TAGE
Orbit Camera Controller (based on ex.02b)

```
. . .
import java.lang.Math;
import java.util.ArrayList;

public class MyGame extends VariableFrameRateGame
{ . . .
  private CameraOrbitController orbitController;
  . . .
```

```
  loadShapes(), loadTextures(), buildObjects()  are unchanged
  . . .
  @Override
  public void initializeGame()
  { . . .
    // ----------------- initialize camera ----------------
    im = engine.getInputManager();
    String gpName = im.getFirstGamepadName();

    Camera c = (engine.getRenderSystem())
                                .getViewport("MAIN").getCamera();
    orbitController = new CameraOrbitController(
                                c, avatar, gpName, engine);

    // ----------------- OTHER INPUTS SECTION -----------------------------
    FwdAction fwdAction = new FwdAction(this);
    TurnAction turnAction = new TurnAction(this);

    // attach the action objects for moving the avatar
    im.associateActionWithAllKeyboards(
          net.java.games.input.Component.Identifier.Key.W,
          fwdAction,
          InputManager.INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);

    im.associateActionWithAllKeyboards(
          net.java.games.input.Component.Identifier.Key.D,
          turnAction,
          InputManager.INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);
  }
```

```
  public void update()
  { . . .
    orbitController.updateCameraPosition();
  }

// private void positionCameraBehindAvatar()
          *** (REMOVED) ***
```

FwdAction.java *and* **TurnAction.java** — unchanged

OrbitRadiusAction *and* **OrbitElevationAction**
would also need to be written

CameraOrbitController.java

```
. . .
import java.lang.Math;

public class CameraOrbitController
{  private Engine engine;
   private Camera camera;          // the camera being controlled
   private GameObject avatar;      // the target avatar the camera looks at
   private float cameraAzimuth;    // rotation around target Y axis
   private float cameraElevation;  // elevation of camera above target
   private float cameraRadius;     // distance between camera and target

   public CameraOrbitController(Camera cam, GameObject av,
                                String gpName, Engine e)
   {  engine = e;
      camera = cam;
      avatar = av;
      cameraAzimuth = 0.0f;       // start BEHIND and ABOVE the target
      cameraElevation = 20.0f;    // elevation is in degrees
      cameraRadius = 2.0f;        // distance from camera to avatar
      setupInputs(gpName);
      updateCameraPosition();
   }

   private void setupInputs(String gp)
   {  OrbitAzimuthAction azmAction = new OrbitAzimuthAction();
      InputManager im = engine.getInputManager();
      im.associateAction(gp,
         net.java.games.input.Component.Identifier.Axis.RX, azmAction,
         InputManager.INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);
   }

   // Compute the camera's azimuth, elevation, and distance, relative to
   // the target in spherical coordinates, then convert to world Cartesian
   // coordinates and set the camera position from that.

   public void updateCameraPosition()
   {  Vector3f avatarRot = avatar.getWorldForwardVector();
      double avatarAngle = Math.toDegrees((double)
         avatarRot.angleSigned(new Vector3f(0,0,-1), new Vector3f(0,1,0)));
      float totalAz = cameraAzimuth –
        (float)avatarAngle;double theta = Math.toRadians(cameraAzimuth);
      double phi = Math.toRadians(cameraElevation);
      float x = cameraRadius * (float)(Math.cos(phi) * Math.sin(theta));
      float y = cameraRadius * (float)(Math.sin(phi));
      float z = cameraRadius * (float)(Math.cos(phi) * Math.cos(theta));
      camera.setLocation(new
                  Vector3f(x,y,z).add(avatar.getWorldLocation()));
      camera.lookAt(avatar);
   }

   private class OrbitAzimuthAction extends AbstractInputAction
   {  public void performAction(float time, Event event)
      {  float rotAmount;
         if (event.getValue() < -0.2)
         {  rotAmount=-0.2f;  }
         else
         {  if (event.getValue() > 0.2)
            {  rotAmount=0.2f;  }
            else
            {  rotAmount=0.0f;  }
         }
         cameraAzimuth += rotAmount;
         cameraAzimuth = cameraAzimuth % 360;
         updateCameraPosition();
   }  }
```