KAIST

CS 380

Introduction to Computer Graphics

LAB (7)

2015.04.15

VISUAL COMPUTING Lab

Goals

KAIST

- Draw two robots in scene using hierarchical structure (scene graph)
- Implement “part picking algorithm”
- Transform any part with keeping hierarchical structure

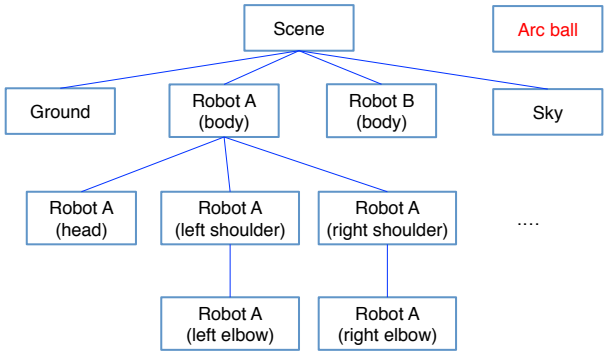
2

VISUAL COMPUTING Lab

Scene Graph

KAIST

- Whole scene = 1 tree structure



3

VISUAL COMPUTING Lab

Scene Graph

KAIST

- Two kinds of nodes on scene graph
  - Transform nodes
    - RBT with respect to its parent frame

$$\begin{aligned} \vec{o}^t &= \vec{w}^t O \\ \vec{s}^t &= \vec{o}^t S \\ \vec{l}^t &= \vec{s}^t L \end{aligned}$$

  - Shape nodes
    - Matrix4 (AffineMatrix) : geometry to be drawn

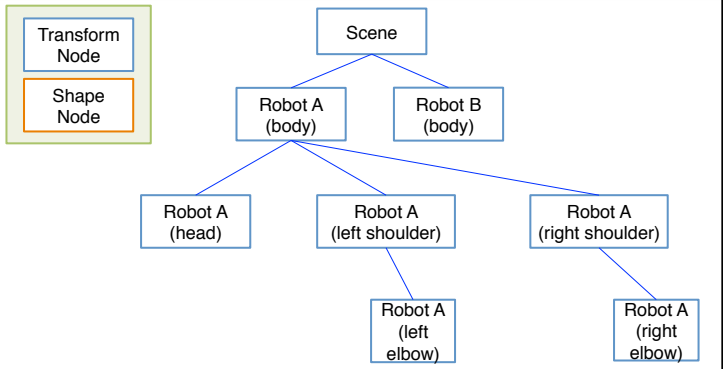
$$\vec{b}^t = \vec{l}^t B = \vec{l}^t \cdot \mathrm{Trans} \cdot \mathrm{Scale}$$

4

VISUAL COMPUTING Lab

Scene Graph

KAIST

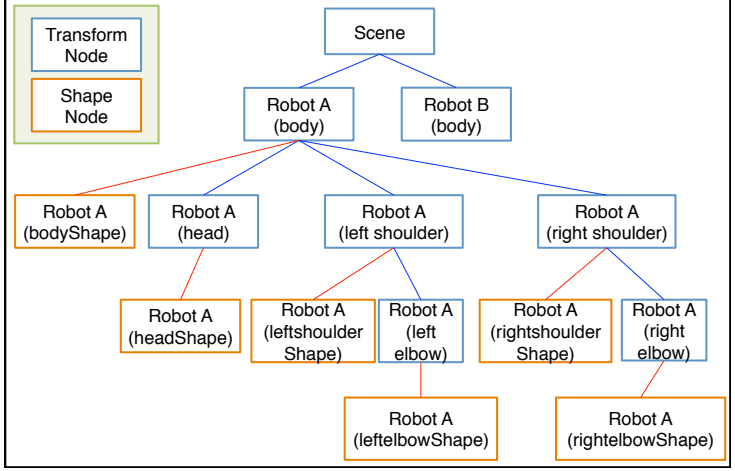


5

VISUAL COMPUTING Lab

Scene Graph

KAIST



Code Migration

KAIST

- Please read pdf file and description in codes carefully
- 'asst4-snippets.cpp' can help you for modifying your asst4.cpp code
  - Construct the scene graph
  - Draw the scene graph

7

VISUAL COMPUTING Lab

Visitor (SgNodeVisitor)

KAIST

- Class for easy traversal on the scene graph
- We need to do various operations through the nodes of scene graph

```
class SgNodeVisitor {
public:
  virtual bool visit(SgTransformNode& node);
  virtual bool visit(SgShapeNode& node);

  virtual bool postVisit(SgTransformNode& node);
  virtual bool postVisit(SgShapeNode& node);
};
```

8

VISUAL COMPUTING Lab

Visitor (SgNodeVisitor)

KAIST

- There are three types of visitor
  - Drawer, Picker and RbtAccumVisitor
  - Each visitor has different role, variables and functions (visit( ), postVisit( ), ..)

```
class Drawer : public SgNodeVisitor {
protected:
  std::vector<RigTForm> rbtStack_;
```

```
class Picker : public SgNodeVisitor {
  std::vector<std::tr1::shared_ptr<SgNode> > nodeStack_;
```

```
class RbtAccumVisitor : public SgNodeVisitor {
protected:
    vector<RigTForm> rbtStack_;
```


VISUAL COMPUTING Lab

accept( ) and visit( )

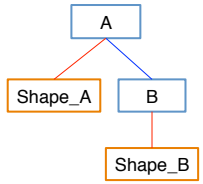
KAIST

Each node has accept( )

: apply visit( ) for itself (node) and pass the visitor to its children

node's accept() function

```
bool SgTransformNode::accept(SgNodeVisitor& visitor) {
  if (!visitor.visit(*this))
    return false;
  for (int i = 0, n = children_.size(); i < n; ++i) {
    if (!children_[i]->accept(visitor))
      return false;
  }
  return visitor.postVisit(*this);
}
```



Each visitor has visit( )

: visit( ) do some works on certain node

VISUAL COMPUTING Lab

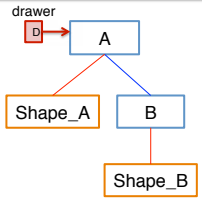
accept( ) and visit( )

KAIST

Example

A->accept(drawer)

Operation



drawer's visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

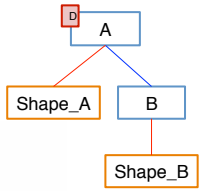
Example

A->accept(drawer)

drawer.visit(A)

Operation

Stack = {A}



drawer's visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

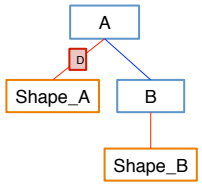
A->accept(drawer)

drawer.visit(A)

pass ‘drawer’ to Shape_A

Operation

Stack = {A}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

A->accept(drawer)

drawer.visit(A)

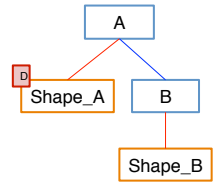
pass ‘drawer’ to Shape_A

drawer.visit(Shape_A)

Operation

Stack = {A}

draw{A * Shape_A}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

A->accept(drawer)

drawer.visit(A)

pass ‘drawer’ to Shape_A

drawer.visit(Shape_A)

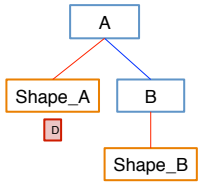
pass ‘drawer’ to no child

Operation

Stack = {A}

draw{A * Shape_A}

None, Stack = {A}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

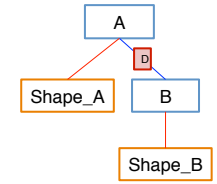
KAIST

Example

pass ‘drawer’ to B

Operation

Stack = {A}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

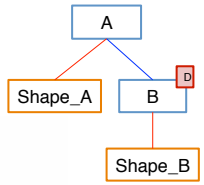
pass ‘drawer’ to B

drawer.visit(B)

Operation

Stack = {A}

Stack = {AB}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

pass ‘drawer’ to B

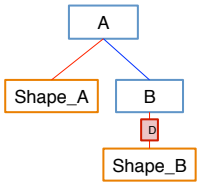
drawer.visit(B)

pass ‘drawer’ to Shape_B

Operation

Stack = {A}

Stack = {AB}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

pass ‘drawer’ to B

drawer.visit(B)

pass ‘drawer’ to Shape_B

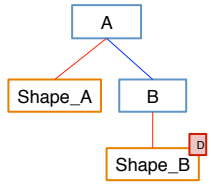
drawer.visit(Shape_B)

Operation

Stack = {A}

Stack = {AB}

draw{AB * Shape_B}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

accept( ) and visit( )

KAIST

Example

pass ‘drawer’ to B

drawer.visit(B)

pass ‘drawer’ to Shape_B

drawer.visit(Shape_B)

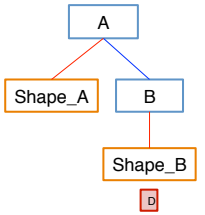
pass ‘drawer’ to no child

Operation

Stack = {A}

Stack = {AB}

draw{AB * Shape_B}



drawer’s visit() function

```
virtual bool visit(SgTransformNode& node) {
  rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  return true;
}

virtual bool visit(SgShapeNode& shapeNode) {
  const Matrix4 MVM = rigTFormToMatrix(rbtStack_.back()) * shapeNode.getAffineMatrix();
  sendModelViewNormalMatrix(curSS_, MVM, normalMatrix(MVM));
  shapeNode.draw(curSS_);
  return true;
}
```

Part Picking

KAIST

- If picking mode is on,
  - do not swap front buffer and back buffer
  - render the different scene on back buffer using colors defined by each object's ID
  - do not shade according to light direction
- If some object is picked,
  - you can distinguish which object is picked using back buffer's color
  - redisplay(swap) the scene and new arc ball
- Refer to Picker class

VISUAL COMPUTING Lab

Transform Any Part

KAIST

- Transform any part with keeping hierarchical structure using scene graph
- Refer to RbtAccumVisitor class

VISUAL COMPUTING Lab

Transform Any Part

KAIST