bodies)
{
    xtream.TokenMustBe("BODY_B");
    xtream>>prot_buffer;
    priv_rigid_body_B = s->GetRigidBody(prot_buffer);
    if (!priv_rigid_body_B)
       throw_exception("Rigidbody unknown in Spring");
}
xtream.TokenMustBe("POINT_B"); xtream>>point_B;
//…
```

Ideal Versioning System

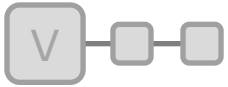
Don’t constrain my changes

Don’t slow me down

Help me fix it up

Don’t make me revisit

Don’t make me think



Snapshot Versioning

Compare **all** previous vs current metadata

Function updates changed members

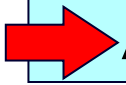
Check up-to-date with CRC of reflection

Snapshot example

```
A { int x; int y; }
B { A* a; }
```

```
A { int y; }
B { A* a; int x; }
```

```
void Update_B( void* oldB, void* newB ) {
  // newB->x = oldB->a->x; }
```



```
{ “A”, 0x1234, 0x8989, NULL },
{ “B”, 0xabcd, 0xfed4, Update_B }.
```

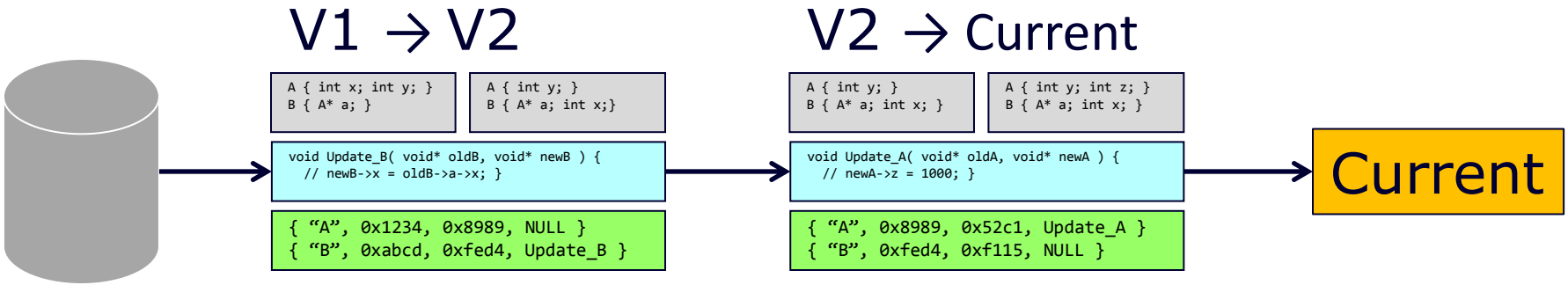
Version Function

```
A { int x; int y; }
B { A* a; }
```

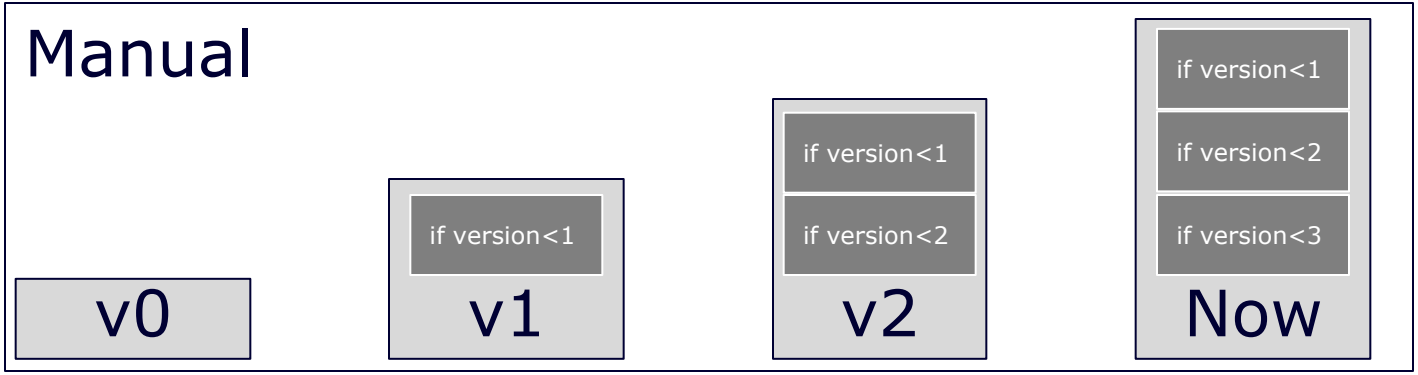
```
A { int y; }
B { A* a; int x; }
```

```
void Update_B( Object& oldB, Object& newB )
{
  newB["x"] = oldB["a"].asObject()["x"].asInt();
}
```

Chaining Snapshot Updates



What Have We Gained?



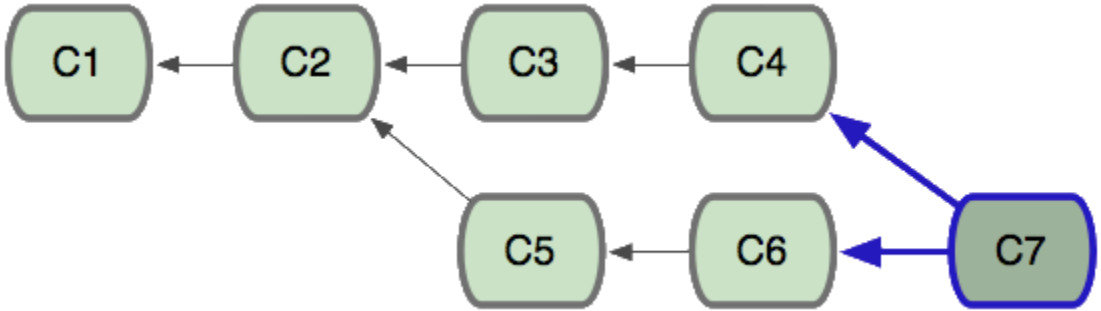
Finer Grained Changes

Full snapshots too heavy

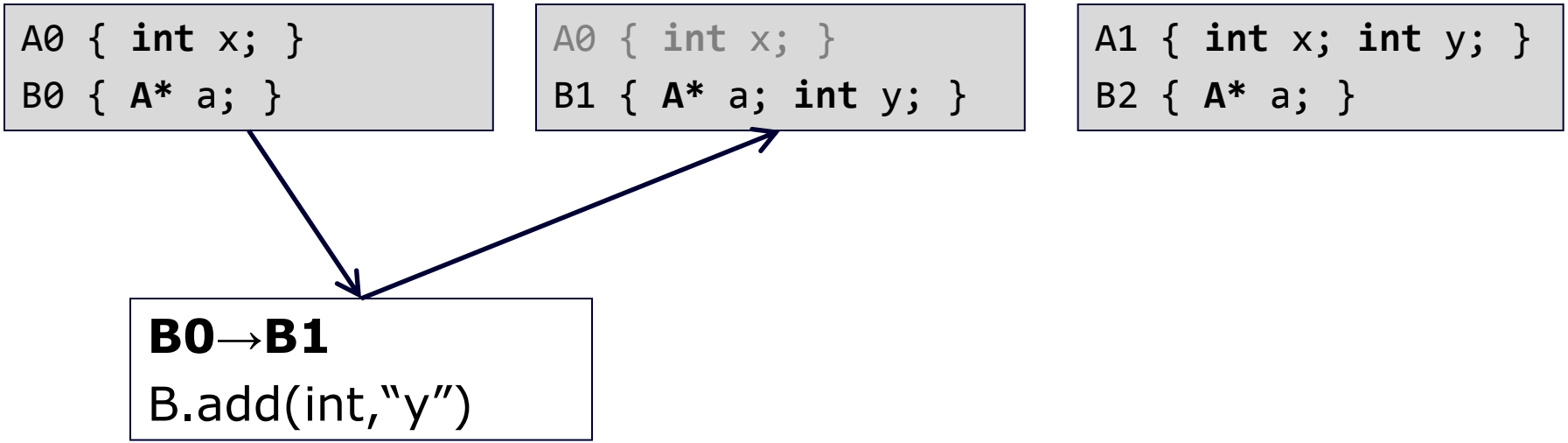
We hired artists

New products, lots of branches

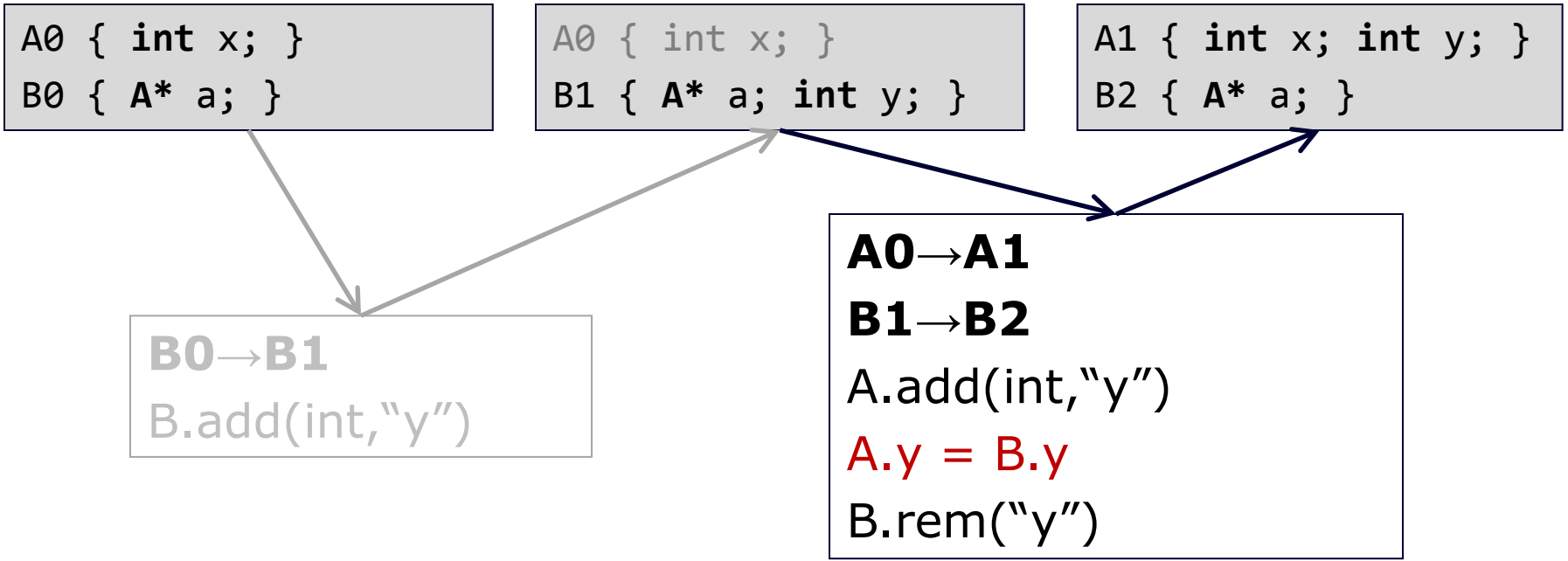
- No global timeline



Patching (1)


A0 { int x; }
B0 { A* a; }
A0 { int x; }
B1 { A* a; int y; }
A1 { int x; int y; }
B2 { A* a; }
B0→B1
B.add(int,"y")

Patching (2)


A0 { int x; }
B0 { A* a; }
A0 { int x; }
B1 { A* a; int y; }
A1 { int x; int y; }
B2 { A* a; }
B0→B1
B.add(int,“y”)
A0→A1
B1→B2
A.add(int,“y”)
A.y = B.y
B.rem(“y”)

Atomic Patch Types

Add/Remove/Rename member

Add/Remove/Rename class

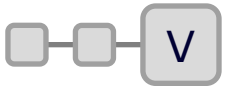
Change member default

Cast object type

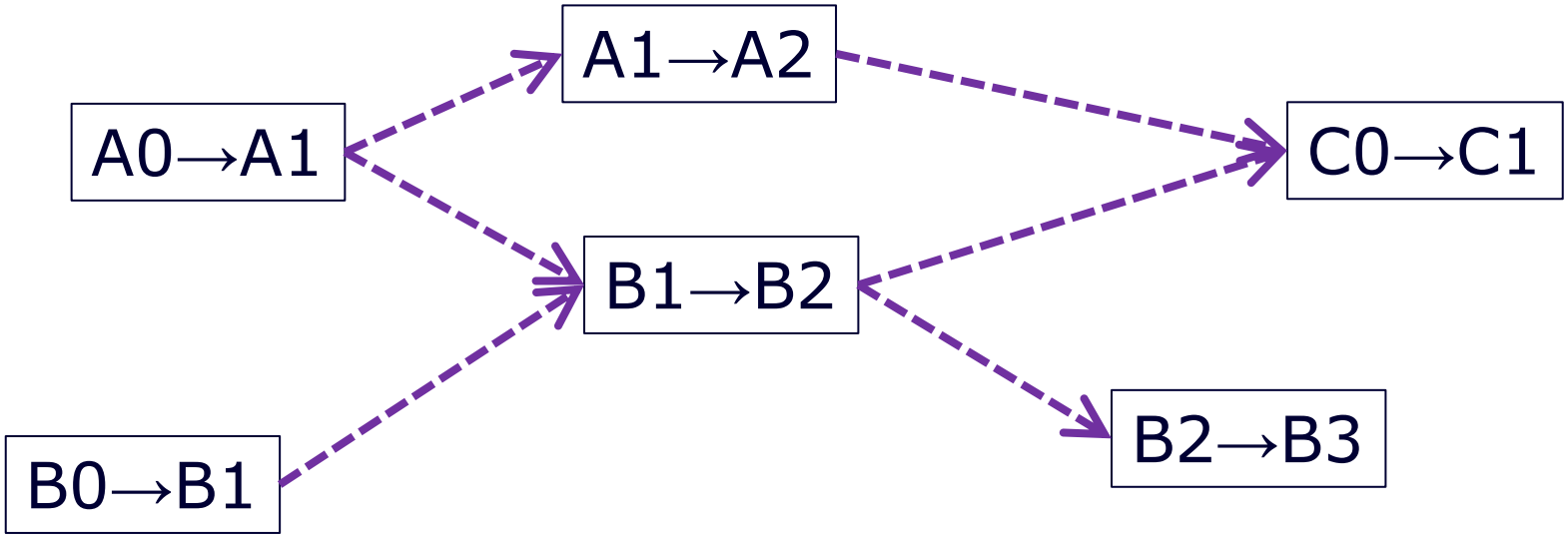
Set parent

Depends

Function

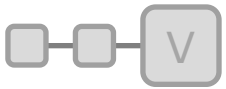


Graph Of Patches

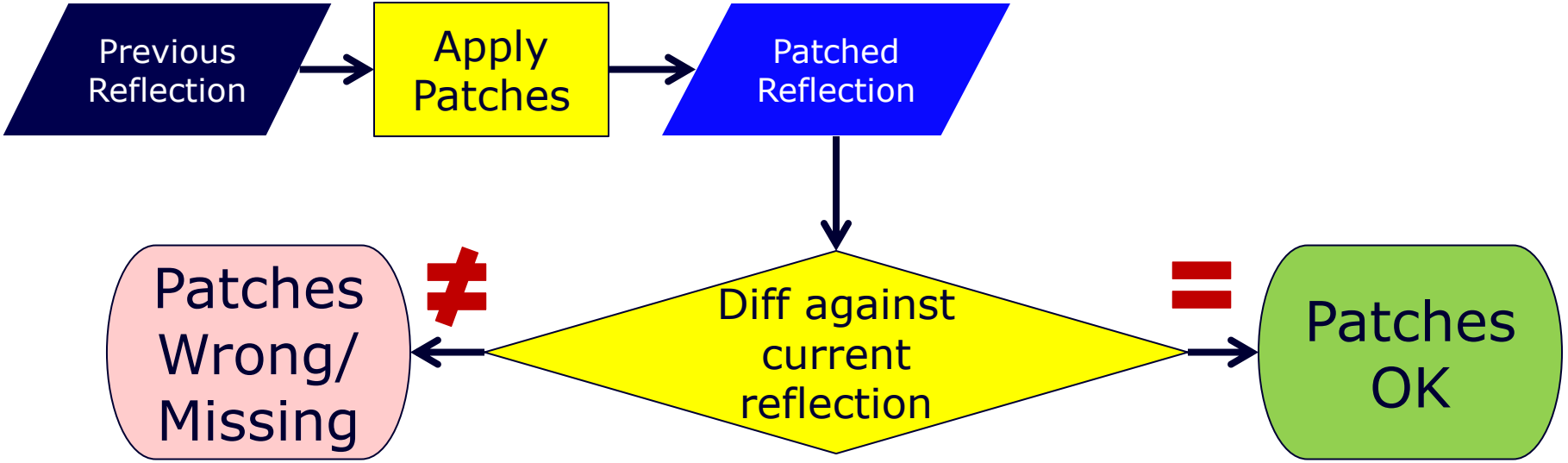


Sample Patch

```
// This block is generated by the metadata comparison.
// It is pasted into the source and expands out to a few constant structures.
PATCH_BEGIN("hkbLookAtModifier", 2, "hkbLookAtModifier", 3)
  // Require hkbEventBase version 3 before running this patch
  PATCH_DEPENDS("hkbEventBase", 3)

  PATCH_MEMBER_ADDED_VEC_4("neckForwardLS", 0.0f,1.0f,0.0f,0.0f)

  // user edit: add & remove changed to rename
  PATCH_MEMBER_RENAMED("headForwardHS", "headForwardLS")

  // user edit: call a C function here
  PATCH_FUNCTION( hkbLookAtModifier_2_to_3 )
PATCH_END()
```



Verification



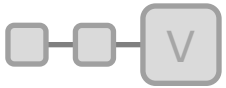
Versioning Workflow

Make your changes

Run a reflection diff utility

- Copy & paste the suggested patch
- Edit (add&remove → rename)
- Write version function if necessary

Done



Versioning Conclusion

Massively reduced tax

Quick to add versioning

Cheap to test

Non-prescriptive workflow

Conclusion

Reflection

Serialization

Memory

Binding

Versioning