CMPS385: Assignment 4

Authors:

Ahmad Ghizzawi and Chukri Soueidi.

Run Instructions

```
$ make
$ ./main
```

Changes

Task 1

We integrated the code from asst4-snippets.cpp as described in the document. This snippet shows how we are using scenegraph to draw:

```
static void drawStuff(const ShaderState& curSS, bool picking) {
  .
  .
  .
  // draw robots, and ground
  // ==========
  //

  if (!picking) {
      Drawer drawer(invEyeRbt, curSS);
      g_world->accept(drawer);
  }
  else {
    Picker picker(invEyeRbt, curSS);
    g_world->accept(picker);
    glFlush();
    g_currentPickedRbtNode = picker.getRbtNodeAtXY(g_mouseClickX, g_mouseClickY);
    if (g_currentPickedRbtNode == g_groundNode)
      g_currentPickedRbtNode = g_skyNode;
  }

  // draw arcball
  // ==========
  //
  if (isArcballActive()) {
      if (!g_mouseMClickButton && !(g_mouseLClickButton && g_mouseRClickButton) &&
!(g_spaceDown)) {
        g_arcballScale = getScreenToEyeScale(
             (inv(getEyeRbt()) * getArcballRbt()).getTranslation()[2],
             g_frustFovY,
             g_windowHeight
          );
```

```
      }

      drawArcball(curSS, invEyeRbt);
  }
}
```

Task 2: Picker

When visiting a shape node, we implemented the following:

```
bool Picker::visit(SgShapeNode& node) {
  // increment id counter
  idCounter_++;
  // Get the parent node (which should be an SgRbtNode object)
  shared_ptr<SgRbtNode> parent =
dynamic_pointer_cast<SgRbtNode>(nodeStack_.back());
  // Associate the id with the parent
  addToMap(idCounter_, parent);
  // Retrieve the color of the object based on the id
  Cvec3 objectColor = Picker::idToColor(idCounter_);
  // Set the uniform variable to have the given color before drawing.
  safe_glUniform3f(drawer_.getCurSS().h_uIdColor, objectColor[0], objectColor[1],
                   objectColor[2]);

  return drawer_.visit(node);
}
```

We also called the pick() function provided in the asst4-snippets.cpp file in the bottom of the mouse function. Notice that we added a new global boolean variable g_pickObject that is set true whenever the key 'p' is clicked, which allows this condition to be executed.

```
static void mouse(const int button, const int state, const int x, const int y) {
  ...
  // left button down?
  if (g_mouseLClickButton && !g_mouseRClickButton && !g_spaceDown && g_pickObject)
{
    pick();
    g_pickObject = false;
  }
}
```

We also implemented all the TODO functionality described in Task 2 description.

Task 3: Transforming

We implemented the TODO functionality inside the RbtAccumVisitor. Here is a snippet of the visit function of the RbtAccumVisitor that we implemented:

```
virtual bool visit(SgTransformNode& node) {
  // TODO
```

```
  if (rbtStack_.empty()) {
    rbtStack_.push_back(node.getRbt());
  } else {
    rbtStack_.push_back(rbtStack_.back() * node.getRbt());
  }

  return !(node == target_);
}
```

We also migrated our code to use getPathAccumRbt() functions for retrieving the matrices required to be able to view from any node. Here is a snippet of the code that does that:

```
// Returns the current eyeRbt based on the active view.
static RigTForm getEyeRbt() {
  switch (g_activeEye) {
  case OBJECT0:
    return getPathAccumRbt(g_world, g_robot1Node);
  case OBJECT1:
    return getPathAccumRbt(g_world, g_robot2Node);
  case SKY:
  default:
    return getPathAccumRbt(g_world, g_skyNode);
  }
}
```

Also, we tuned the Arcball interface so that it works flawlessly. Here is the code that does that:

```
// Arcball Rbt will be based on the current case:
// 1. Center is the world's origin
// 2. Center is the cube being manipulated.
// Returns the RBT of arcball.
static RigTForm getArcballRbt() {
  // Active object is a cube and isn't wrt to itself.
  if (isCubeActive()) {
    return getPathAccumRbt(g_world, g_currentPickedRbtNode);
  }
  // Active object and eye are the SKY camera wrt world-sky frame.
  return g_world->getRbt();
}
```

Finally, we updated the way we transform nodes so that we use getPathAccumRbt(). Here is a small snippet that shows how we did it:

```
// Manipulates objects' RBTs based on the active objects and eye.
static void manipulateObjects(RigTForm &Q) {
  ...
    // Active eye is sky and active object is a cube.
    if (isCubeActive() && g_activeEye == SKY) {
      RigTForm A = makeMixedFrame(g_currentPickedRbtNode->getRbt(),
g_currentView->getRbt());
      g_currentPickedRbtNode->setRbt(doQtoOwrtA(Q,
```

```
g_currentPickedRbtNode->getRbt(), A));
    }
  ...
}
```

Task 4: Building the robot.

We constructed a robot that includes a head, left/right upper and lower arms, and left/right upper and lower legs. Here is the code snippet that we wrote to build the robot (taken from constructRobot function in main.cpp):

```
ShapeDesc shapeDesc[NUM_SHAPES] = {
  {0, 0,         0, 0, TORSO_WIDTH, TORSO_LEN, TORSO_THICK, g_cube}, // torso
  {1, ARM_LEN/2, 0, 0, ARM_LEN, ARM_THICK, ARM_THICK, g_cube}, // upper right arm
  {2, ARM_LEN/2, 0, 0, ARM_LEN, ARM_THICK, ARM_THICK, g_cube}, // lower right arm
  {3, 0, HEAD_RAD, 0, HEAD_RAD, HEAD_RAD, HEAD_RAD, g_arcball}, // head
  {4, -ARM_LEN/2, 0, 0, ARM_LEN, ARM_THICK, ARM_THICK, g_cube}, // upper left arm
  {5, -ARM_LEN/2, 0, 0, ARM_LEN, ARM_THICK, ARM_THICK, g_cube}, // lower left arm
  {6, 0, -LEG_LEN/2, 0, LEG_THICK, LEG_LEN, LEG_THICK, g_cube}, // upper right Leg
  {7, 0, -LEG_LEN/2, 0, LEG_THICK, LEG_LEN, LEG_THICK, g_cube}, // lower right Leg
  {8, 0, -LEG_LEN/2, 0, LEG_THICK, LEG_LEN, LEG_THICK, g_cube}, // upper left Leg
  {9, 0, -LEG_LEN/2, 0, LEG_THICK, LEG_LEN, LEG_THICK, g_cube}, // lower left Leg
};


JointDesc jointDesc[NUM_JOINTS] = {
  {-1}, // torso
  {0,  TORSO_WIDTH/2, TORSO_LEN/2, 0},  // upper right arm
  {1,  ARM_LEN, 0, 0},                  // lower right arm
  {0, 0, TORSO_LEN/2, 0},               // head
  {0,  -TORSO_WIDTH/2, TORSO_LEN/2, 0}, // upper left arm
  {4,  -ARM_LEN, 0, 0},                 // lower left arm
  {0,  TORSO_WIDTH/2, -TORSO_LEN/2, 0}, // upper right leg
  {6,  0, -LEG_LEN, 0},                 // lower right leg
  {0,  -TORSO_WIDTH/2, -TORSO_LEN/2, 0},// upper left leg
  {8,  0, -LEG_LEN, 0}                  // lower left leg
};
```

Known Issues:

- When using the spacebar + right mouse click combo to translate in the Z-directions, the behavior is not the expected one.
- When picking the sky using the picker interface, an error occurs. To reproduce the error, press 'p' and then click on the blue section of the screen. However, you can select the sky by clicking on the ground instead.