3D Features

3D Features Planned for Version 8

Introduction

This page contains the list of 3D features schedule for JavaFX 8.0

We have been thinking about the possible 3D features for JavaFX 8 for a while. We are now ready to present the plan to the community for review. This information has also been presented at this year's JavaOne ***3D Made Easy with JavaFX*** technical session.

Three umbrella JIRA issues were created to capture most of the work: Movable Camera , 3D Geometry, and 3D Attributes

Here this a link to FX Experience blog on JavaFX 3D Early Access

A list simple examples

- A blue sphere and a red box with mouse control: SphereAndBox.java
- A rectangular mesh: SimpleMeshView.java

Proposed API

Proposed Features

Movable cameras and SubScene

- Camera is now a Node
- Camera can be added to a scene graph
- Camera's position and aim (or orientation) is set using standard Node transform properties and methods
- For backward compatibility, fixed camera need not be added to the scene
- Added new properties for near & far clipping plane
- SubScene is a special Node for scene separation
- It can be used to render part of the Scene with different camera
- Possible use cases are:
    - Overlay for UI controls (needs a static camera)
    - Underlay for background (static or updated less frequently)
    - "Heads-up" display

Camera Class Hierarchy

- javafx.scene.Node
    - javafx.scene.Camera (abstract)
        - javafx.scene.ParallelCamera
        - javafx.scene.PerspectiveCamera
    - javafx.scene.SubScene

Code segment

Specifying a fixed Camera (existing 2.2 API)

```
// Create a camera and add it to the Scene
Camera camera = new PerspectiveCamera();
scene.setCamera(camera);
```

Specifying a movable Camera

```
// Create a camera and add it to the Scene
Camera camera = new PerspectiveCamera();
scene.setCamera(camera);

// Add camera to scene graph (so it can move)
Group cameraGroup = new Group();
cameraGroup.getChildren().add(camera);
root.getChildren().add(cameraGroup);

// Rotate the camera
camera.rotate(45);

// Move the cameraGroup (camera moves with it)
cameraGroup.setTranslateZ(-75);
```

3D primitives

- Added two type of 3D shapes, extending from an abstract Shape3D base class:
    - User-defined shapes (MeshView)
    - Predefined shapes

User-defined shapes

- User defined mesh (geometry) of a shape by specifying a set of points, texture coordinates, and faces (triangles that describe the topology)
- User defined smoothing group to specify group of faces that are part of the same curved surface
- A mesh is sharable among mutiple user-defined shapes

Predefined shapes

- Three commonly used predefined 3D shapes are introduced: Box, Cylinder and Sphere

Shape3D Class Hierarchy

- javafx.scene.Node
    - javafx.scene.shape.Shape3D (abstract)
        - javafx.scene.shape.MeshView
        - javafx.scene.shape.Box
        - javafx.scene.shape.Cylinder
        - javafx.scene.shape.Sphere

Mesh Class Hierarchy

- java.lang.Object
    - javafx.scene.shape.Mesh (abstract)
        - javafx.scene.shape.TriangleMesh

Code Segment

Defining a MeshView

```
// Create the arrays of positions, texCoords
float[] positions = createPositions();
float[] texCoords = createUVs();

// Create faces (indices into the positions, texCoord arrays)
int[] faces = createFaces();

// Create a mesh
TriangleMesh mesh = new TriangleMesh();
mesh.setPositions(positions);
mesh.setTexCoords(texCoords);
mesh.setFaces(faces);

// Create meshView
MeshView mv = new MeshView(mesh);
```

Using Predefined Shapes

```
// Create a sphere with the given radius
Sphere sphere = new Sphere(10.0);

// Create a sphere with the given radius, number of divisions
Sphere sphere = new Sphere(10.0, 40);

// Create a cylinder with the given radius, and height
Cylinder cylinder = new Cylinder(10.0, 30.0);

// Create a box with the given width, height, and depth
Box box = new Box(1.0, 1.0, 1.0);

// NOTE: Predefined 3D shapes are centered at (0,0,0)
```

3D attributes

- Added lights and 3D materials to add realism for 3D shapes.
- Material specifies the appearance of a 3D shape
- Light interacts with the geometry of a Shape3D and its material to provide the rendering result
- A Shape3D can be rendered as a filled shape or as a wireframe
- A Shape3D has a face culling property (front, back or none)

Light

- Light is defined as a node in the scene graph
- A scene contains a set of active lights
    - A default light is provided when the set of active light is empty
- Each light contains a set of affected nodes
    - If a Parent is in the set, all its children are affected
    - Default is the root node of the Scene
- Added 2 type of light sources:
    - AmbientLight and PointLight
    - May add more in the future

Material

- Material contains a set of rendering properties
- PhongMaterial is a concrete subclass of Material
    - It has the following properties:
        - Ambient color
        - Diffuse color, diffuse map
        - Specular color, specular map
        - Specular power
        - Bump map
        - Self-illumination map
- Sharable among multiple Shape3D nodes

Light Class Hierarchy

- javafx.scene.Node
    - javafx.scene.LightBase (abstract)
        - javafx.scene.AmbientLight
        - javafx.scene.PointLight

Material Class Hierarchy

- java.lang.Object
    - javafx.scene.paint.Material (abstract)
        - javafx.scene.paint.PhongMaterial

Code Segment

Defining Lights

```
// Create point light and add it to the Scene
PointLight light = new PointLight();
light.setColor(Color.RED);
scene.getLights().add(light);

// Add light to scene graph (so it can move)
Group lightGroup = new Group();
lightGroup.getChildren().add(light);
root.getChildren().add(lightGroup);

// Rotate the light
light.rotate(45);

// Move the lightGroup (light moves with it)
lightGroup.setTranslateZ(-75);
```

Defining Materials

```
// Create material
Material mat = new PhongMaterial();
Image diffuseMap = new Image("diffuseMap.png");
Image bumpMap = new Image("normalMap.png");

// Set material properties
mat.setDiffuseMap(diffuseMap);
mat.setBumpMap(normalMap);
mat.setSpecularColor(Color.WHITE);

// Use the material for a shape
shape3d.setMaterial(mat);
```

3D picking

- 3D ray picking already used for 2D primitives with PerspectiveCamera
- Existing limitation when used with depth buffer will be fixed (JIRA: RT-13740)
- We will add support for picking 3D geometry (JIRA: RT-24646)

Proposed API to support 3D picking: Picking3dAPI

Methods added to Node

- A LOD helper method that returns the area of the Node projected onto the physical screen in pixel units.

```
public double computeAreaInScreen()
```

- A set of 3D transform methods

```
public Point3D sceneToLocal(Point3D scenePoint)
public Point3D sceneToLocal(double sceneX, double sceneY, double sceneZ)
public Point3D localToScene(Point3D localPoint)
public Point3D localToScene(double x, double y, double z)
public Point3D parentToLocal(Point3D parentPoint)
public Point3D parentToLocal(double parentX, double parentY, double parentZ)
public Point3D localToParent(Point3D localPoint)
public Point3D localToParent(double x, double y, double z)
```

Loader support

- Many 3D file formats exist, such as:
    - Obj, Maya, 3D Studio Max, Collada, KRML
- We will not provide a loader as part of the JavaFX runtime
- We will make sample code available for one or two popular formats