University of Michigan

The SVS Spatial/Visual Reasoning System

Samuel Wintermute
University of Michigan

Soar Workshop 28, Ann Arbor, MI

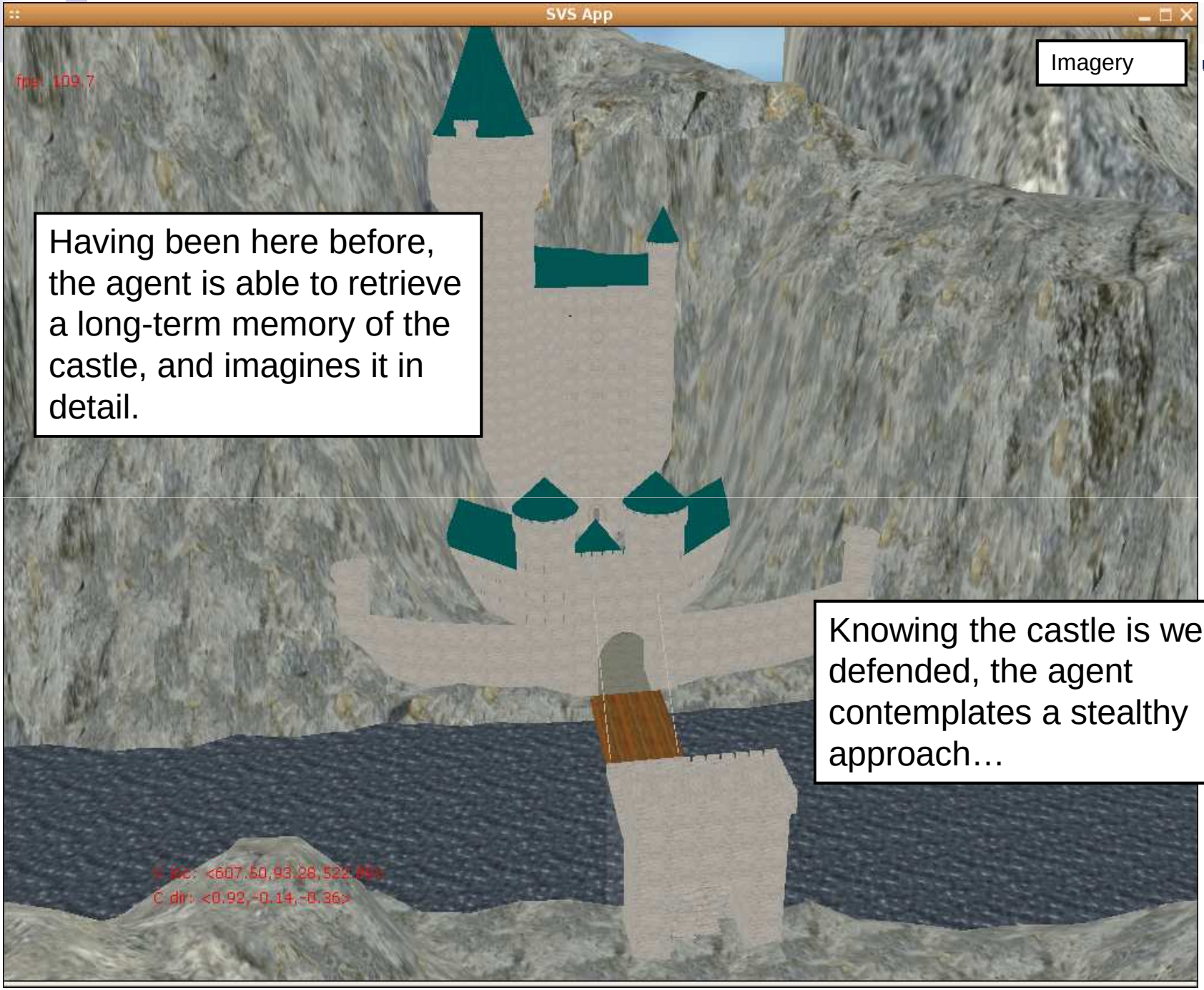
Outline

- Motivation
- Design
- SVS Scene Graphs
- Spatial Generation and Extraction
- Visual Generation and Extraction
- Example Problem
- Nuggets and Coal

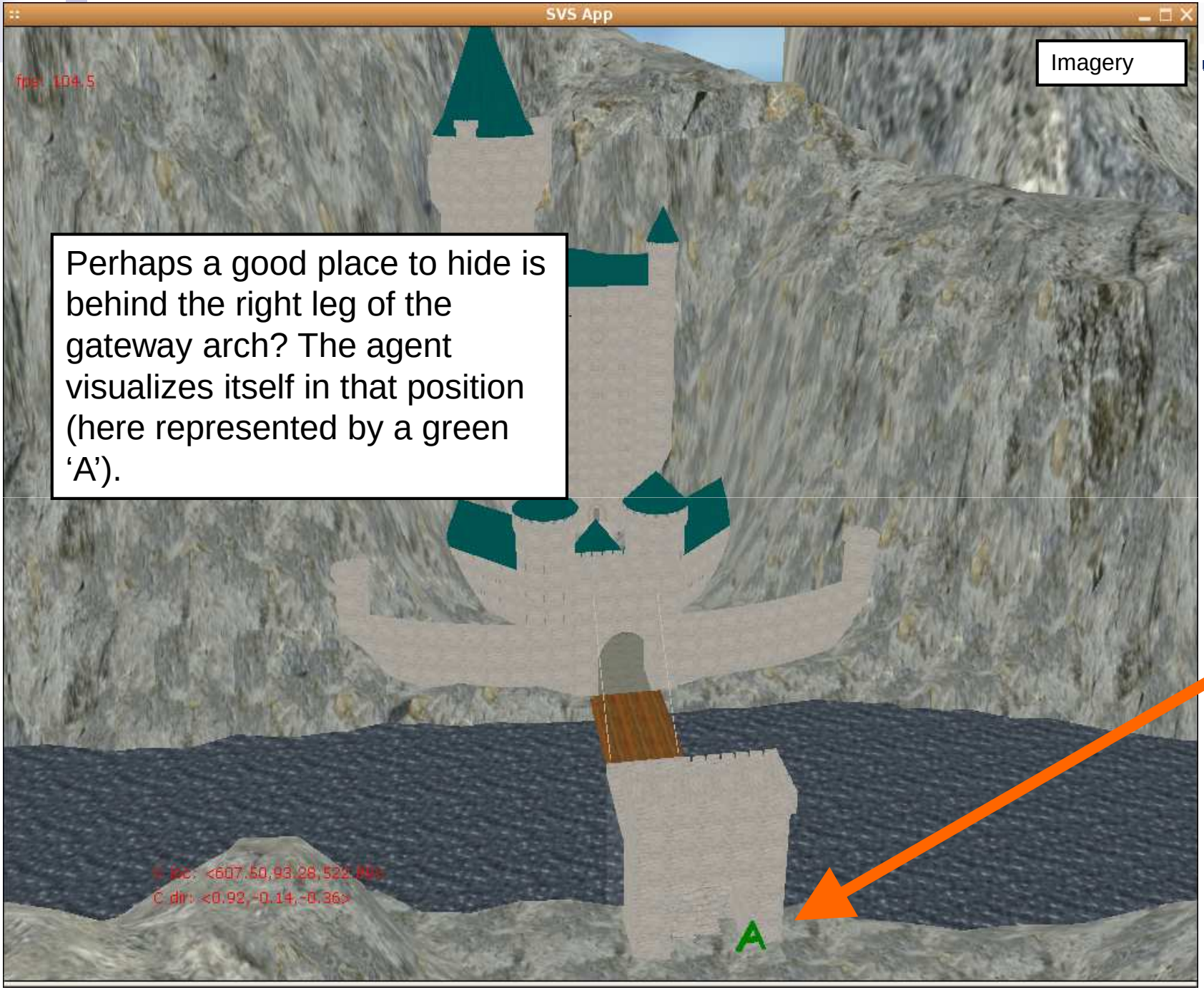
Motivation

- Much work has been done in the last few years to enhance Soar with lower-level spatial and visual reasoning
- Two systems, SVI and SRS, have been developed
- SVS is the unification of SVI and SRS, with additional features

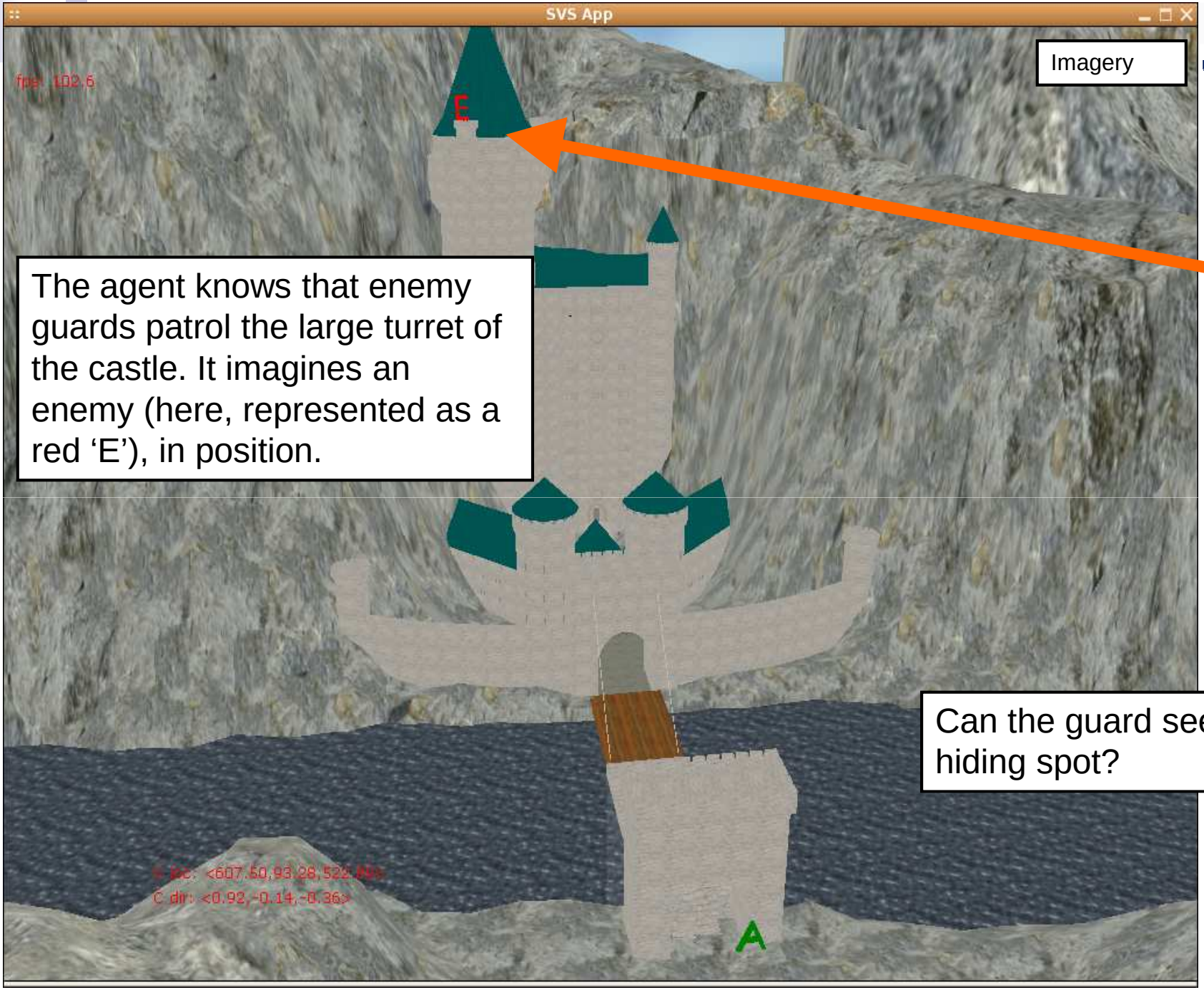
University of Michigan

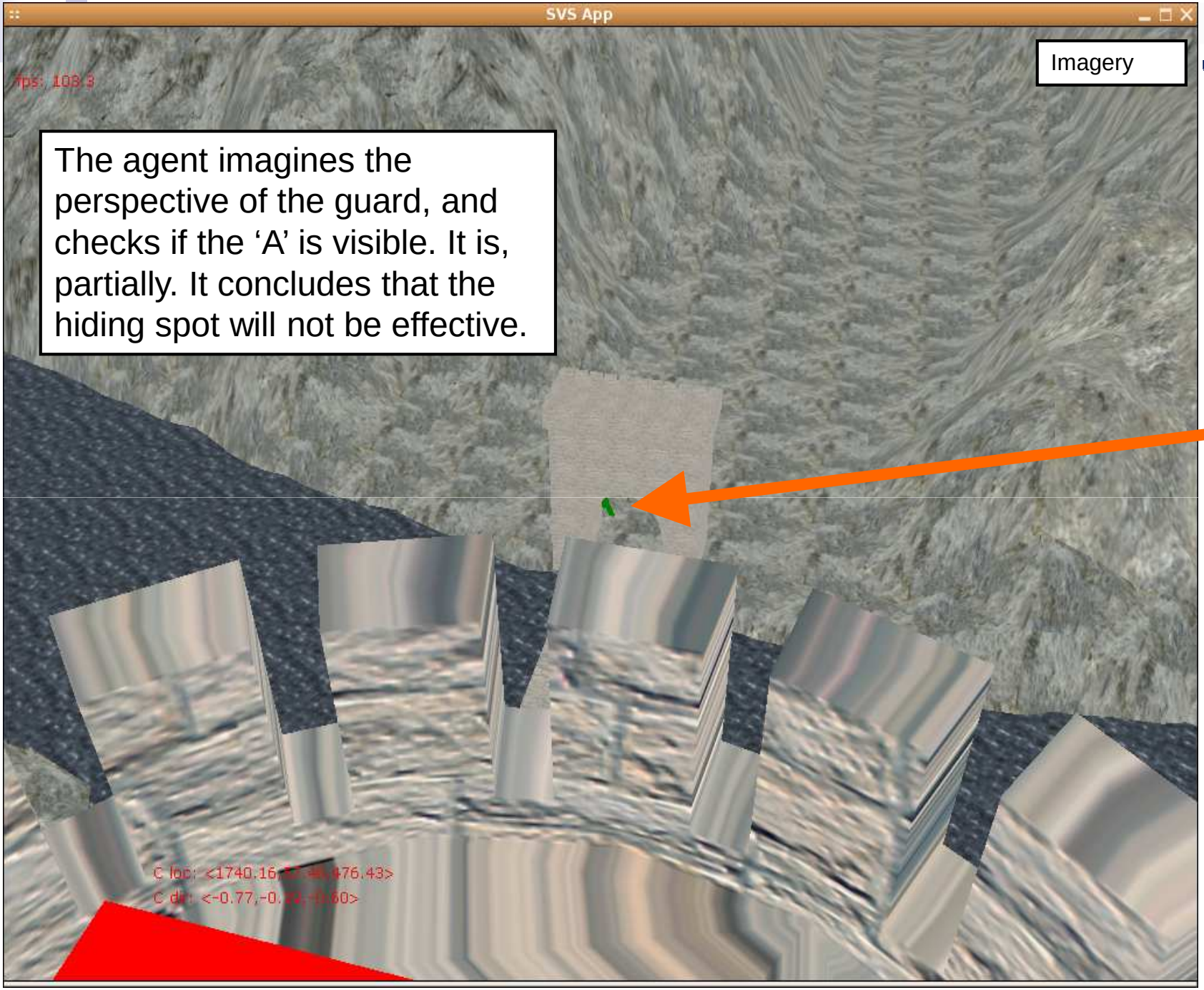


University of Michigan



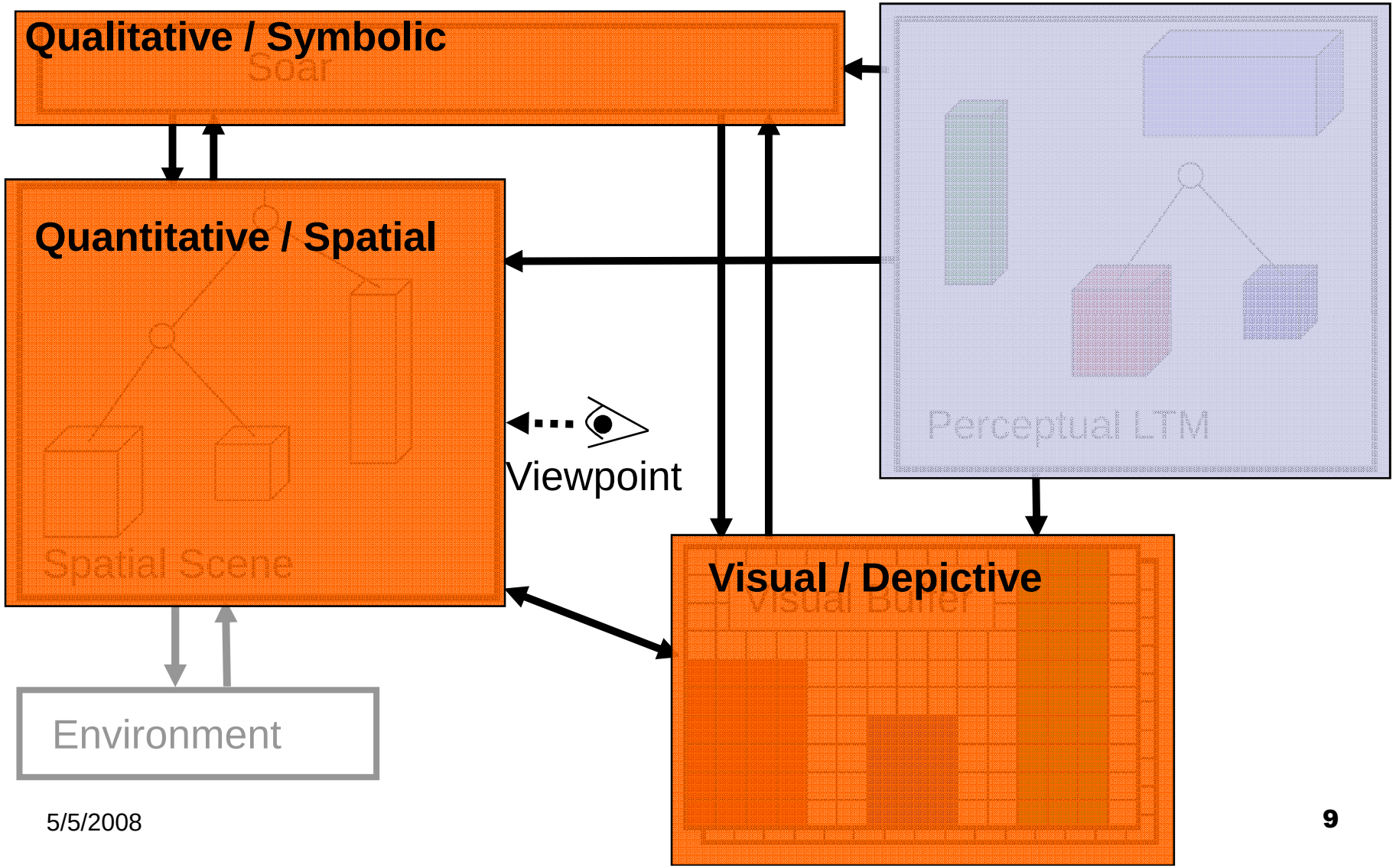
University of Michigan


SVS App
Imagery
E
A
The agent knows that enemy guards patrol the large turret of the castle. It imagines an enemy (here, represented as a red ‘E’), in position.
Can the guard see the hiding spot?



University of Michigan

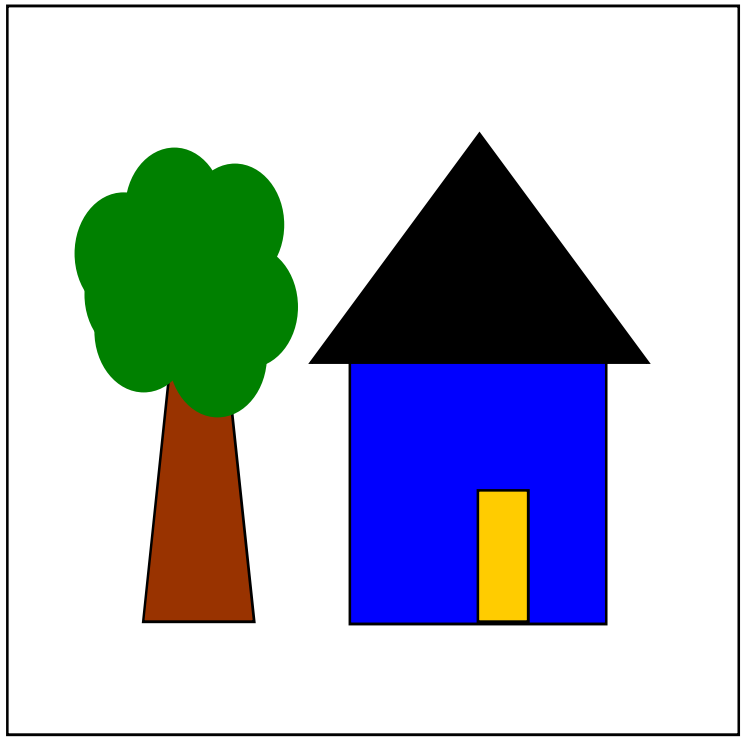
High-Level Design

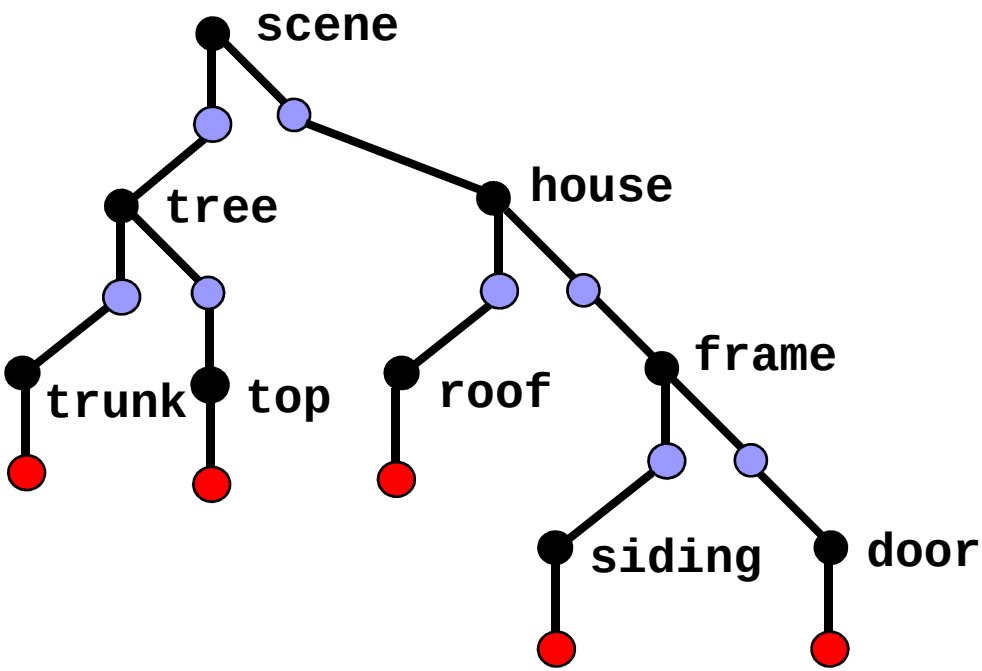


SVS Scene Graphs

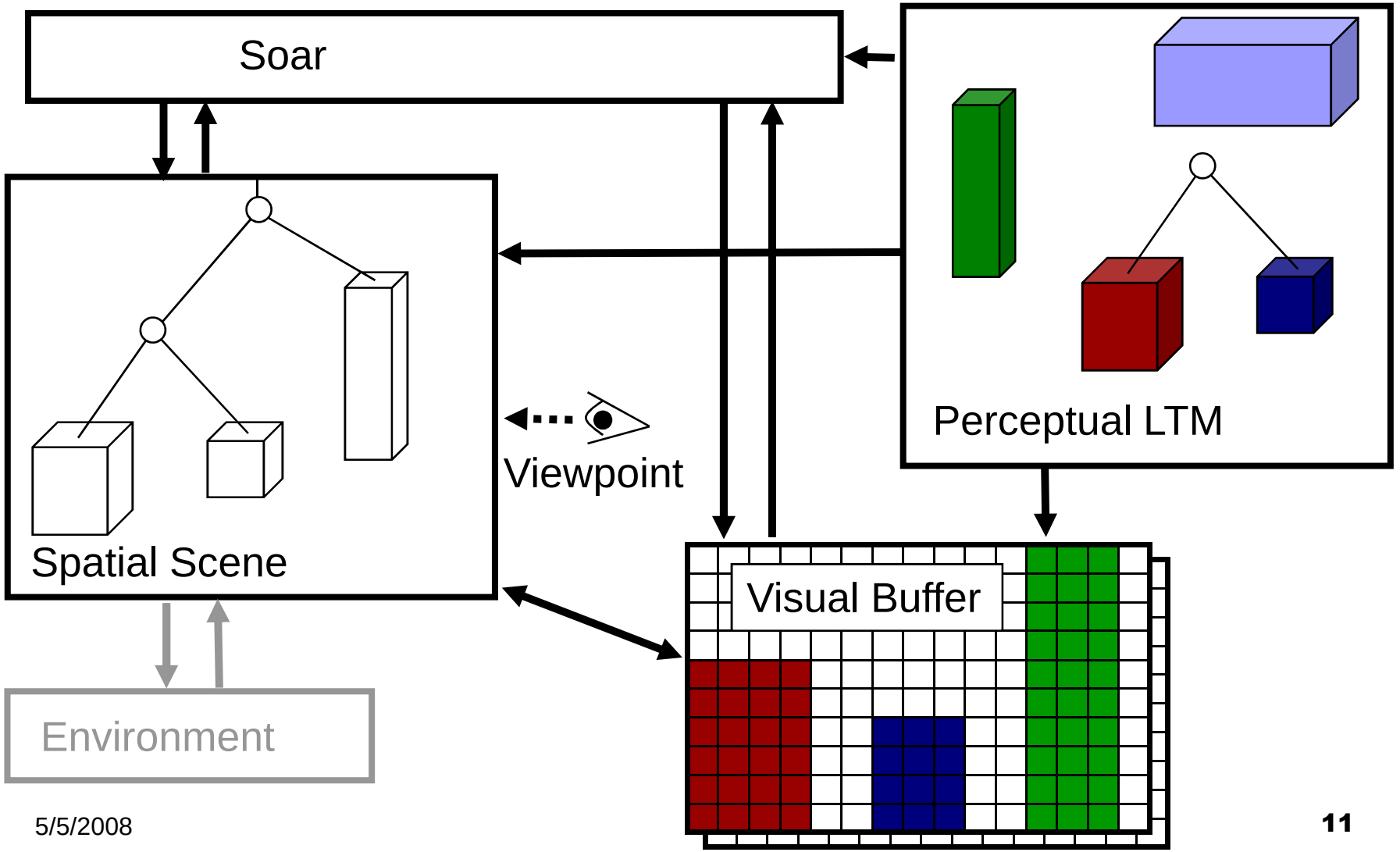
- **object**
- **transformation**
- **texture**

- SVS now makes scene graphs (in Spatial STM and LTM) accessible to Soar
- Soar can expand and contract nodes



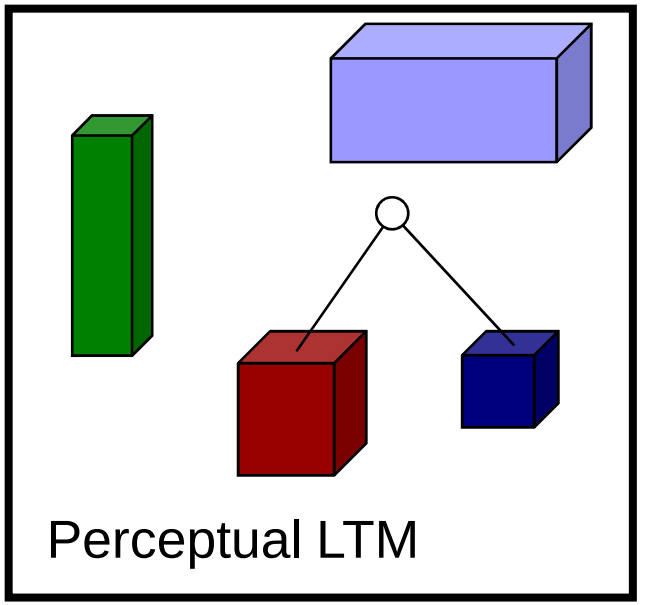


High-Level Design

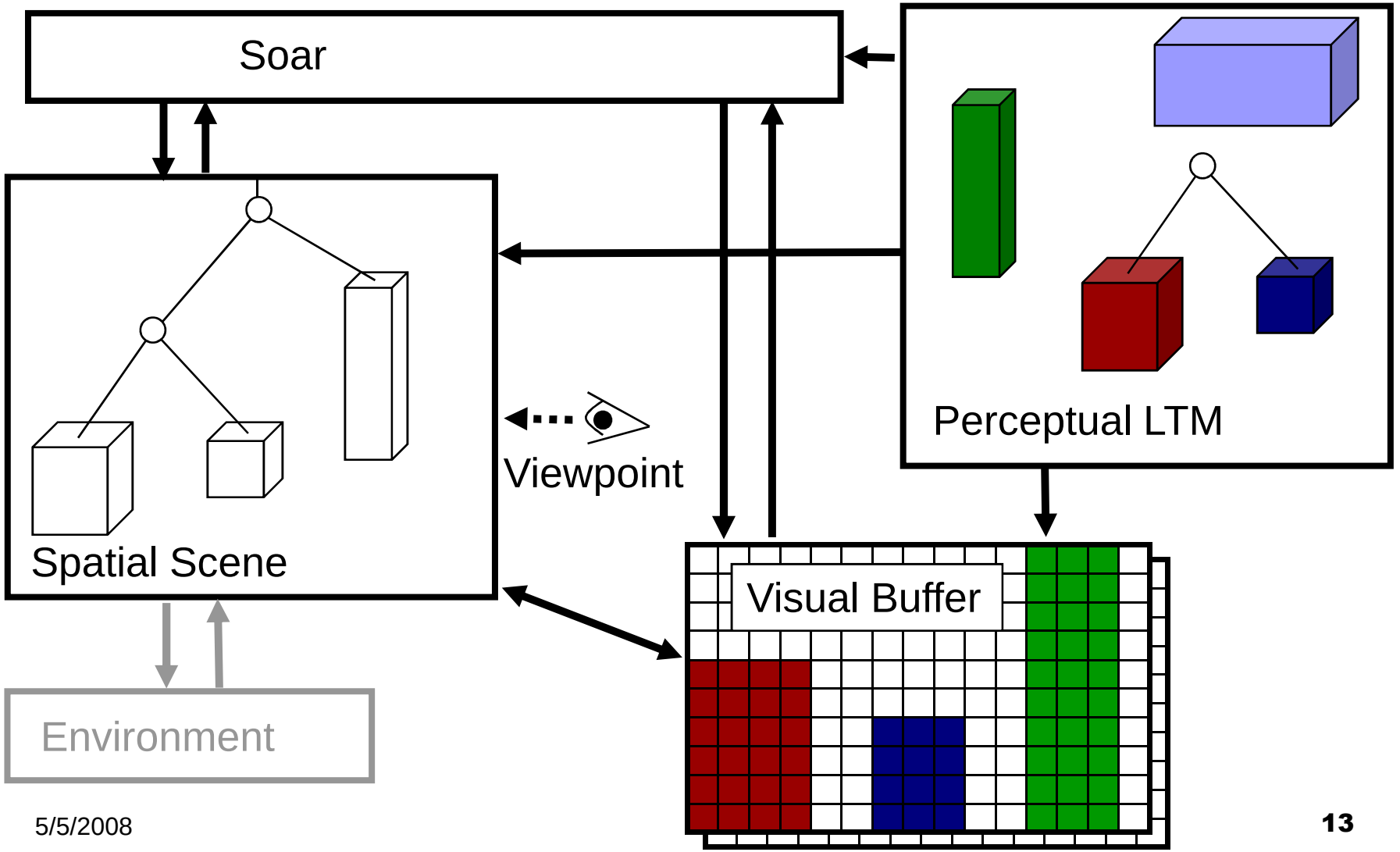


Perceptual LTM

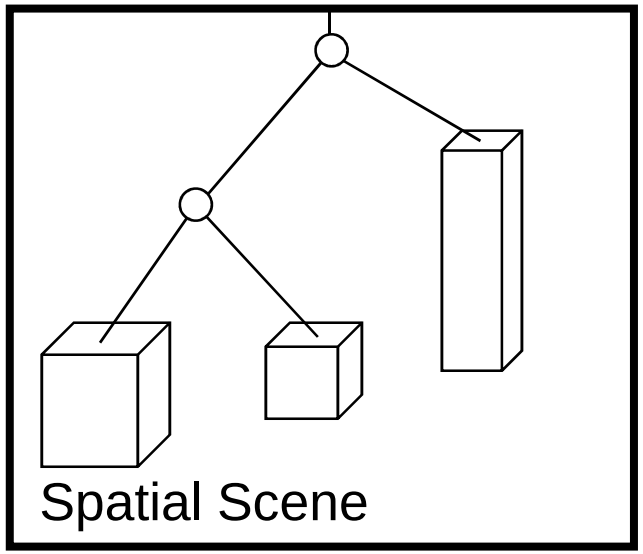
- Stores prototypical objects (shapes), textures, and transformations
- Hierarchically organized in scene graphs



High-Level Design

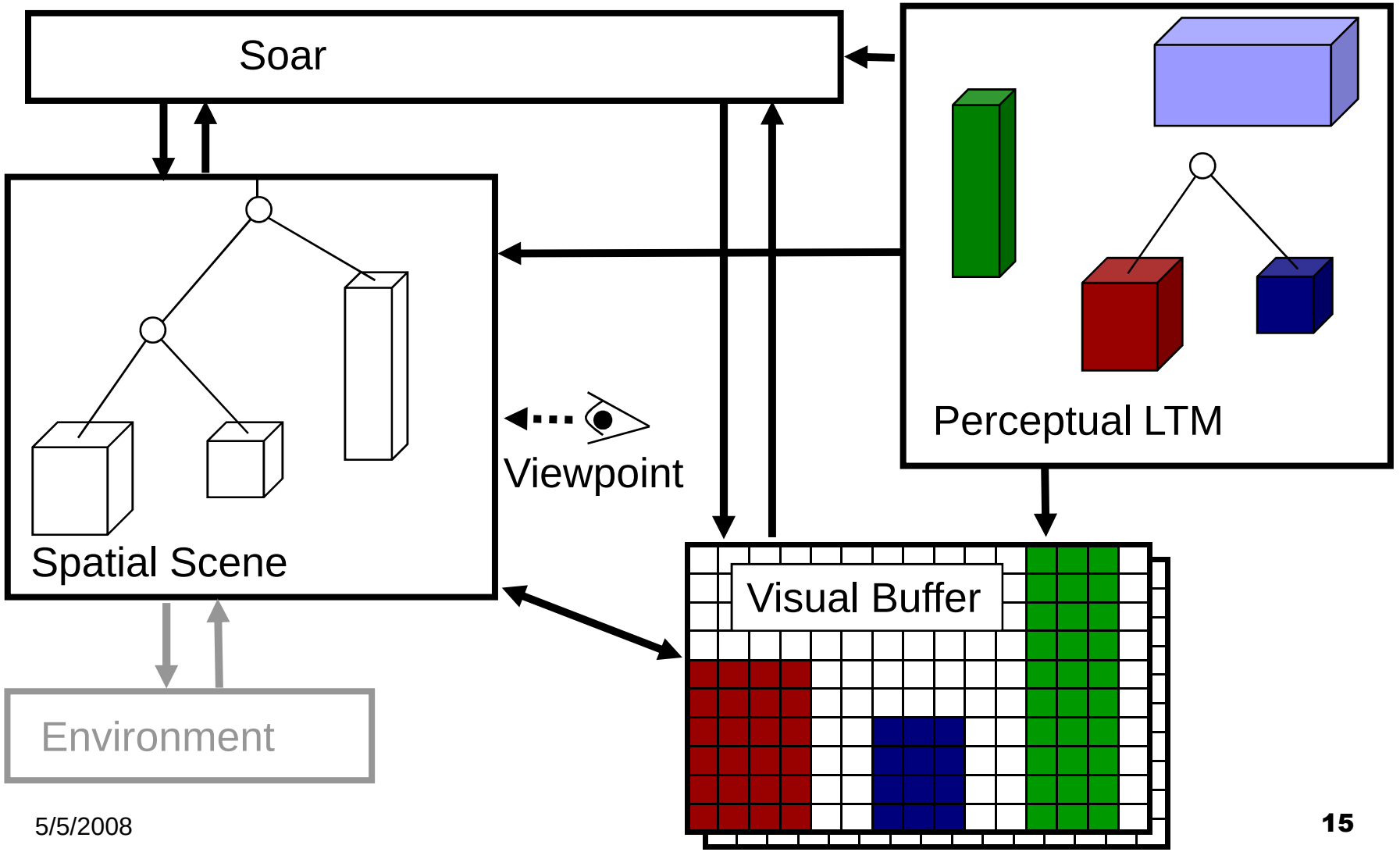


Spatial Scene



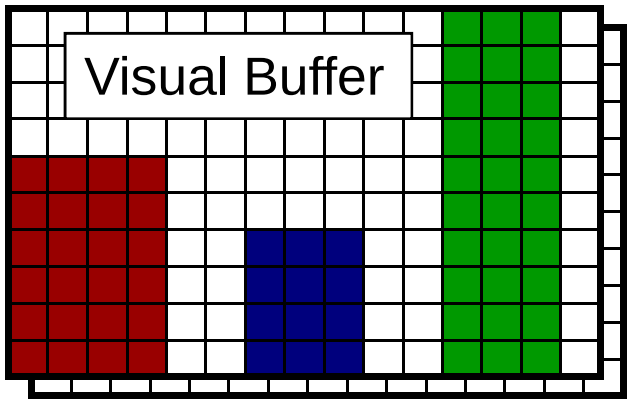
- 3D spatial short-term memory
- Scene graph
  - But grounded in coordinates

High-Level Design

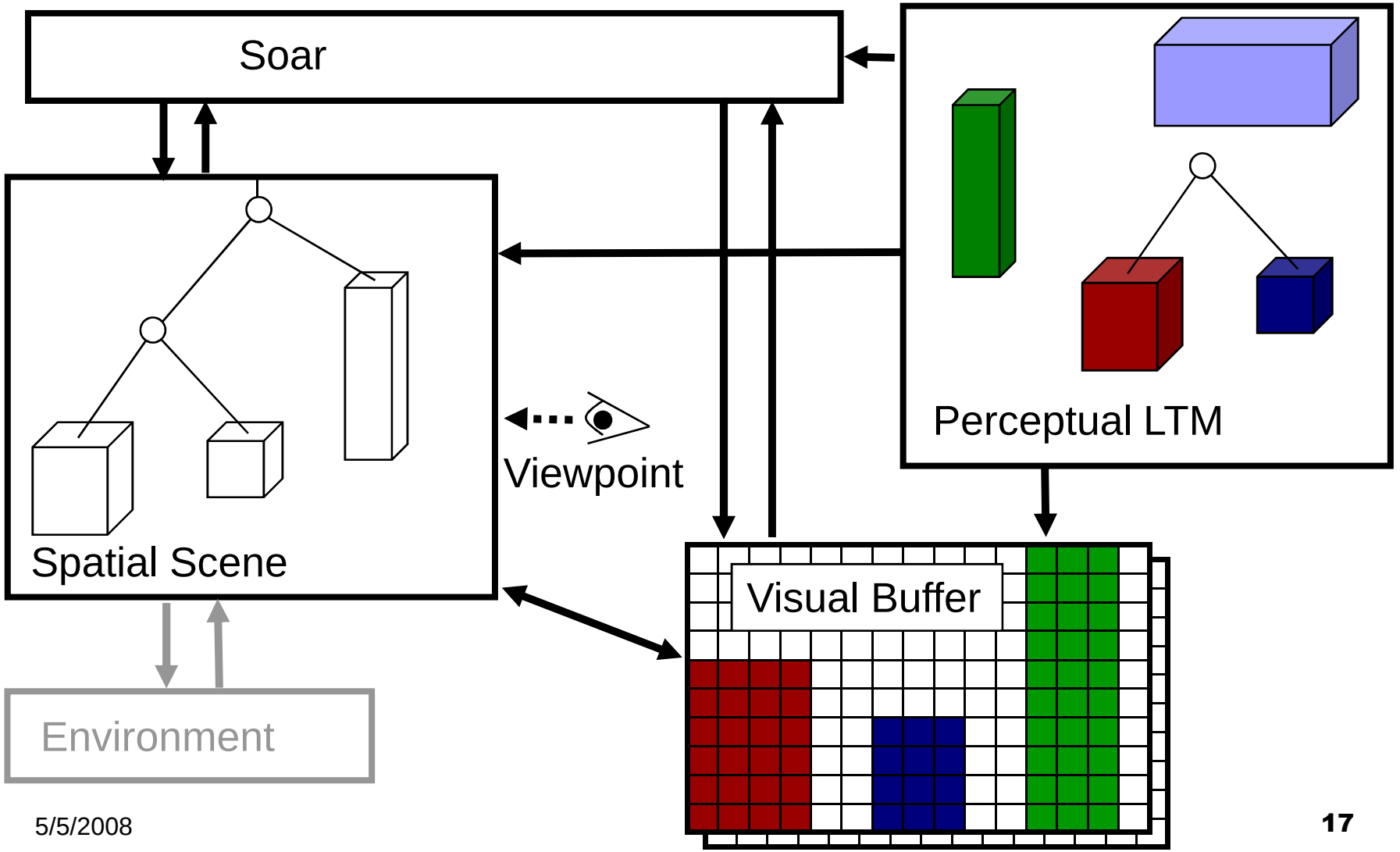


Visual Buffer

- Short-term memory
- Contains a set of 2D, depictive representations
- Derived from the scene, at a given viewpoint
  - Theoretically, the opposite is true



High-Level Design



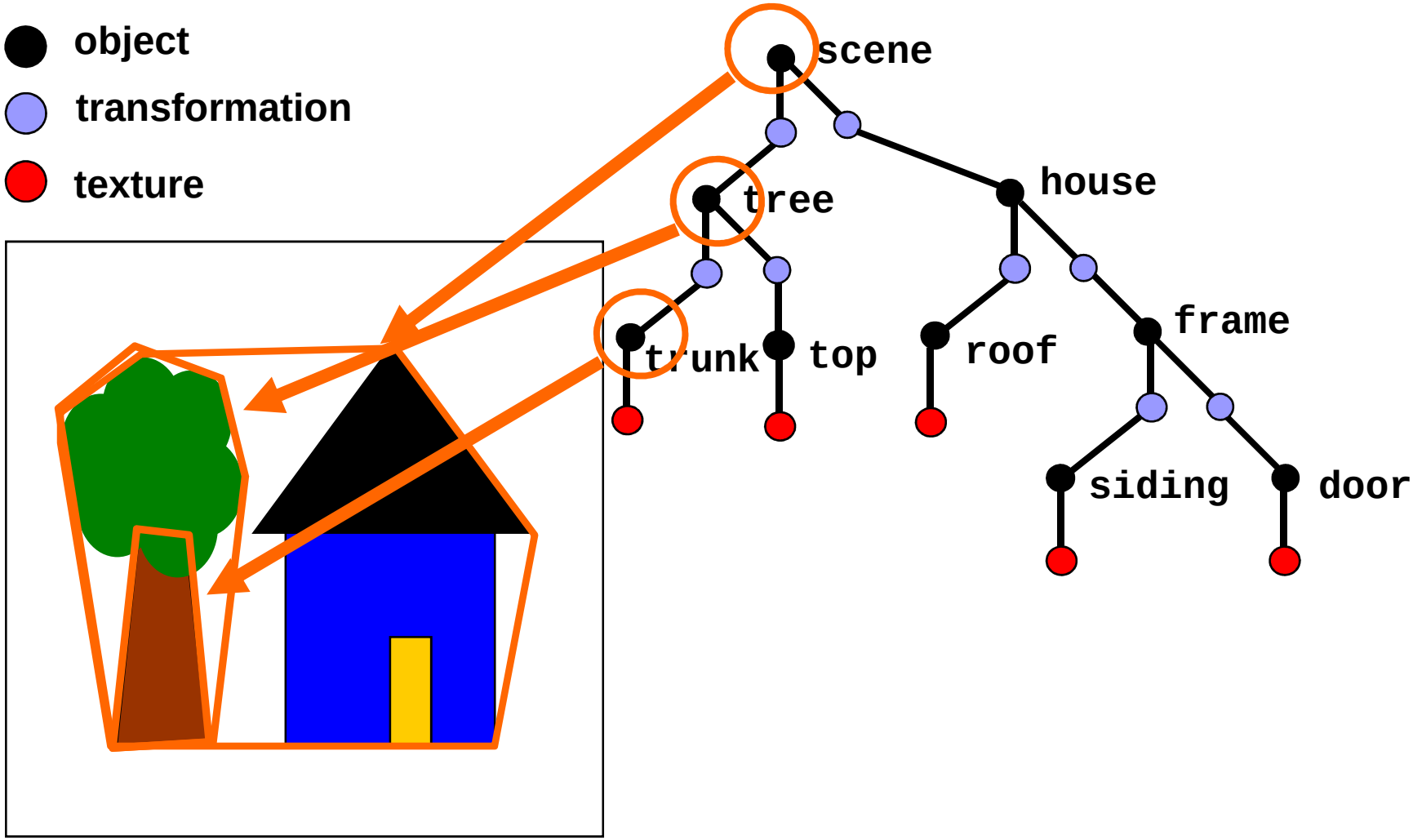
Soar Interface

- Interfaces are (mostly) qualitative: all quantitative information stays within SVS
- SVS provides contents of all memories to Soar
  - Scene graphs are expandable/collapsable
- Soar can add new information to SVS
  - Generate new objects in the scene, and new depictions in the visual buffer
- Soar can retrieve information from SVS
  - Extract predicates from the scene and visual buffer through queries

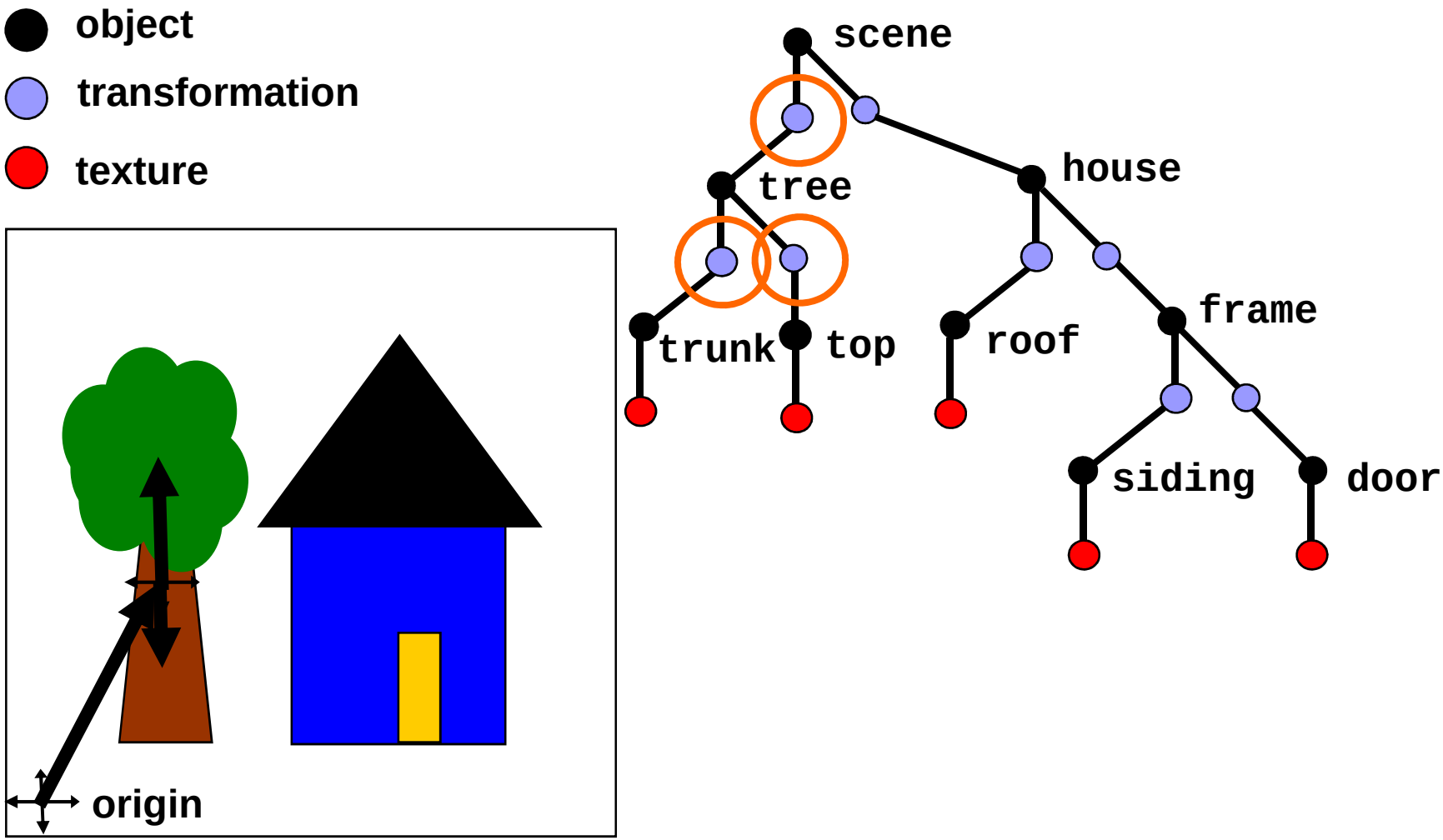
Outline

- Motivation
- Design
- SVS Scene Graphs
- Spatial Generation and Extraction
- Visual Generation and Extraction
- Example Problem
- Nuggets and Coal

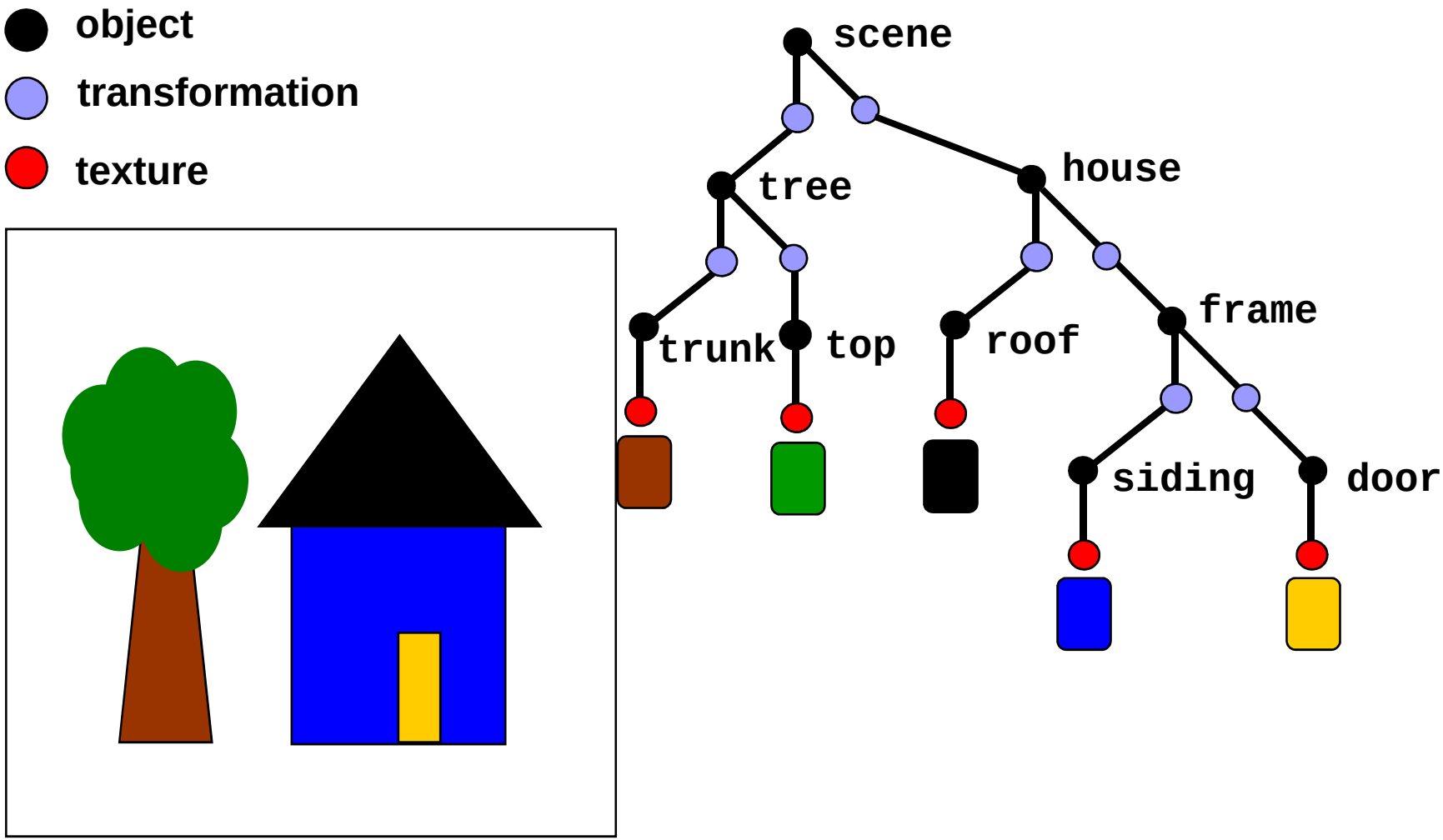
Scene Graph Example: Objects



Scene Graph Example: Transformations



Scene Graph Example: Textures



Outline

- Motivation
- Design
- SVS Scene Graphs
- Spatial Generation and Extraction
- Visual Generation and Extraction
- Example Problem
- Nuggets and Coal

Spatial Generation

- SVS supports imagery where Soar directly provides a scene graph
  - Scene graphs are composable, any valid combination of nodes yields a scene
  - These operations are called “graph generations”
- SVS also supports imagery where objects and/or transformations are qualitatively described
  - Scene graph is implicit in the description
  - These operations are called “projections”
- Motion models allow SVS to represent movement (my other talk)
- All images (spatial and visual) exist as long as the command is on the output-link
  - Images can be manipulated by changing the command in place

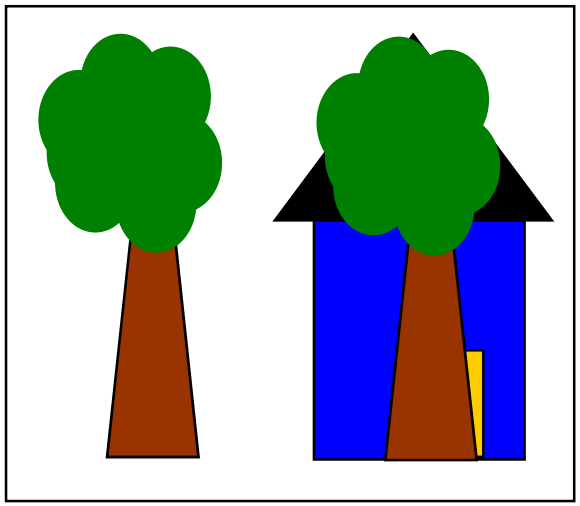
Accessing Long-Term Memory

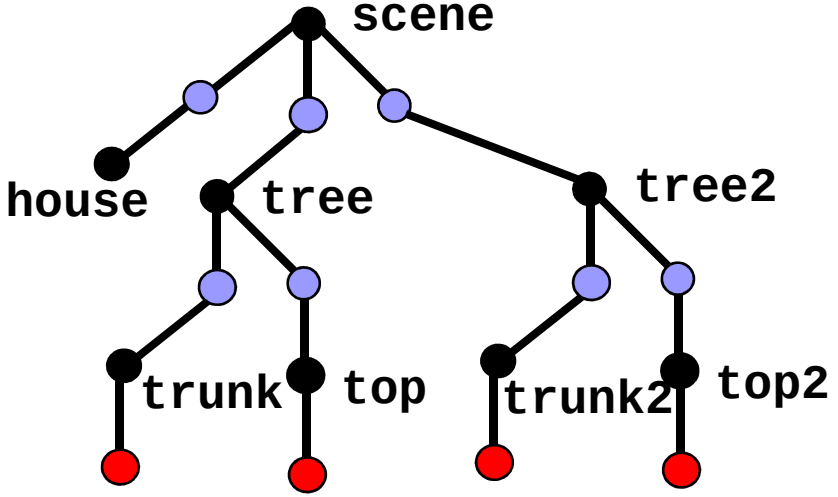
- Items in LTM have a ^class-id
  - e.g. car, car37, car-transform37, car-texture37
- Items in the scene have a ^class-id and an ^instance-id
  - e.g. ^class-id car, ^instance-id car:i23
- A retrieval from LTM happens when a ^class-id appears in a generate command
  - Objects are copied all the way down to the leaves
  - Graph generations can access LTM nodes identically to nodes in the scene

Spatial Graph Generation Example

```
^generate-spatial
  ^transformation
    ^parent scene
    ^source house-trans
    ^new-id tree-trans2
    ^child-new-id tree2
```

```
^object
  ^source tree
  ^new-id tree2
```

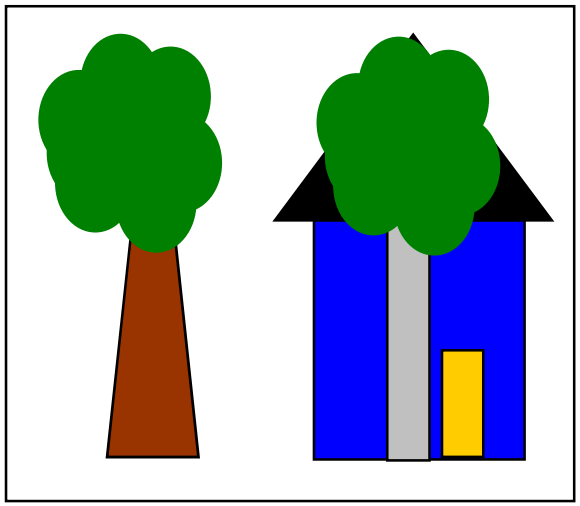


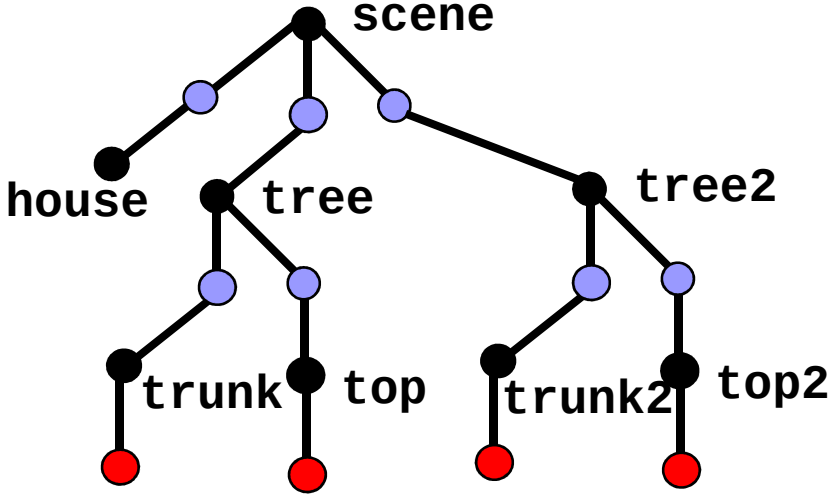


Spatial Graph Generation Example

```
^generate-spatial
  ^transformation
    ^parent scene
    ^source house-trans
    ^new-id tree-trans2
    ^child-new-id tree2
```

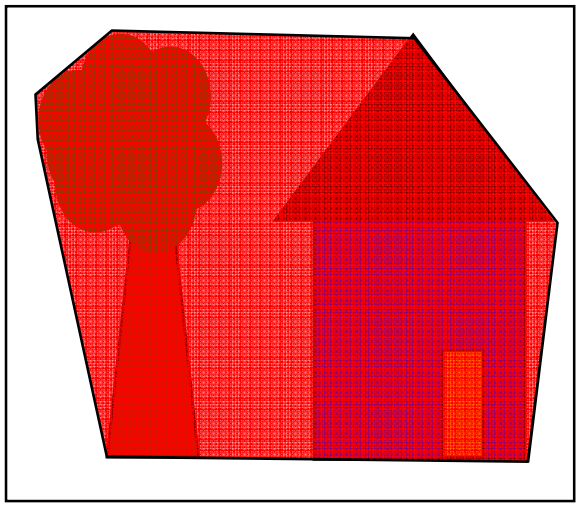
```
^object
  ^source-class birch-tree
  ^new-id tree2
```

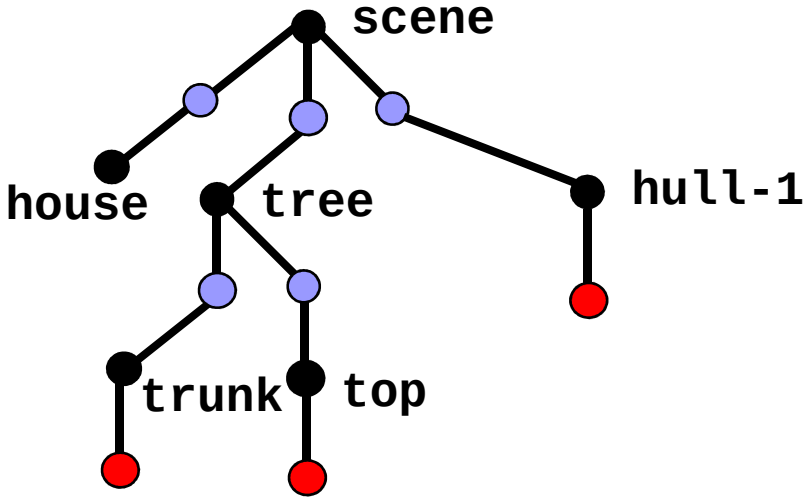




Spatial Projection Example

```
^generate-spatial
  ^projection
    ^hull
      ^instance-id tree
      ^instance-id house
    ^transform-parent scene
    ^texture-source.class-id red
```

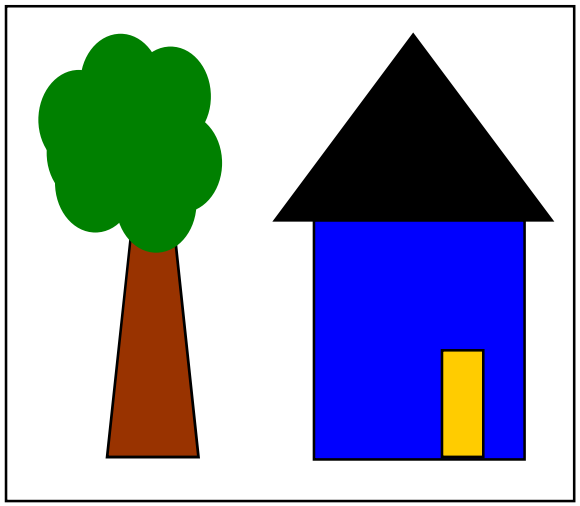


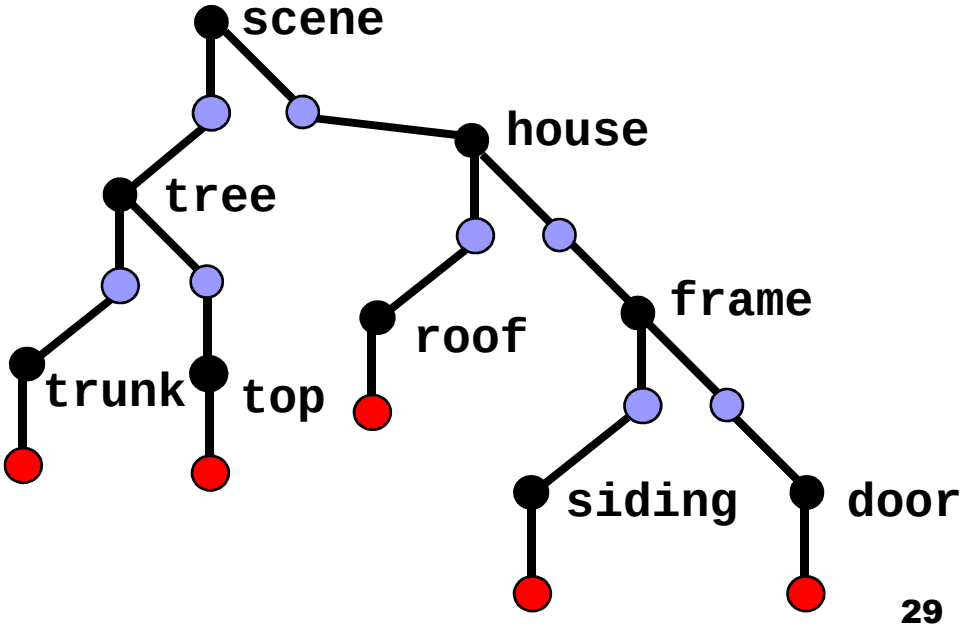


Spatial Extraction Example

```
^extract-spatial
  ^relationship adjacent
  ^primary-object frame
  ^reference-object ?
  ^value true
```

```
^extract-spatial-result
  ^retrieved  *5
    ^relationship adjacent
    ^primary-object frame
    ^reference-object {door, siding,
                       roof, house, scene}
    ^value true
```





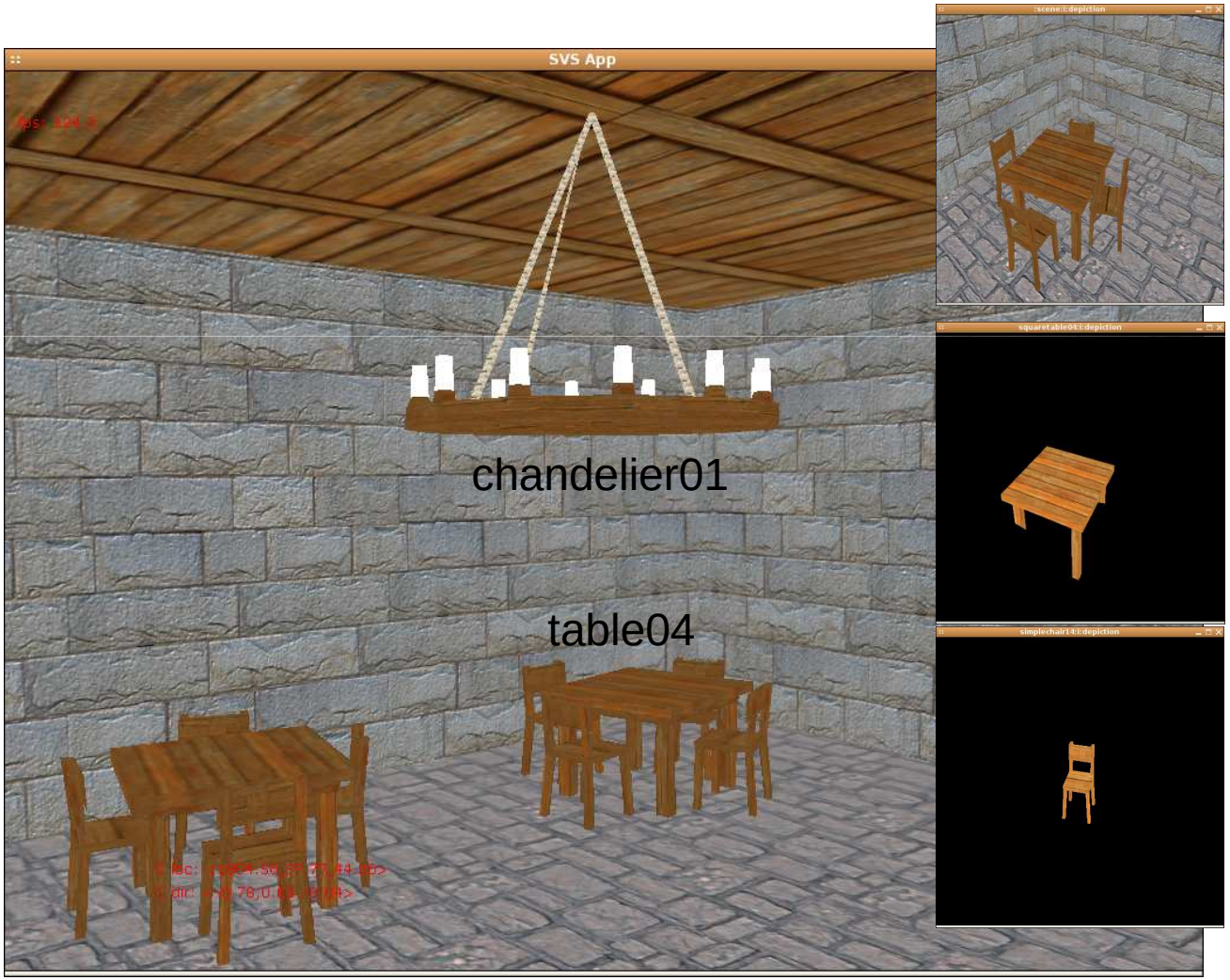
Outline

- Motivation
- Design
- SVS Scene Graphs
- Spatial Generation and Extraction
- Visual Generation and Extraction
- Example Problem
- Nuggets and Coal

Visual Generation

- Visual generation commands add depictions to the visual buffer
- The scene generator renders all or part of the spatial scene to the VB, from a given camera position
- Other visual generators make new depictions without accessing the spatial scene
  - Depictive manipulation (Scott Lathrop's talk) is a subset of this

Visual Scene Generation

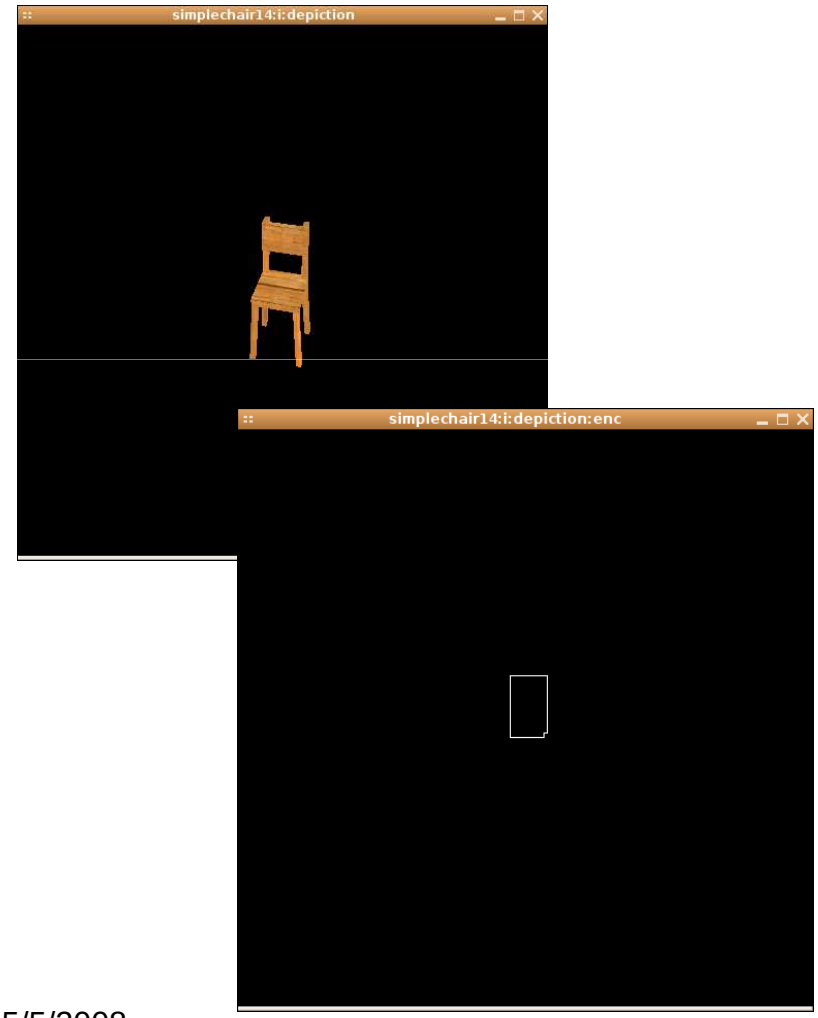


from the perspective of chandelier01 looking towards table04, generate the scene

from the perspective of chandelier01 looking towards table04, generate table04

from the perspective of chandelier01 looking towards table04, generate chair14, ignoring table04

Visual Generation and Extraction



generate a depiction outlining the enclosed space in chair14:depiction

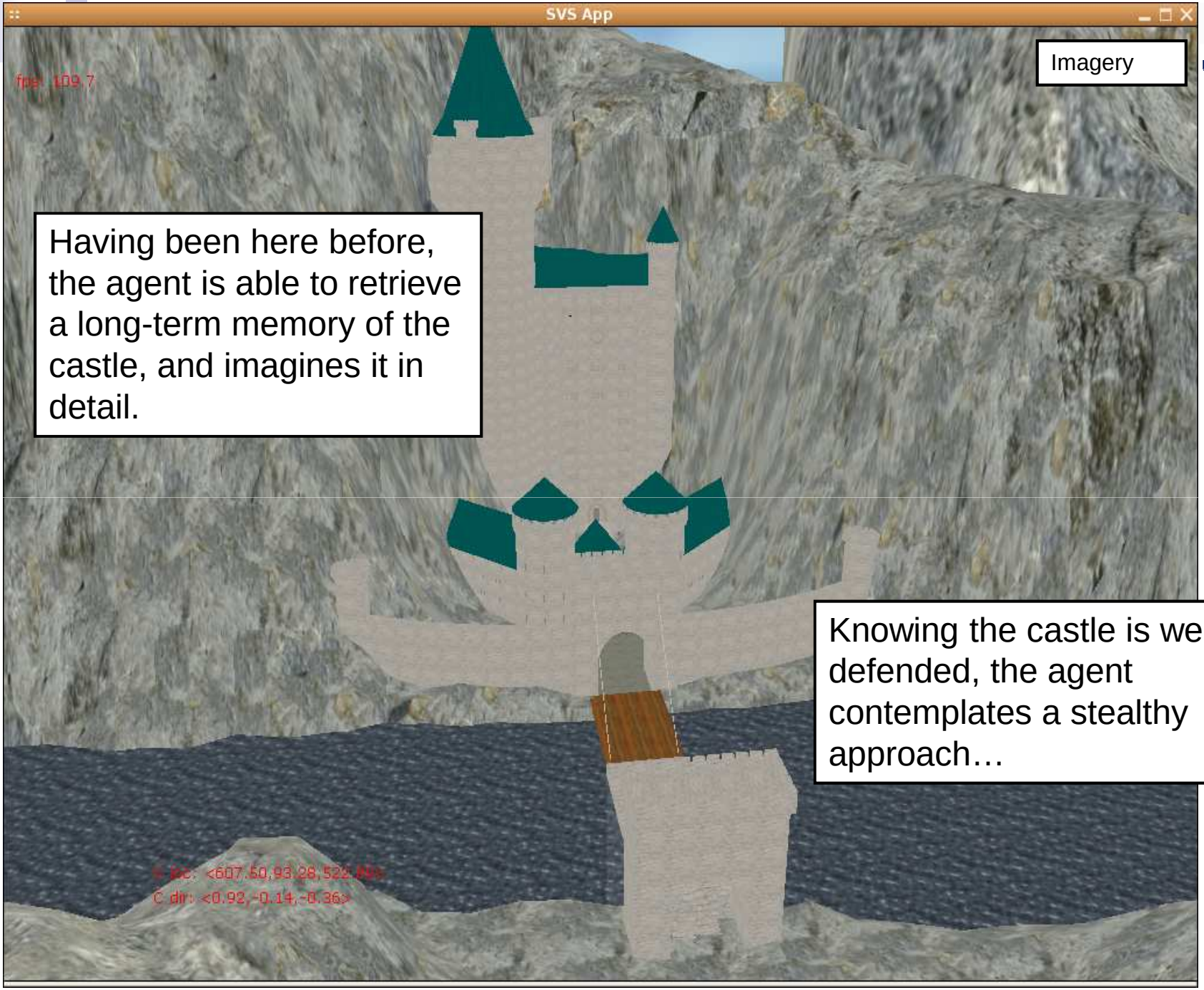
query: are any pixels in chair14:depiction:enc filled in?

result: yes

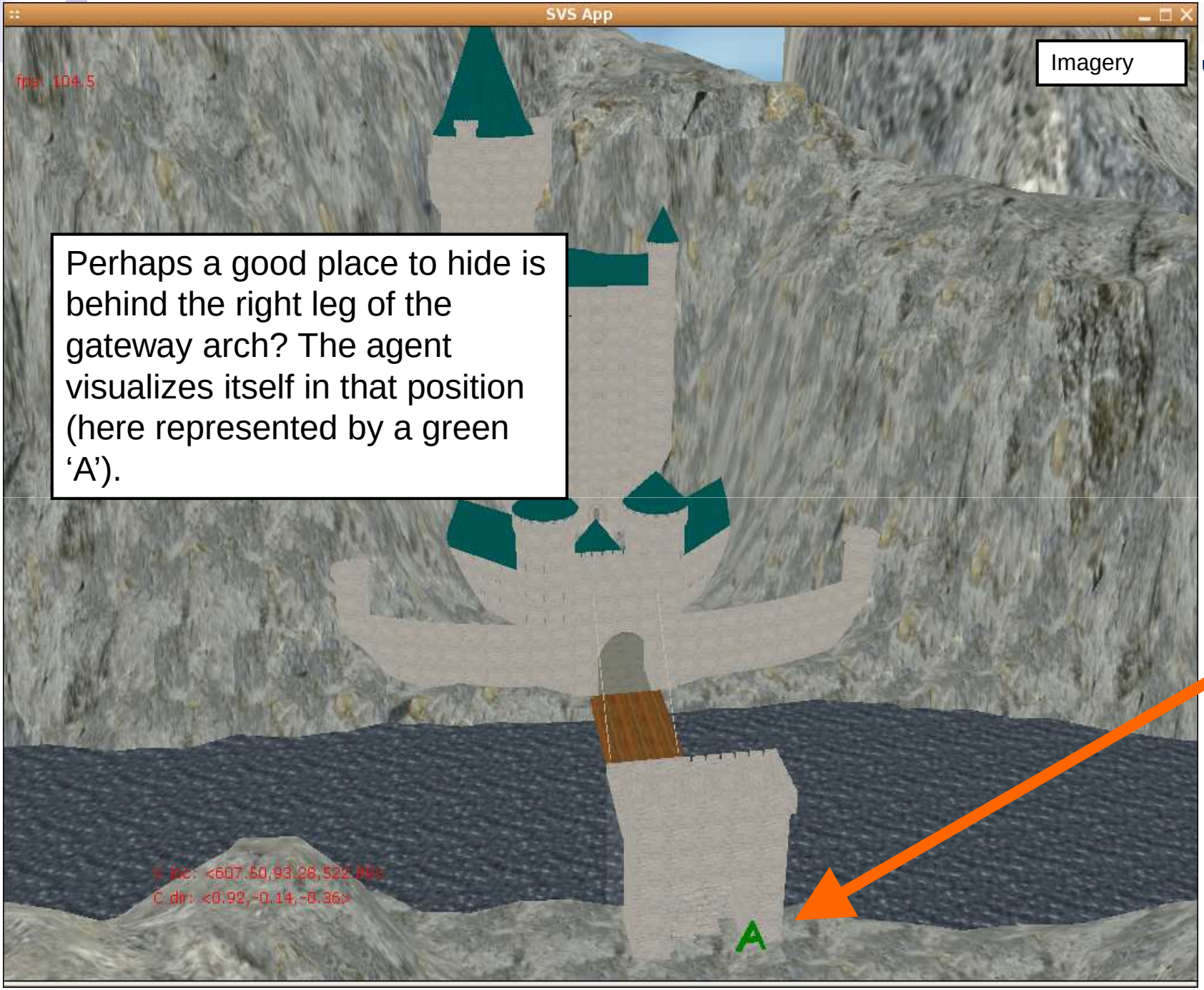
Outline

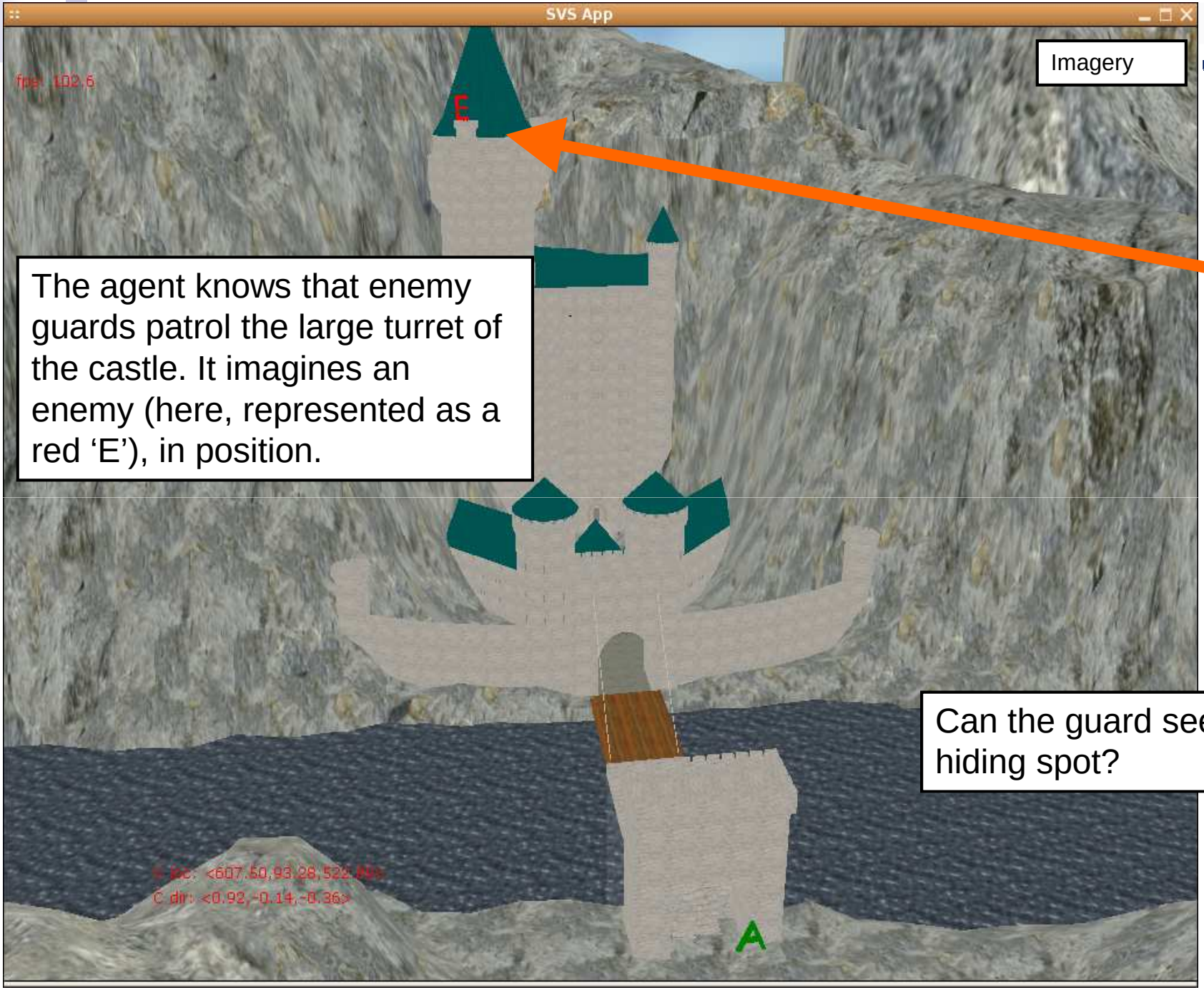
- Motivation
- Design
- SVS Scene Graphs
- Spatial Generation and Extraction
- Visual Generation and Extraction
- Example Problem
- Conclusion, Nuggets and Coal

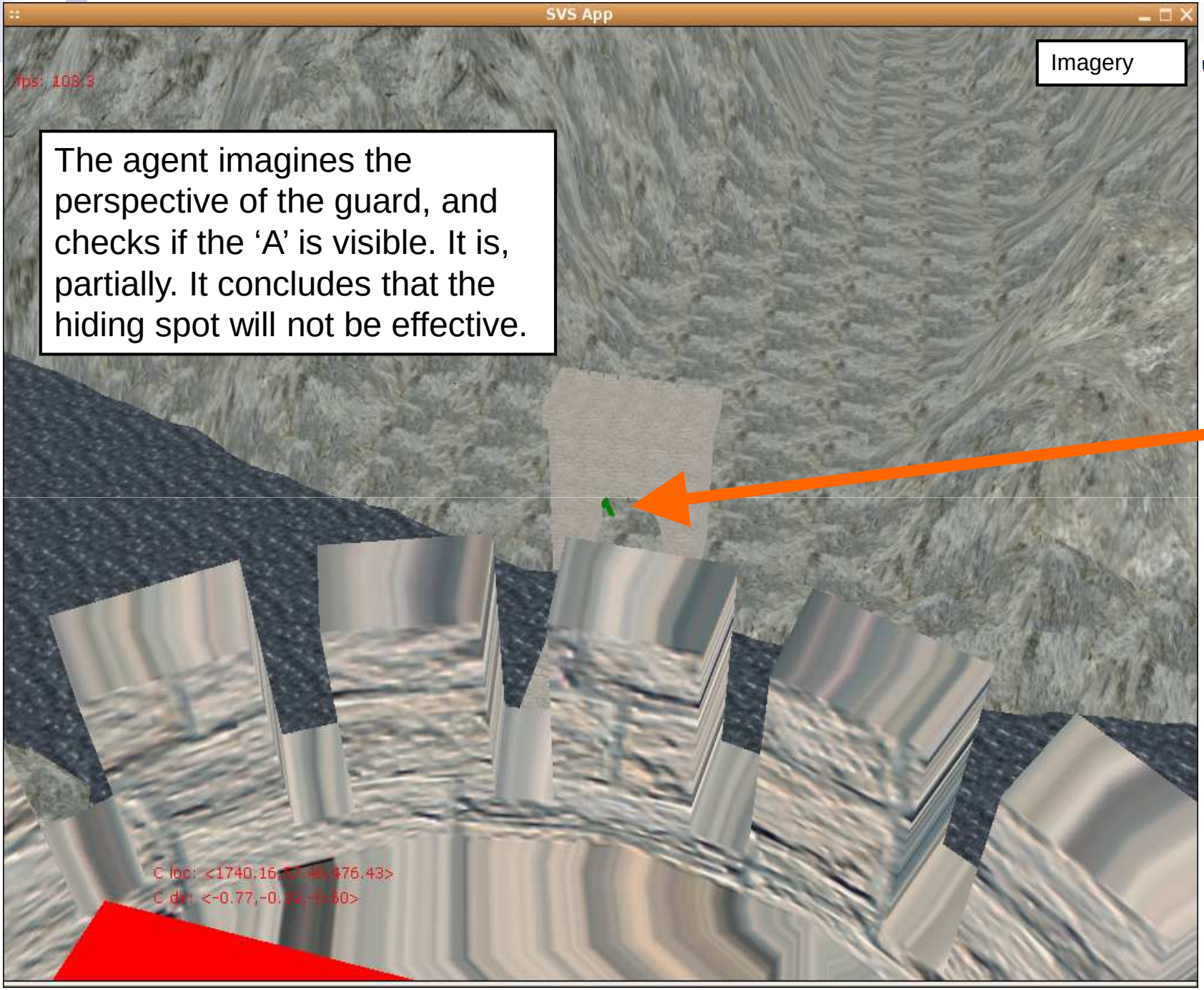
University of Michigan



University of Michigan


SVS App
Imagery
Perhaps a good place to hide is behind the right leg of the gateway arch? The agent visualizes itself in that position (here represented by a green 'A').
A


SVS App
Imagery
E
A
The agent knows that enemy guards patrol the large turret of the castle. It imagines an enemy (here, represented as a red ‘E’), in position.
Can the guard see the hiding spot?



University of Michigan

Nuggets and Coal

- Nuggets
  - Architecture seems to be becoming simpler, not more complex
  - Soar now has direct access to the scene graph
  - Episodic memory integration is close to being possible

- Coal
  - Requires the environment (or low-level perceptual system) to provide a scene graph
  - Integration with environments still requires lots of custom code
  - Scene graph representation in SVS is dictated by graphical modelers and their tools
  - Implementation isn't done yet