How to Use SVS

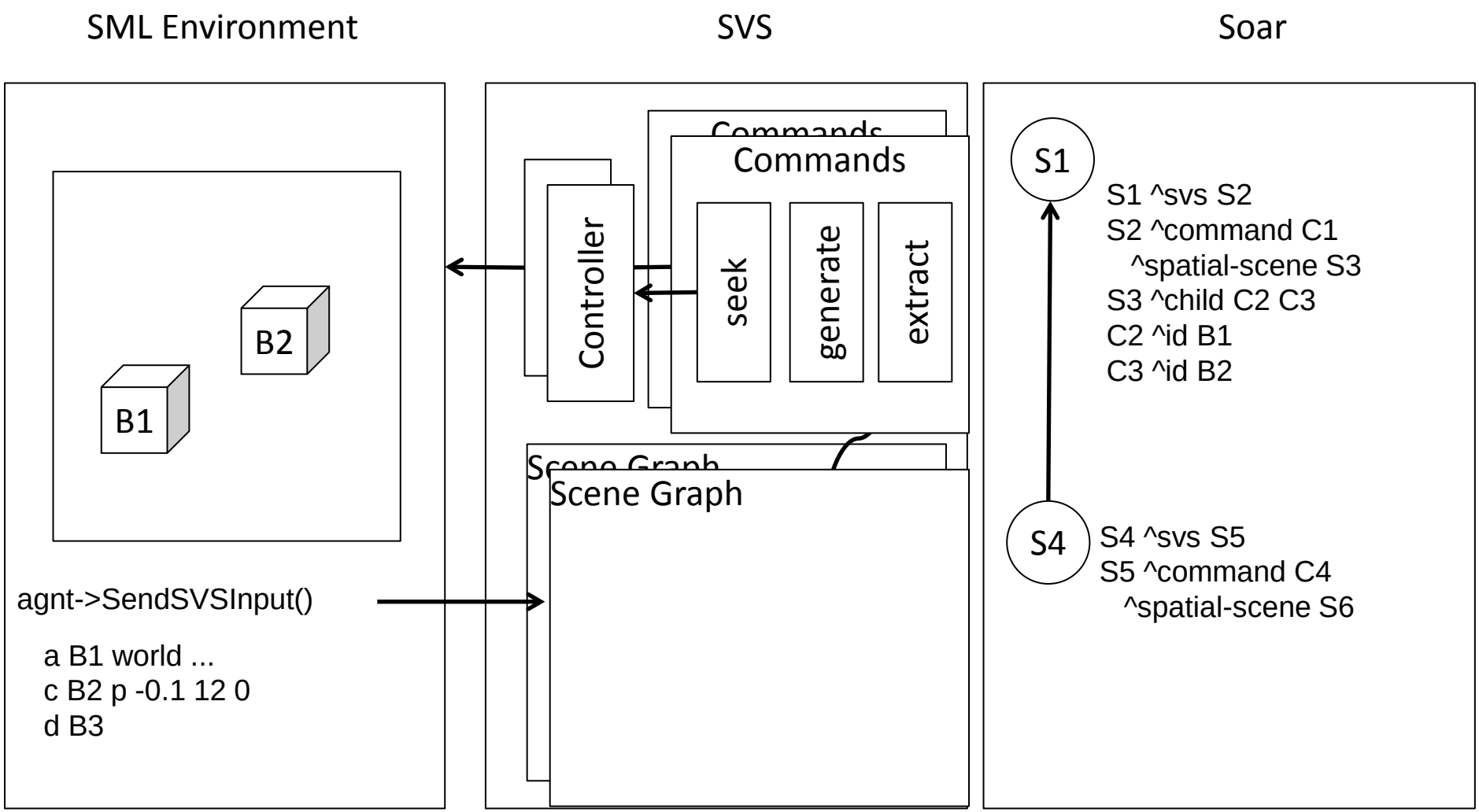
Joseph Xu

Soar Workshop 2012

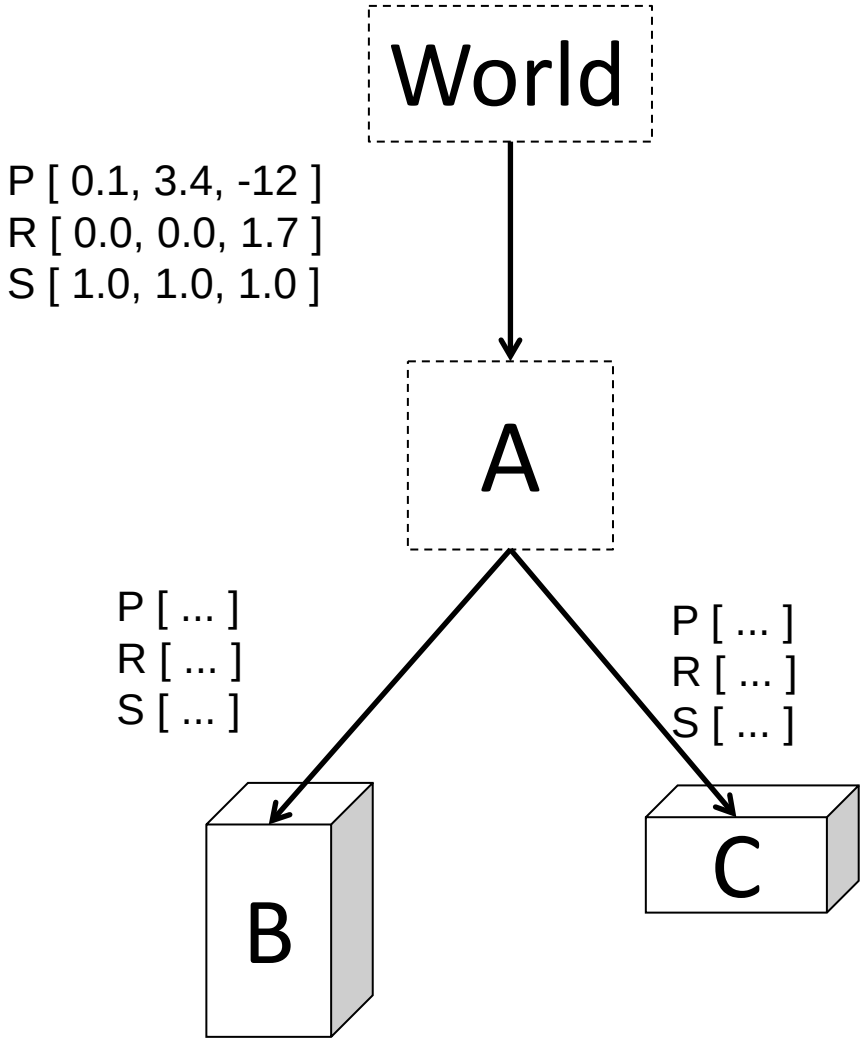
History

- Soar Visual Interface (SVI) – Scott Lathrop
- Spatial Reasoning System (SRS) – Sam Wintermute
- Soar Visual System = SVI + SRS – Sam Wintermute
- SVS “lite” – What this presentation is on
  - Kernel & SML integration
  - Traded functionality for simplicity

Overview


SML Environment
SVS
Soar
B2
B1
agnt->SendSVSInput()
a B1 world ...
c B2 p -0.1 12 0
d B3
Commands
Commands
Controller
seek
generate
extract
Scene Graph
Scene Graph
S1
S1 ^svs S2
S2 ^command C1
^spatial-scene S3
S3 ^child C2 C3
C2 ^id B1
C3 ^id B2
S4
S4 ^svs S5
S5 ^command C4
^spatial-scene S6

Scene Graph




```
(<s> ^svs <svs>)
(<svs> ^spatial-scene
       <scn>)
(<scn> ^id world
       ^child <c1>)
(<c1> ^id A
      ^child <c2>
      ^child <c3>)
(<c2> ^id B)
(<c3> ^id C)
```

Scene Graph Editing Language

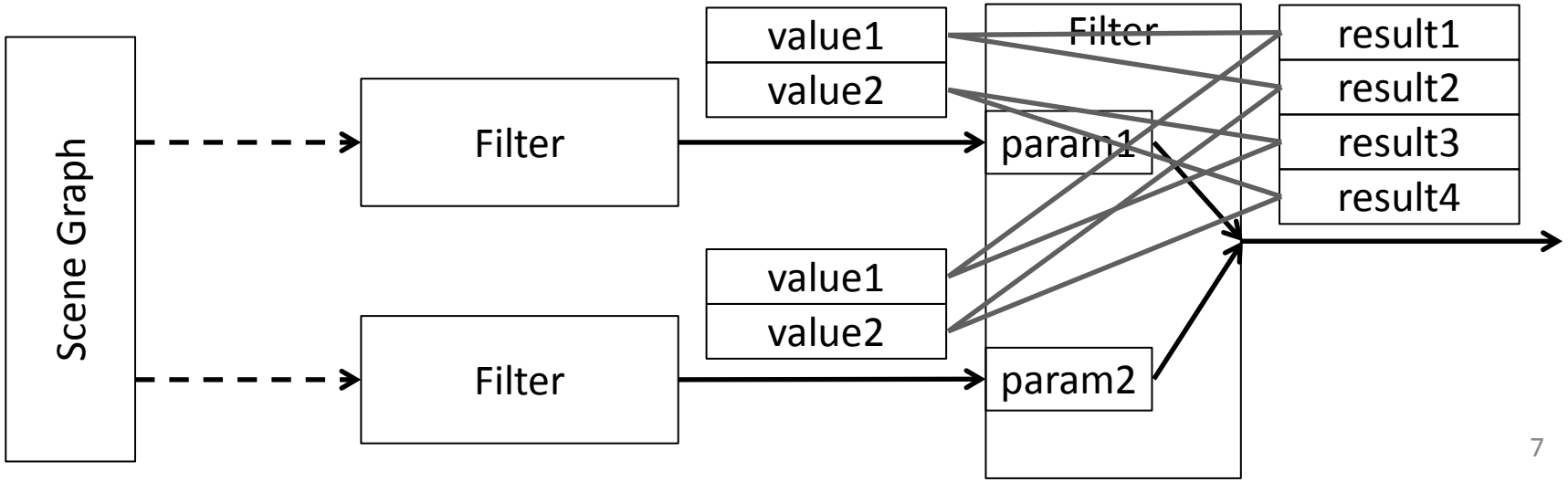
- Environment modifies SVS scene graph with agent->SendSVSInput("<sgel>")
- Add an object named <o> as a child of <p>
  - a <o> <p> v <vertices> p <position> r <rotation> s <scaling>
- Change an object's transforms or vertices
  - c <o> v <vertices> p <position> r <rotation> s <scaling>
- Delete an object
  - d <o>

Recommended Paradigm

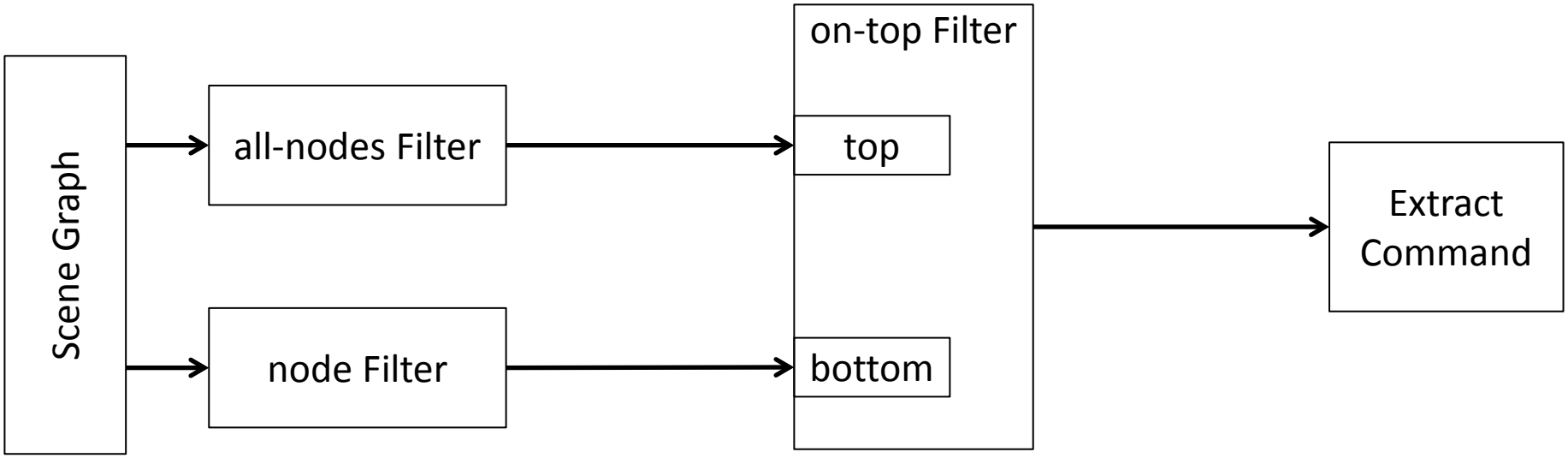
```
void output_handler(smlRunEventId id, void *env,
                    Agent *a, smlPhase phase)
{
    int ncmds = a->GetNumberCommands();
    for (int i = 0; i < ncmds; ++i) {
        <handle output link commands>
    }
    <run environment simulation with output>
    string in = <generate SGEL from environment state>;
    a->SendSVSInput(in.c_str());
}

int main() {
    <initialize agent and environment>
    a->RegisterForRunEvent(smlEVENT_AFTER_OUTPUT_PHASE,
                           &output_handler, env, true);
    <run>
}
```

Filters

- Basic unit of computation
- Multiple list inputs, single list output
- Different ways to combine inputs:
  - Full product. Ex: N of param A, M of param B -> N * M results
  - Tuples. Ex. N of param A, N of param B -> N results

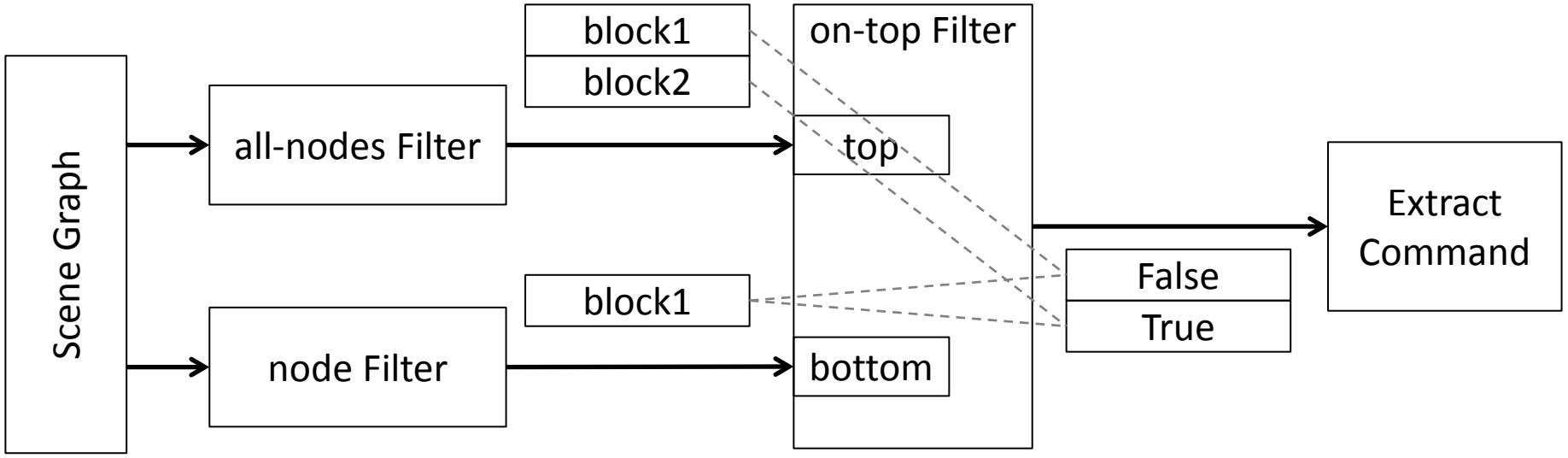


Extract Command




```
(S2 ^command.extract E1)
(E1 ^type on-top
    ^top T1
    ^bottom B1)
(T1 ^type all-nodes)
(B1 ^type node ^name block1)
```

Extract Command




```
(S2 ^command.extract E1)
(E1 ^type on-top
    ^top T1
    ^bottom B1
    ^status success
    ^result R1)
(T1 ^type all-nodes)
(B1 ^type node ^name block1)
```

```
(R1 ^positive P1
    ^negative N1)
(P1 ^atom A1)
(A1 ^a block1 ^b block2)
(N1 ^atom A2)
(A2 ^a block1 ^b block1)
```

Special Case: Intersection Filter

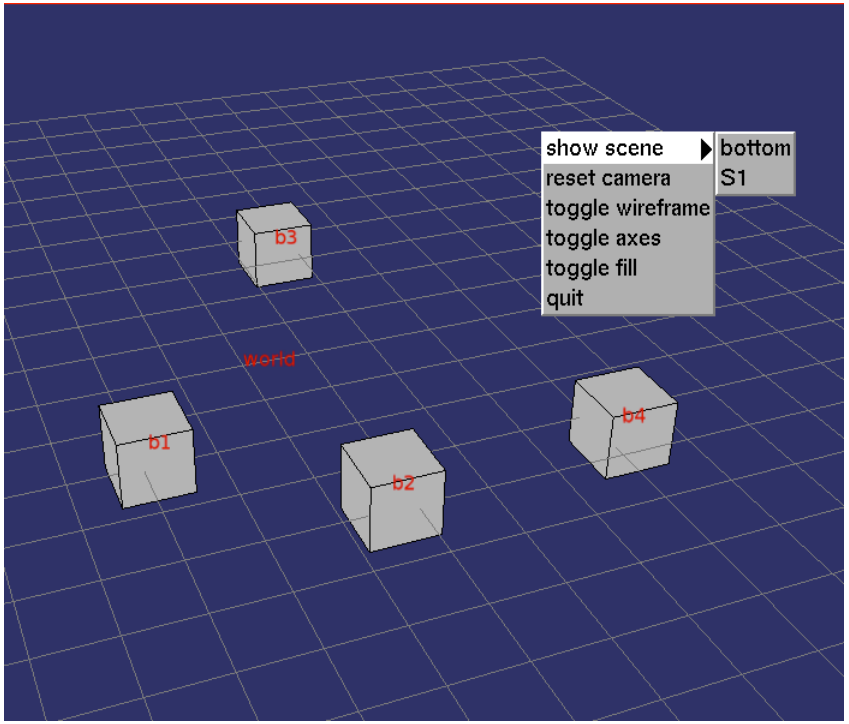
- SVS objects are arbitrary convex polyhedrons, intersection calculations are expensive
- Handle with collision detector from Bullet Physics engine
  - Uses cheap broad-phase calculation to rule out collisions
  - Precompiled libs are distributed with SVS
- Caveat: All objects in scene are inserted into collision detector. If you calculate one intersection, you've calculated them all

Writing Your Own Filter

- Inherit the map_filter<result type> class
  - Just need to code logic for calculating one result from one set of input parameters
  - Result caching and input combinations all taken care of for you

Visualization

- Separate 3D viewer program to minimize SVS dependencies
  - OpenSceneGraph, GLUT, quickhull
- SVS talks to viewer via file socket /tmp/viewer
- Plain-text language very similar to SGEL
  - Not specific to SVS



How to Get It

- Clone this git repo into Core/SVS
  - https://github.com/jzxu/SVS.git
- Patch the kernel to use SVS:

  $ patch -p0 < Core/SVS/patch
- Compile and run Soar like normal
- To run viewer:

  Install OpenSceneGraph, GLUT, and quickhull (all available in most Linux package repos)

  $ cd Core/SVS/vis/viewer

  $ scons

  $ ./viewer

Evaluation

Coal

- Only works on *nix and Mac OS
  - Uses OS-specific BSD sockets, and I haven't written Windows version

Nuggets

- Works out of the box on Linux
- Successfully used in Bolt project