CS 2334: Project 5
Java Graphics
Project 4 Lessons
Project 4 Lessons

• Get layout working first, then interaction
• May not have anything selected by a JList at a given instant in time: code must be robust to this
• Tracking down Exceptions in “tall” stack traces
Animate the infant:
• Show top, side and rear view of the infant
• Allow user to select time instant to be displayed
• Animation using Timers
• Demonstration ....
Objectives

Create a Graphical User Interface for Animation
• Create a tree data structure for representing a kinematic tree
• Create a class for flexibly transforming 3D data into a 2D picture
• Use JSliders to accept input
• Use Timers to create animations
• Continue to exercise good coding practices for Javadoc and for testing Drop-down menus (JMenuBar)
Kinematic Tree

Representation of a biological or robot body. In our case:

• Rigid **links** connected by rotational **joints**
• Root point: base of the tree
  • For us: center point between hips

Sanini et al., 2015
Kinematic Tree

A single point can branch along multiple paths

• In this figure: the root point branches in three directions:
  • Left hip
  • Right hip
  • Point between the shoulder blades (the back)

We refer to these as the children of the root point

Sanini et al., 2015
Drawing a Kinematic Tree

Assume that we know the locations of all of the points

1. Start at the root point
2. For each child point:
   1. Draw a line from this point to the child
   2. Recursively draw the child

Sanini et al., 2015
Infant Model

• Our data structure contains most of the points we need within a single State
  • Back, shoulders, elbows, wrists, ...

• Some points are not defined, but they are fixed
  • Location of the hip sockets
  • Location of the small of the back
Infant Kinematic Model

ROOT (0,0,0)

LOWER BACK (.1,0,0)

RIGHT HIP (0, -.05, 0)

right_knee

right_ankle

right_foot

upper_back

right_shoulder

right_elbow

right_wrist
Rendering

We are not working in 3D, here. Instead: we are creating simple 2D projections

• Top view: Map X to screen X and Y to screen Y (flipped)
• Side view: Map X to screen X and Z to screen Y (flipped)
• Rear view: Map Y to screen X (flipped) and Z to screen Y (flipped)

Scaling: we must also translate from meters to pixels
KinematicPoint

A KinematicPoint must be able to:

• Describe the location of some specified dimension (subfield)
• Given subfields for each of the screen X and screen Y dimensions, draw the point and its children
Drawing a Single KinematicPointAbstract

• Extract the GeneralValue for the screen X and screen Y subfields

• For each child:
  • Extract the GeneralValue for the screen X and screen Y subfields
  • If all GeneralValues are valid, then draw a line from this point to the child point (specifically, we are drawing a BasicStroke)
  • Draw the child
KinematicPanel

JPanel that renders a single view of the kinematic model
• Maps from 3D point to 2D screen coordinates
• Scale translates from real coordinates (meters) to pixels
• In some cases, flips the sign of the pixel coordinates
Graphical User Interface

• Menu is the same
• Selection of week only
• DataPanel:
  • Three different views
  • Textual information
  • Control of time step to render
```plaintext
InfantFrame
- selectionPanel: SelectionPanel
- dataPanel: DataPanel
- infant: Infant
- trial: Trial
- COLUMN_FIELD_WIDTH: int
- FONT: Font
+ InfantFrame()
+ loadData(directory: String, infantID: String): void
+ update(): void

SelectionPanel
- trialList: JList<String>
- trialListModel: DefaultListModel<String>
- trialScrollPane: JScrollPane
- trialLabel: JLabel
+ SelectionPanel()
+ updateSelections(): void

FileMenuBar
- menu: JMenu
- menuOpen: JMenuItem
- menuExit: JMenuItem
- fileChooser: JFileChooser
+ FileMenuBar()

DataPanel
- rootPoint: KinematicPointAbstract
- viewPanel: JPanel
- topViewPanel: KinematicPanel
- sideViewPanel: KinematicPanel
- rearViewPanel: KinematicPanel
- textPanel: JPanel

- infantTextField: JTextField
- timeTextField: JTextField

- timeSlider: JSlider
- currentTime: int
- runButton: JButton

- timePanel: JPanel
- timer: Timer

- FIELD_WIDTH: int
- LINE_WIDTH: int
+ DataPanel()
+ setTime(newTime: int)
+ update(state: State)
+ createKinematicModel(): KinematicPointAbstract

KinematicPanel
- flipX: double
- flipY: double
- screenXSubfield: String
- screenYSubfield: String
- rootPoint: KinematicPointAbstract
- state: State
- title: String
```

Deadlines

• Project must be submitted by Friday, Dec 1\textsuperscript{st} @6:00pm
• Code review must be completed by Friday, Dec 8\textsuperscript{th}
  • This is an absolute deadline