CS 5410

Intro to Mouse Input Handling

Handling Mouse Input

1. Define a Input Manager
   - Have to write our own code to create these events
     - mousedown
     - mouseup
     - mousemove
   - Invoke handlers on update
2. Register handler for mouse events
   - Concept of ‘capture’

Input Manager – Define Mouse Events

- Define an enum for the type of mouse events we will create
- Track the up/down state of the mouse so we can decide when mouse up/down has occurred

```
public enum MouseEvent
{
      MouseDown,
      MouseUp,
      MouseMove
}

private bool m_mouseDown = false;
private MouseState m_mousePreviousState = Mouse.GetState();
```

Input Manager – Track Registered Commands

```
private struct CommandEntry
{
      public CommandEntry(MouseEvent evt, InputDeviceHelper.CommandDelegatePosition callback)
      {
            this.evt = evt;
            this.callback = callback;
      }

      public MouseEvent evt;
      public InputDeviceHelper.CommandDelegatePosition callback;
}

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

Input Manager - Register

```
public void registerCommand(MouseEvent evt, InputDeviceHelper.CommandDelegatePosition callback)
{
      if (m_commandEntries.ContainsKey(evt))
      {
            m_commandEntries.Remove(evt);
      }
      m_commandEntries.Add(evt, new CommandEntry(evt, callback));
}
```

Client Code - Register

```
m_inputMouse.registerCommand(
      MouseInput.MouseEvent.MouseDown,
      new InputDeviceHelper.CommandDelegatePosition(onMouseDown));

m_inputMouse.registerCommand(
      MouseInput.MouseEvent.MouseUp,
      new InputDeviceHelper.CommandDelegatePosition(onMouseUp));

m_inputMouse.registerCommand(
      MouseInput.MouseEvent.MouseMove,
      new InputDeviceHelper.CommandDelegatePosition(onMouseMove));
```

Input Manager – Process Registered Handlers

```
public void Update(GameTime gameTime)
{
	MouseState state = Mouse.GetState();
	foreach (CommandEntry entry in this.m_commandEntries.Values)
	{
		// Transitioning from mouse up to mouse down
		if (entry.evt == MouseEvent.MouseDown && state.LeftButton == ButtonState.Pressed && !m_mouseDown)
		{
			entry.callback(gameTime, state.X, state.Y);
		}
		// Transitioning from mouse down to mouse up
		if (entry.evt == MouseEvent.MouseUp && state.LeftButton == ButtonState.Released && m_mouseDown)
		{
			entry.callback(gameTime, state.X, state.Y);
		}
		if (entry.evt == MouseEvent.MouseMove)
		{
			if (state.X != m_mousePreviousState.X || state.Y != m_mousePreviousState.Y)
			{
				entry.callback(gameTime, state.X, state.Y);
			}
		}
	}

	m_mouseDown = (state.LeftButton == ButtonState.Pressed);
	m_mousePreviousState = state;
}
```