Session 3.2.2

Adding Vibration Effects
Session Overview

- Describe how the vibration feature of the gamepad works
- Show how an XNA program can control gamepad vibration
- Add vibration to the Color Nerve game which we created based on the MoodLight program
- Look at how we can change the behavior of a game by modifications to the code
The Xbox Gamepad Vibration

- The gamepad contains two motors which turn weighted flywheels
- The wheels turn at different speeds to vibrate the pad
- Games can control the power going into the motors to get different levels of vibration
Vibration Speeds

- The motor on the left provides low-frequency rumble
- The motor on the right provides higher-frequency vibration
- XNA allows you to control the power going to either of these motors from a C# program
Controlling Gamepad Vibration from XNA

GamePad.SetVibration ( PlayerIndex.One, 1, 0 ) ;

- We have used the `GamePad` class before
- It provides the `GetState` method that we used to read the gamepad state
- It also provides a method called `SetVibration` which is used to control the vibration of a gamepad
- The method is given three parameters
- A parameter is a way of feeding information into a method
Selecting the Gamepad to Be Controlled

GamePad.SetVibration ( PlayerIndex.One, 1, 0 ) ;

- This parameter value identifies the gamepad to be controlled
- We used it to tell GetState which gamepad to read
- You can try to control gamepads that aren’t there, but the method call won’t do anything
Setting the Vibration Levels

- This parameter value gives the amount of vibration the left (low frequency) motor should produce.
- It is given as a floating point value between 0 and 1.
- If we give the value 1 it will produce maximum vibration.
- If we give the value 0 the vibration is turned off.

GamePad.SetVibration ( PlayerIndex.One, 1, 0 );
Setting the Vibration Levels

- This parameter value gives the amount of vibration the right (high frequency) motor should produce.
- It is given as a floating point value between 0 and 1.
- It works in the same way as the left-hand value, but the vibration effect is different.
- Once you have set a vibration level the pad will continue to vibrate until you tell it something different.

```csharp
GamePad.SetVibration ( PlayerIndex.One, 1, 0 );
```
protected override void Update(GameTime gameTime) 
{
    // Allows the game to exit
        this.Exit();

    GamePadState pad1 = GamePad.GetState(PlayerIndex.One);

    if (pad1.Buttons.X == ButtonState.Pressed)
        GamePad.SetVibration(PlayerIndex.One, 1, 0);

    base.Update(gameTime);
}
1. Vibration Demonstration

- This program implements the vibration behavior that we have just seen.
- However, it does have a problem...
Making the Vibration Stop

```csharp
if (pad1.Buttons.X == ButtonState.Pressed)
   GamePad.SetVibration(PlayerIndex.One, 1, 0);
else
    GamePad.SetVibration(PlayerIndex.One, 0, 0);
```

- The game program must tell the gamepad to stop vibrating when the button is not pressed.
- The easiest way to achieve this is to add an `else` part to the conditional statement.
- This version of the code will only make the gamepad vibrate when the Blue button is pressed.
Adding Vibration to Color Nerve

```csharp
if (redIntensity > 220)
    GamePad.SetVibration(PlayerIndex.One, 1, 0);
else
    GamePad.SetVibration(PlayerIndex.One, 0, 0);
```

- The Color Nerve program could make the gamepad vibrate when the intensity values get close to wrapping round.
- The condition must compare two values and trigger when one is greater than the other.
- We can use the “greater than” operator to do this.
The Greater Than Operator

- The Greater Than operator can be placed between two numeric values
- It returns true if the number on the left is greater than the number on the right
- In all other situations (including equals) it returns false

```csharp
if (redIntensity > 220)
    GamePad.SetVibration(PlayerIndex.One, 1, 0);
else
    GamePad.SetVibration(PlayerIndex.One, 0, 0);
```
Combining Tests

```csharp
if (redIntensity > 220 ||
    greenIntensity > 220 ||
    blueIntensity > 220)
    GamePad.SetVibration(PlayerIndex.One, 1, 0);
else
    GamePad.SetVibration(PlayerIndex.One, 0, 0);
```

- If we want the vibration to start when any of the color intensities exceeds 220 we have to test all of them
- We can combine the results using logical OR, so that the vibration starts if any of them exceeds the limit
Using Blocks to Improve Code Appearance

```csharp
if (redIntensity > 220 ||
    greenIntensity > 220 ||
    blueIntensity > 220)
{
    GamePad.SetVibration(PlayerIndex.One, 1, 0);
}
else
{
    GamePad.SetVibration(PlayerIndex.One, 0, 0);
}
```

- Creating blocks can make your code clearer
- In this case it separates the code from the conditions
2. Color Nerve with Vibration

- This version of the game provides vibration feedback when any of the color intensity values exceeds 220
Summary

- The Xbox gamepad has two vibration motors for different frequency vibration of the gamepad.
- The GamePad class which is part of XNA provides a method called SetVibration which controls the intensity of the vibration the motors produce.
- Once a gamepad has been told to vibrate it will vibrate at that level until given another vibration command or the Exit method is called to end an XNA game.