Session 3.2.1

Using the Keyboard
Session Overview

- Introduce the keyboard device
- Show how keys on a keyboard can be represented by enumerated types
- Show how an XNA program can use both the keyboard and the gamepad to control game programs
XNA and the Keyboard

- The Xbox 360 is not supplied with a keyboard
- You can connect a keyboard to any of the USB ports on the Xbox itself
- A Windows PC already has a keyboard available
- The keyboard is used in exactly the same way in an XNA game program for Xbox or Windows PC
- It is not possible to use the keyboard on a Zune device
Keyboards in XNA Game Programs

- The keyboard is used in a very similar way to the gamepad.
- A game can tell whether or not a particular key is pressed or released.
- Keys do not generate a character as such, instead a game can get the keyboard status and check the status of particular keys.
- All the keys on a keyboard can be tested.
- **Shift** and **control** keys can be tested as well.
The KeyboardState type

- The state of the keyboard is represented in XNA by a variable of type `KeyboardState`
- This type provides methods that can be used to find out if particular keys have been pressed
- They are used in a slightly different way to the properties of `GamePadState`
- But the underlying principle is the same
Creating a KeyboardState Variable

```csharp
KeyboardState keys;
```

- The game will use a variable to hold the state of the keyboard.
- It will then test the values of the keys in this variable so that the keyboard can be used to control the game.
- The variable must be declared like any other variable in the game program.
- It has been given the identifier `Keys`.
Setting the Keys Variable

```csharp
KeyboardState keys = Keyboard.GetState();
```

- The **Keyboard** class is part of XNA and provides a method called **GetState** to read the keyboard state.

- You do not need to tell **GetState** which keyboard to read as the XNA Framework only supports one keyboard.

- The **GetState** method delivers a **KeyboardState** object that holds the state of the keyboard at that instant.
Testing for a Key Press

```csharp
if (keys.IsKeyDown(Keys.R)) redIntensity++;  
```

- The method `IsKeyDown` is told which key is to be tested.
- The method returns `true` if the key is pressed.
- The above code would increase the red intensity if the R key is pressed down.
- The programmer can identify the key to be tested by using a value from the `Keys` enumeration.
The Keys enumeration

- The designers of XNA have created an enumeration which holds values that represent the keys on the keyboard.

- An enumeration is a type which can hold a particular set of values.

- They are created for use in specific situations where you only want a variable to hold particular values.

- Later you will create your own enumerations to manage the state of a game.
Enumerations and Microsoft Visual Studio

- A Visual Studio feature called “Intellisense” can suggest values from an enumeration that can be used.
- This makes it much easier for the programmer to create correct code.
The empty project that XNA creates can only be stopped by pressing the Back button.

XNA provides a method called `Exit` to stop the game.

The program above calls the `Exit` method on the running game when the Escape key is pressed.

This allows the program to be keyboard controlled.
Handling the Keyboard and Gamepad

```csharp
if (keys.IsKeyDown(Keys.R) ||
   pad1.DPad.Right == ButtonState.Pressed ||
{
    redIntensity++;
}
```

- This code can use either gamepad, Zune, or keyboard input to control the red intensity
- `Keyboard.GetState()` method always returns
- If there is no keyboard present it returns a `KeyboardState` value with no keys pressed
Advanced Logic

```csharp
if ( (keys.IsKeyDown(Keys.LeftShift) &&
    keys.IsKeyDown(Keys.RightShift)) ||
    (pad1.Buttons.LeftShoulder == ButtonState.Pressed &&
{
    redIntensity = 0;
    greenIntensity = 0;
    blueIntensity = 0;
}
```

- This code sets the intensities to 0 if both Shift keys are pressed or both shoulder buttons are pressed on the gamepad
- It combines a number of conditions to do this
Advanced Logic

```csharp
if ( (keys.IsKeyDown(Keys.LeftShift) &&
     keys.IsKeyDown(Keys.RightShift)) ||
    (pad1.Buttons.LeftShoulder == ButtonState.Pressed &&
{
    redIntensity = 0;
    greenIntensity = 0;
    blueIntensity = 0;
}
```

- The two conditions are enclosed in brackets to tell the C# compiler they need to be worked out first and then combined using an OR operator.
- The brackets work as they would in arithmetic.
1. Keyboard Light

- This program implements the Color Nerve game using the keyboard, gamepad or Dpad
- It will work on Windows PC, Zune or Xbox 360
Summary

- The XNA Framework provides a type called \texttt{KeyboardState} to represent the state of the keyboard at a particular instant.

- The \texttt{Keyboard} class provides \texttt{GetState} which returns a \texttt{KeyboardState} value.

- The \texttt{GetState} method is given a value of type \texttt{Keys} to identify the key being tested.

- A \texttt{KeyboardState} value provides a method called \texttt{IsKeyDown} which can check if a particular key is pressed.