RAGE
Input Devices and Actions

two Action classes:

```
// An AbstractInputAction that quits the game.
// It assumes availability of a method "shutdown" in the game
// (this is always true for classes that extend BaseGame).

import ray.input.action.AbstractInputAction;
import ray.rage.game.*;
import net.java.games.input.Event;

public class QuitGameAction extends AbstractInputAction
{
   private MyGame game;

   public QuitGameAction(MyGame g)
   { game = g;
   }

   public void performAction(float time, Event event)
   { System.out.println("shutdown requested");
     game.exit();
   }
}
```

```
// An AbstractInputAction that increments a counter in the game.
// It assumes availability of a method "incrementCounter" in the game.

import ray.input.action.AbstractInputAction;
import ray.rage.game.*;
import net.java.games.input.Event;

public class IncrementCounterAction extends AbstractInputAction
{
   private MyGame game;

   public IncrementCounterAction(MyGame g)
   { game = g;
   }

   public void performAction(float time, Event e)
   { System.out.println("counter action initiated");
     game.incrementCounter();
   }
}
```

a RAGE game that uses a GamePad:

```
... imports as before, plus these:
import ray.rage.rendersystem.states.*;
import ray.rage.asset.texture.*;
import ray.input.*;
import ray.input.action.*;

public class MyGame extends VariableFrameRateGame
{  ... same variable declarations as before, plus these:

   private InputManager im;
   private Action quitGameAction, incrementCounterAction;

   ... constructor and main() same as before
   ... setupWindow(), setupCameras(), incrementCounter() same as before

   protected void setupScene(Engine eng, SceneManager sm)
                                                    throws IOException
   {  setupInputs(); // new function (defined below) to set up input actions

      ... the rest is the same
      ... except that this time we show how to attach a texture manually:

      TextureManager tm = eng.getTextureManager();
      Texture redTexture = tm.getAssetByPath("redDolphin.jpg");
      RenderSystem rs = sm.getRenderSystem();
      TextureState state = (TextureState)
                  rs.createRenderState(RenderState.Type.TEXTURE);
      state.setTexture(redTexture);
      dolphinE.setRenderState(state);
   }

   protected void setupInputs()
   {  im = new GenericInputManager();
      String kbName = im.getKeyboardName();
      String gpName = im.getFirstGamepadName();

      // build some action objects for doing things in response to user input
      quitGameAction = new QuitGameAction(this);
      incrementCounterAction = new IncrementCounterAction(this);

      // attach the action objects to keyboard and gamepad components
      im.associateAction(kbName,
         net.java.games.input.Component.Identifier.Key.ESCAPE,
         quitGameAction,
         InputManager.INPUT_ACTION_TYPE.ON_PRESS_ONLY);

      im.associateAction(gpName,
         net.java.games.input.Component.Identifier.Button._9,
         quitGameAction,
         InputManager.INPUT_ACTION_TYPE.ON_PRESS_ONLY);

      im.associateAction(gpName,
         net.java.games.input.Component.Identifier.Button._3,
         incrementCounterAction,
         InputManager.INPUT_ACTION_TYPE.ON_PRESS_ONLY);

      im.associateAction(kbName,
         net.java.games.input.Component.Identifier.Key.C,
         incrementCounterAction,
         InputManager.INPUT_ACTION_TYPE.ON_PRESS_ONLY);
   }

   protected void update(Engine engine)
   {  ...same as before, plus the following:
      // tell the input manager to process the inputs
      im.update(elapsTime);
   }
}
```