TAGE
Manual Objects & Camera Movement

```
. . .
public class MyGame extends VariableFrameRateGame
{  . . .
   private GameObject pyr, sph;
   private ObjShape pyrS, sphS;
   private TextureImage earth, brick;
   . . .
```

```
   @Override
   public void loadShapes()
   {  pyrS = new ManualPyramid();
      sphS = new Sphere();
   }
   . . .

   @Override
   public void buildObjects()
   {  . . .
      // build pyramid at center
      pyr = new GameObject(GameObject.root(), pyrS, brick);
      initialTranslation = (new Matrix4f()).translation(0,0,0);
      pyr.setLocalTranslation(initialTranslation);
      pyr.getRenderStates().hasLighting(true);
      . . .
   }

   @Override
   public void initializeGame()
   {  . . .
      // ------------ inputs section ------------------
      im = engine.getInputManager();

      FwdAction fwdAction = new FwdAction(this);
      YawAction yawAction = new YawAction(this);

      im.associateActionWithAllGamepads(
         net.java.games.input.Component.Identifier.Button._1,
         fwdAction, InputManager
                        .INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);
      im.associateActionWithAllGamepads(
         net.java.games.input.Component.Identifier.Axis.X,
         yawAction, InputManager
                        .INPUT_ACTION_TYPE.REPEAT_WHILE_DOWN);
   }
```

FwdAction.java

```
   @Override
   public void performAction(float time, Event e)
   {  c = (game.getEngine().getRenderSystem())
                                 .getViewport("MAIN").getCamera();
      oldPosition = c.getLocation();
      fwdDirection = c.getN();
      fwdDirection.mul(0.01f);
      newPosition = oldPosition.add(fwdDirection.x(),
                                 fwdDirection.y(), fwdDirection.z());
      c.setLocation(newPosition);
   }
```

TurnAction.java

```
   . . .
   @Override
   public void performAction(float time, Event e)
   {  float keyValue = e.getValue();
      if (keyValue > -.2 && keyValue < .2) return;  // deadzone
      c = (game.getEngine().getRenderSystem())
                                    .getViewport("MAIN").getCamera();
      rightVector = c.getU();
      upVector = c.getV();
      fwdVector = c.getN();
      rightVector.rotateAxis(0.01f, upVector.x(), upVector.y(), upVector.z());
      fwdVector.rotateAxis(0.01f, upVector.x(), upVector.y(), upVector.z());
      c.setU(rightVector);
      c.setN(fwdVector);
   }
```

ManualPyramid.java

```
package myGame;

import tage.*;
import tage.shapes.*;

public class ManualPyramid extends ManualObject
{
   private float[] vertices = new float[]
   { -1.0f, -1.0f, 1.0f,    1.0f, -1.0f, 1.0f,    0.0f, 1.0f, 0.0f,      //front
     1.0f, -1.0f, 1.0f,     1.0f, -1.0f, -1.0f,   0.0f, 1.0f, 0.0f,      //right
     1.0f, -1.0f, -1.0f,   -1.0f, -1.0f, -1.0f,   0.0f, 1.0f, 0.0f,      //back
    -1.0f, -1.0f, -1.0f,   -1.0f, -1.0f, 1.0f,    0.0f, 1.0f, 0.0f,      //left
    -1.0f, -1.0f, -1.0f,    1.0f, -1.0f, 1.0f,   -1.0f, -1.0f, 1.0f,     //LF
     1.0f, -1.0f, 1.0f,    -1.0f, -1.0f, -1.0f,   1.0f, -1.0f, -1.0f };  //RR

   private float[] texcoords = new float[]
   {  0.0f, 0.0f,   1.0f, 0.0f,   0.5f, 1.0f,
      0.0f, 0.0f,   1.0f, 0.0f,   0.5f, 1.0f,
      0.0f, 0.0f,   1.0f, 0.0f,   0.5f, 1.0f,
      0.0f, 0.0f,   1.0f, 0.0f,   0.5f, 1.0f,
      0.0f, 0.0f,   1.0f, 1.0f,   0.0f, 1.0f,
      1.0f, 1.0f,   0.0f, 0.0f,   1.0f, 0.0f };

   private float[] normals = new float[]
   {  0.0f, 1.0f, 1.0f,     0.0f, 1.0f, 1.0f,     0.0f, 1.0f, 1.0f,
      1.0f, 1.0f, 0.0f,     1.0f, 1.0f, 0.0f,     1.0f, 1.0f, 0.0f,
      0.0f, 1.0f, -1.0f,    0.0f, 1.0f, -1.0f,    0.0f, 1.0f, -1.0f,
     -1.0f, 1.0f, 0.0f,    -1.0f, 1.0f, 0.0f,    -1.0f, 1.0f, 0.0f,
      0.0f, -1.0f, 0.0f,    0.0f, -1.0f, 0.0f,    0.0f, -1.0f, 0.0f,
      0.0f, -1.0f, 0.0f,    0.0f, -1.0f, 0.0f,    0.0f, -1.0f, 0.0f };

   public ManualPyramid()
   {  super();

      setNumVertices(18);
      setVertices(vertices);
      setTexCoords(texcoords);
      setNormals(normals);

      setMatAmb(Utils.goldAmbient());
      setMatDif(Utils.goldDiffuse());
      setMatSpe(Utils.goldSpecular());
      setMatShi(Utils.goldShininess());
   }
}
```