RAGE
Orbit Controller w/ Spherical Coordinates

```
public class MyGame extends VariableFrameRateGame
{ . . .
  private Camera3Pcontroller orbitController;
  private Action moveFwdAct;  // other action classes as needed
  . . .
  protected void setupScene . . .
  { im = new GenericInputManager();
    . . .
    // make manual object - dolphin avatar
    Entity dolphinE = sm.createEntity("dolphin", "dolphinHighPoly.obj");
    dolphinE.setPrimitive(Primitive.TRIANGLES);
    SceneNode dolphinN =
       sm.getRootSceneNode().createChildSceneNode("dolphinNode");
    dolphinN.attachObject(dolphinE);

    setupOrbitCamera(eng, sm);
    setupInputs(sm);
    dolphinN.yaw(Degreef.createFrom(45.0f));
  }

  protected void setupOrbitCamera(Engine eng, SceneManager sm)
  { SceneNode dolphinN = sm.getSceneNode("dolphinNode");
    SceneNode cameraN = sm.getSceneNode("MainCameraNode");
    Camera camera = sm.getCamera("MainCamera");
    String gpName = im.getFirstGamepadName();
    orbitController =
      new Camera3Pcontroller(camera, cameraN, dolphinN, gpName, im);
  }

  protected void setupInputs(SceneManager sm)
  { String kbName = im.getKeyboardName();
    String gpName = im.getFirstGamepadName();
    SceneNode dolphinN =
        getEngine().getSceneManager().getSceneNode("dolphinNode");

    // avatar movement – move forward
    moveFwdAct = new MoveForwardAction(dolphinN);
    im.associateAction(gpName,
        net.java.games.input.Component.Identifier.Button._3,
        moveFwdAct, InputManager.INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);

    // other Action classes for avatar instantiated as needed
  }

  protected void update(Engine engine)
  { // as before, plus the following:
    orbitController.updateCameraPosition();
} }
```

```
public class MoveForwardAction extends AbstractInputAction
{
  private Node avN;

  public MoveForwardAction(Node n)
  { avN = n;
  }

  public void performAction(float time, Event e)
  { avN.moveForward(0.01f);
  }
}
```

Simple Orbit Controller Class:

```
public class Camera3Pcontroller
{ private Camera camera;          //the camera being controlled
  private SceneNode cameraN;     //the node the camera is attached to
  private SceneNode target;      //the target the camera looks at
  private float cameraAzimuth;   //rotation of camera around Y axis
  private float cameraElevation; //elevation of camera above target
  private float radias;          //distance between camera and target
  private Vector3 targetPos;     //target's position in the world
  private Vector3 worldUpVec;

  public Camera3Pcontroller(Camera cam, SceneNode camN,
                SceneNode targ, String controllerName, InputManager im)
  { camera = cam;
    cameraN = camN;
    target = targ;
    cameraAzimuth = 225.0f;   // start from BEHIND and ABOVE the target
    cameraElevation = 20.0f;  // elevation is in degrees
    radias = 2.0f;
    worldUpVec = Vector3f.createFrom(0.0f, 1.0f, 0.0f);
    setupInput(im, controllerName);
    updateCameraPosition();
  }
  ------------------------------------------------------------------
  // Updates camera position: computes azimuth, elevation, and distance
  // relative to the target in spherical coordinates, then converts those
  // to world Cartesian coordinates and setting the camera position

  public void updateCameraPosition()
  { double theta = Math.toRadians(cameraAzimuth);    // rot around target
    double phi = Math.toRadians(cameraElevation);    // altitude angle
    double x = radias * Math.cos(phi) * Math.sin(theta);
    double y = radias * Math.sin(phi);
    double z = radias * Math.cos(phi) * Math.cos(theta);
    cameraN.setLocalPosition(Vector3f.createFrom
            ((float)x, (float)y, (float)z).add(target.getWorldPosition()));
    cameraN.lookAt(target, worldUpVec);
  }
  ------------------------------------------------------------------
  private void setupInput(InputManager im, String cn)
  { Action orbitAAction = new OrbitAroundAction();
    im.associateAction(cn,
        net.java.games.input.Component.Identifier.Axis.RX, orbitAAction,
        InputManager.INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);

    // similar input set up for OrbitRadiasAction, OrbitElevationAction
  }
  ------------------------------------------------------------------
  private class OrbitAroundAction extends AbstractInputAction
  { // Moves the camera around the target (changes camera azimuth).
    public void performAction(float time, net.java.games.input.Event evt)
    { float rotAmount;
      if (evt.getValue() < -0.2)
      { rotAmount=-0.2f; }
        else
        { if (evt.getValue() > 0.2)
        { rotAmount=0.2f; }
        else
        { rotAmount=0.0f; }
      }
      cameraAzimuth += rotAmount;
      cameraAzimuth = cameraAzimuth % 360;
      updateCameraPosition();
    } }
    // similar for OrbitRadiasAction, OrbitElevationAction
  }
```