Interaction Design

1. Input, Behavior, and Picking

- Input devices
  - Java3D has access to keyboards and mice using the Java API.
  - Java3D also provides access to continuous input devices such as 6 DOF trackers and joysticks via an abstract `InputDevice` interface.
    - Input devices or sensors must be implemented for actual devices.
    - Input data from the sensor data can be read and processed.

- Behavior is a class for specifying animations of or interaction with visual objects.
  - The distinction between animation and interaction is whether the behavior is activated in response to the passing of time or in response to user activities, respectively.
- Mouse interaction
  - Java3D provides 4 utility classes for mouse interaction.
    - Abstract class `MouseBehavior` defines behavior initialization, stimuli processing etc for three subclasses on mouse-based rotation, translation, and zooming.

- The picking API enables selecting objects in the scene
  - Java3D divides picking into two portions
    - Control: clicking with a 2D mouse or move a 6DOF wand
    - Selection: finding shapes that meet search criteria

2. Mouse-based object manipulation

- `MouseRotate` enables the rotation of an object by dragging with the left mouse button.
  - Create a transform group on which the rotate behavior is to operate

```
TransformGroup obj_man = new TransformGroup();
obj_man.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
obj_man.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
```


  - Add the rotate behavior to the transform group to allow the rotation of any object attached to `obj_man`

```
MouseRotate myMouseRotate = new MouseRotate();
myMouseRotate.setTransformGroup(obj_man);
myMouseRotate.setSchedulingBounds(new BoundingSphere());
objRoot.addChild(myMouseRotate);
```

- `MouseTranslate` enables the translation of an object by dragging with the right mouse button.
  - Replacing the rotate with the translate behavior allows the translation of any object attached to `obj_man`

```
MouseTranslate myMouseTranslate = new MouseTranslate();
myMouseTranslate.setTransformGroup(obj_man);
myMouseTranslate.setSchedulingBounds(new BoundingSphere());
```

```
objRoot.addChild(myMouseTranslate);
```

- `MouseZoom` enables the zooming of an object by dragging with the middle mouse button.
  - Replacing the rotate with zoom behavior allows the zooming of any object attached to `obj_man`

```
MouseZoom myMouseZoom = new MouseZoom();
myMouseZoom.setTransformGroup(obj_man);
myMouseZoom.setSchedulingBounds(new BoundingSphere());
objRoot.addChild(myMouseZoom);
```


3. The picking API provides the interactive ability for object manipulation.
   - It supports various selection shapes
   - It can report the first, any, all, or all sorted hits
   - It is designed for speed
     - Picking only works on bounds
     - Utilities provide more fine-grained pick support
   - It is distributed among a number of classes
     - Enable pickability of any node via methods on `Node`
     - Initiate a pick using methods on `Locale` or `BranchGroup`
     - Pick methods take as an argument a `PickShape`, and return one or more `SceneGraphPaths`

4. Picking intersects a `PickShape` with pickable shape bounding volumes

- `PickShape` class hierarchy

Class Hierarchy

```
java.lang.Object
  └ javax.media.j3d.PickShape
      ├ javax.media.j3d.PickBounds
      ├ javax.media.j3d.PickPoint
      ├ javax.media.j3d.PickRay
      └ javax.media.j3d.PickSegment
```

- `PickRay` fires a ray from a position, in a direction
  - Pick occurs for shape bounds the ray strikes
- `PickSegment` fires a ray along a ray segment between two positions

  - Pick occurs for shape bounds the ray segment intersects
- `PickPoint` checks the scene at a position
  - Pick occurs for shape bounds that contain the position
- `PickBounds` checks the scene at a position, in a bounded volume
  - Pick occurs for shape bounds that intersect the bounded volume

5. The pick methods on `Locale` or `BranchGroup` return one or more `SceneGraphPaths`

   - `SceneGraphPath` extends `Object`

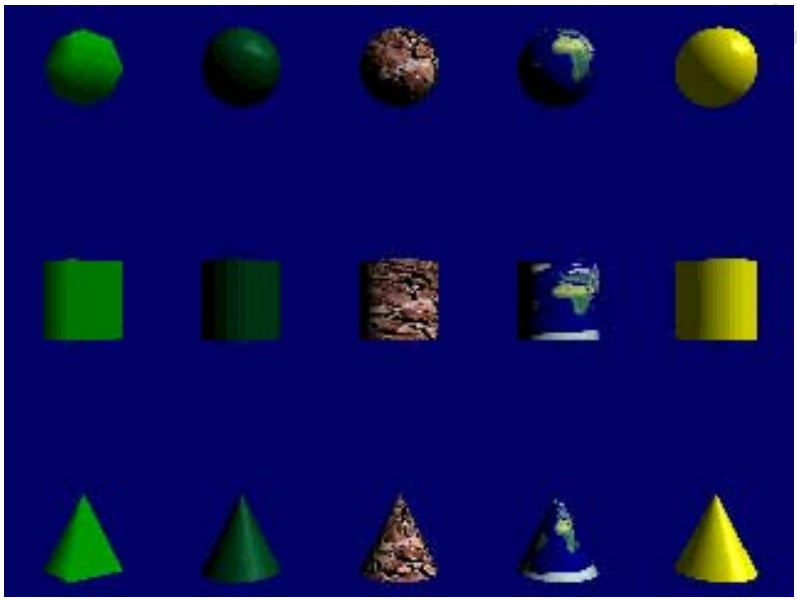
Class Hierarchy
java.lang.Object **└ javax.media.j3d.SceneGraphPath**

   - Each `SceneGraphPath` contains:
     - A `Node` for the shape that was picked
     - The `Locale` above it in the scene graph
     - A list of the `Nodes` from the picked shape up to the `Locale`
     - The world-to-shape transform

6. Using the mouse for a pick requires the creation of a behavior that wakes up on mouse events

   - On a mouse release:
     - Construct a `PickRay` from the eye passing through the 2D mouse screen point
     - Initiate a pick to find all pick hits along the ray, sorted from closest to furthest
     - Get the first pick hit in the returned data
     - Do something to that picked shape
     - (Re)declare interest in mouse events

7. Picking example



8. Picking example code

- Create a pick ray aimed using mouse screen data

```
PickRay myRay = new PickRay( rayOrigin, rayDirection );
```

- Initiate a pick starting at a `Locale`

```
SceneGraphPath[ ] results = myLocale.pickAllSorted(myRay);
```

- Get the first (closest) shape off the results

```
Node pickedObject = results[0].getObject( );
```