Intro to Input Handling

Keyboard Input – Step 1

(get something working)

Handling Keyboard Input – Step 1

1. Obtain the state of the keyboard: `KeyboardState state = Keyboard.GetState();`
2. Check to see if a key is down: `Keyboard.GetState().IsKeyDown(Keys.Escape)`
3. Call some code based upon which key was pressed
4. Profit!

Handling Keyboard Input – Step 1(b)

1. Obtain list of all keys pressed: `Keys[] keys = Keyboard.GetState().GetPressedKeys();`
2. Loop over those keys and do something based on which key is pressed
3. Profit!

Handle Single Key of Interest

```
if (Keyboard.GetState().IsKeyDown(Keys.Escape))
{
    this.Exit();
}
```

Handle All Keys Currently Pressed

```
foreach (Keys key inKeyboard.GetState().GetPressedKeys())
{
    moveOnKey(key, moveDistance, rotateDistance);
}

private void moveOnKey(Keys key, int moveDistance, float rotateDistance)
{
    switch (key)
    {
        case Keys.W:
        case Keys.Up:
            moveUp(moveDistance);
            break;
        case Keys.A:
        case Keys.Left:
            moveLeft(moveDistance);
            break;
    ...
    }
}
```

Is this good enough?

Is this good enough?

(not yet, but getting there)

What are the Issues?

- Update (movement) based on the frame rate
- Weak design

Keyboard Input – Step 2

(update based on elapsed time)

What are the Issues?

- Update (movement) based on the frame rate
- Weak design

Utilize Provided Elapsed Time

```
protected override void Update(GameTime gameTime)
{
    int moveDistance = (int)(gameTime.ElapsedGameTime.TotalMilliseconds *
        SPRITE_MOVE_PIXELS_PER_MS);
    float rotateDistance = (float)(gameTime.ElapsedGameTime.TotalMilliseconds *
        SPRITE_ROTATE_RADIANS_PER_MS);

    foreach (Keys key in Keyboard.GetState().GetPressedKeys())
    {
        moveOnKey(key, moveDistance, rotateDistance);
    }

    ...

}
```

Is this good enough?

Is this good enough?

(not yet, but getting there)

Keyboard Input – Step 3

(semantic separation)

What are the Issues?

- ~~Update (movement) based on the frame rate~~
- Weak design

Semantic Separation

1. Register to receive input events
2. Process registered input handlers

Define an InputDevice Interface & Some Delegates

```
public interface IInputDevice
{
    void Update(GameTime gameTime);
}

public class InputDeviceHelper
{
    public delegate void CommandDelegate(GameTime gameTime, float value);
    public delegate void CommandDelegatePosition(GameTime GameTime, int x, int y);
}
```

```
public class KeyboardInput : IInputDevice
{

    ... good stuff coming soon to a theater near you ...

}
```

Input Manager – Track Registered Commands

```
private struct CommandEntry
{
    public CommandEntry(Keys key, bool keyPressOnly, InputDeviceHelper.CommandDelegate callback)
    {
        this.key = key;
        this.keyPressOnly = keyPressOnly;
        this.callback = callback;
    }

    public Keys key;
    public bool keyPressOnly;
    public InputDeviceHelper.CommandDelegate callback;
}


private Dictionary<Keys, CommandEntry> m_commandEntries = new Dictionary<Keys, CommandEntry>();
```

```
public void registerCommand(Keys key, bool keyPressOnly, InputDeviceHelper.CommandDelegate callback)
{
    if (m_commandEntries.ContainsKey(key))
    {  // Removing any existing entry, because it is being replaced
        m_commandEntries.Remove(key);
    }
    m_commandEntries.Add(key, new CommandEntry(key, keyPressOnly, callback));
}
```

Client – Register for Events

```
m_inputKeyboard = new KeyboardInput();

m_inputKeyboard.registerCommand(Keys.W, false, new InputDeviceHelper.CommandDelegate(onMoveUp));
m_inputKeyboard.registerCommand(Keys.S, false, new InputDeviceHelper.CommandDelegate(onMoveDown));
m_inputKeyboard.registerCommand(Keys.A, false, new InputDeviceHelper.CommandDelegate(onMoveLeft));
m_inputKeyboard.registerCommand(Keys.D, false, new InputDeviceHelper.CommandDelegate(onMoveRight));
m_inputKeyboard.registerCommand(Keys.Q, false, new InputDeviceHelper.CommandDelegate(onRotateLeft));
m_inputKeyboard.registerCommand(Keys.E, false, new InputDeviceHelper.CommandDelegate(onRotateRight));

m_inputKeyboard.registerCommand(Keys.Up, false, new InputDeviceHelper.CommandDelegate(onMoveUp));
m_inputKeyboard.registerCommand(Keys.Down, false, new InputDeviceHelper.CommandDelegate(onMoveDown));
m_inputKeyboard.registerCommand(Keys.Left, false, new InputDeviceHelper.CommandDelegate(onMoveLeft));
m_inputKeyboard.registerCommand(Keys.Right, false, new InputDeviceHelper.CommandDelegate(onMoveRight));
```

Game Loop – Process Input

```
protected override void Update(GameTime gameTime)
{
    ...
    m_inputKeyboard.Update(gameTime);

    base.Update(gameTime);
}
```

Input Manager – Process Registered Handlers

```
public void Update(GameTime gameTime)
{
    KeyboardState state = Keyboard.GetState();
    foreach (CommandEntry entry in this.m_commandEntries.Values)
    {
        if (entry.keyPressOnly && keyPressed(entry.key))
        {
            entry.callback(gameTime, 1.0f);
        }
        else if (!entry.keyPressOnly && state.IsKeyDown(entry.key))
        {
            entry.callback(gameTime, 1.0f);
        }
    }
    m_statePrevious = state;
}

private bool keyPressed(Keys key)
{
    return (Keyboard.GetState().IsKeyDown(key) && !m_statePrevious.IsKeyDown(key));
}

private KeyboardState m_statePrevious;
```

Is this good enough?

Is this good enough?

(reasonable, but still room for improvement)

What are the Issues?

- ~~Update (movement) based on the frame rate~~
- ~~Weak design~~
- Ability to unregister from an event
- Multiple subscribers per event
- Input patterns
  - One time only
  - Repeat based on elapsed time (a firing rate)
    - currently fires every frame...remember, that's bad!
- Mouse and other input devices