# Games Development

© 2010 Haim Michael

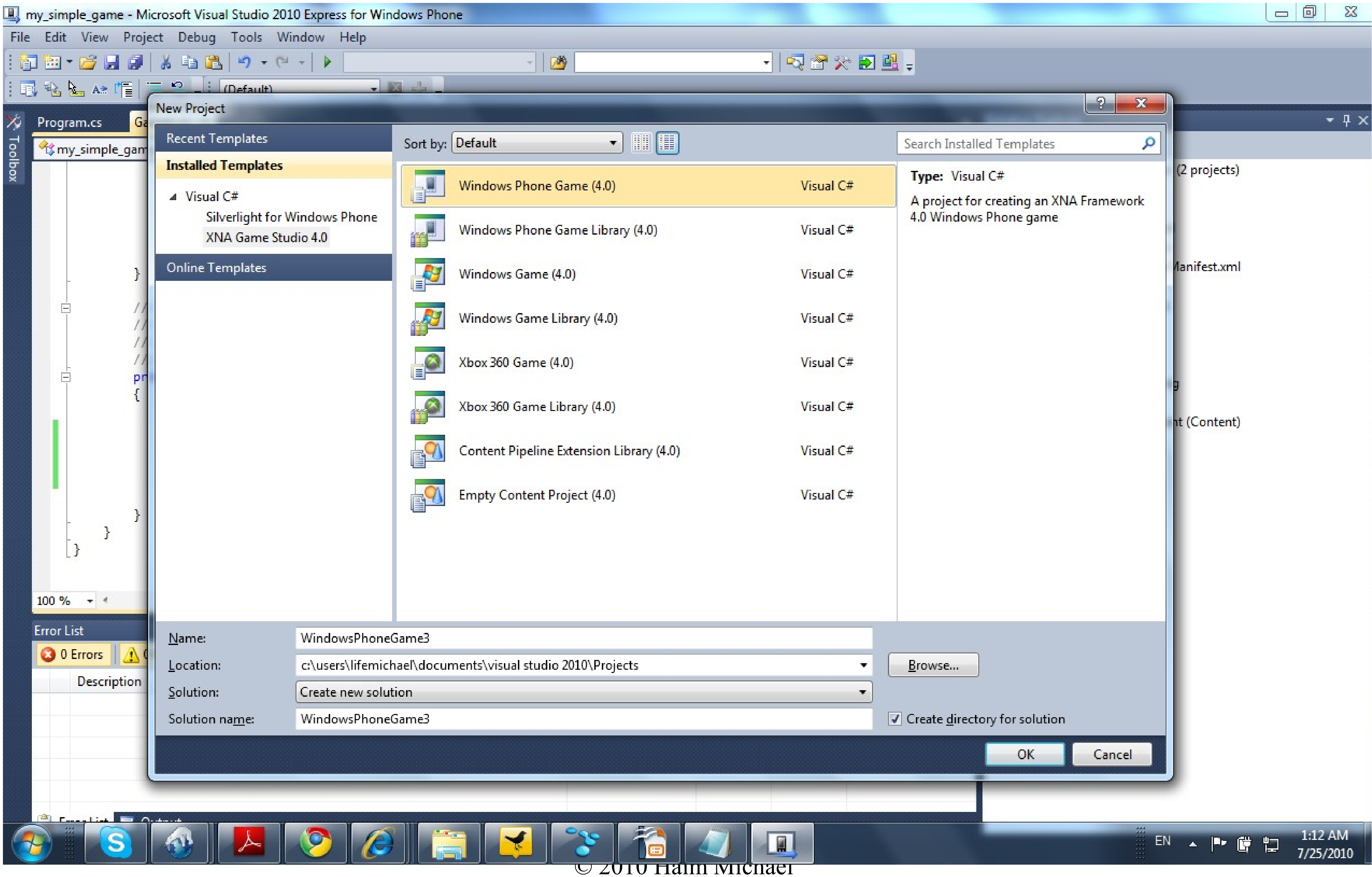
## Introduction

❖ The WP7 games development is based on the XNA framework.

## The XNA Game Studio Template

- When creating a new project we should select the XNA Game Studio Windows Phone Game template.

© 2010 Haim Michael



my_simple_game - Microsoft Visual Studio 2010 Express for Windows Phone
File Edit View Project Debug Tools Window Help
New Project
Recent Templates
Installed Templates
Visual C#
Silverlight for Windows Phone
XNA Game Studio 4.0
Online Templates
Sort by: Default
Search Installed Templates

| | |
|---|---|
| Windows Phone Game (4.0) | Visual C# |
| Windows Phone Game Library (4.0) | Visual C# |
| Windows Game (4.0) | Visual C# |
| Windows Game Library (4.0) | Visual C# |
| Xbox 360 Game (4.0) | Visual C# |
| Xbox 360 Game Library (4.0) | Visual C# |
| Content Pipeline Extension Library (4.0) | Visual C# |
| Empty Content Project (4.0) | Visual C# |

**Type:** Visual C#

A project for creating an XNA Framework 4.0 Windows Phone game

Name: WindowsPhoneGame3

Location: c:\users\lifemichael\documents\visual studio 2010\Projects

Solution: Create new solution

Solution name: WindowsPhoneGame3

☑ Create directory for solution

© 2010 Haim Michael

## The `Texture2D` Class

- Each object of this type represents a 2D grid of texels. A texel is the smallest unit that can be stored by the graphics processing unit (GPU). Each texel includes the color and the transparanecy values.

© 2010 Haim Michael

## The `Vector2` Class

- Each object of this type represents a 2D vector. In order to create a `Vector2` object we need to specify two numeric values of the `double` type.

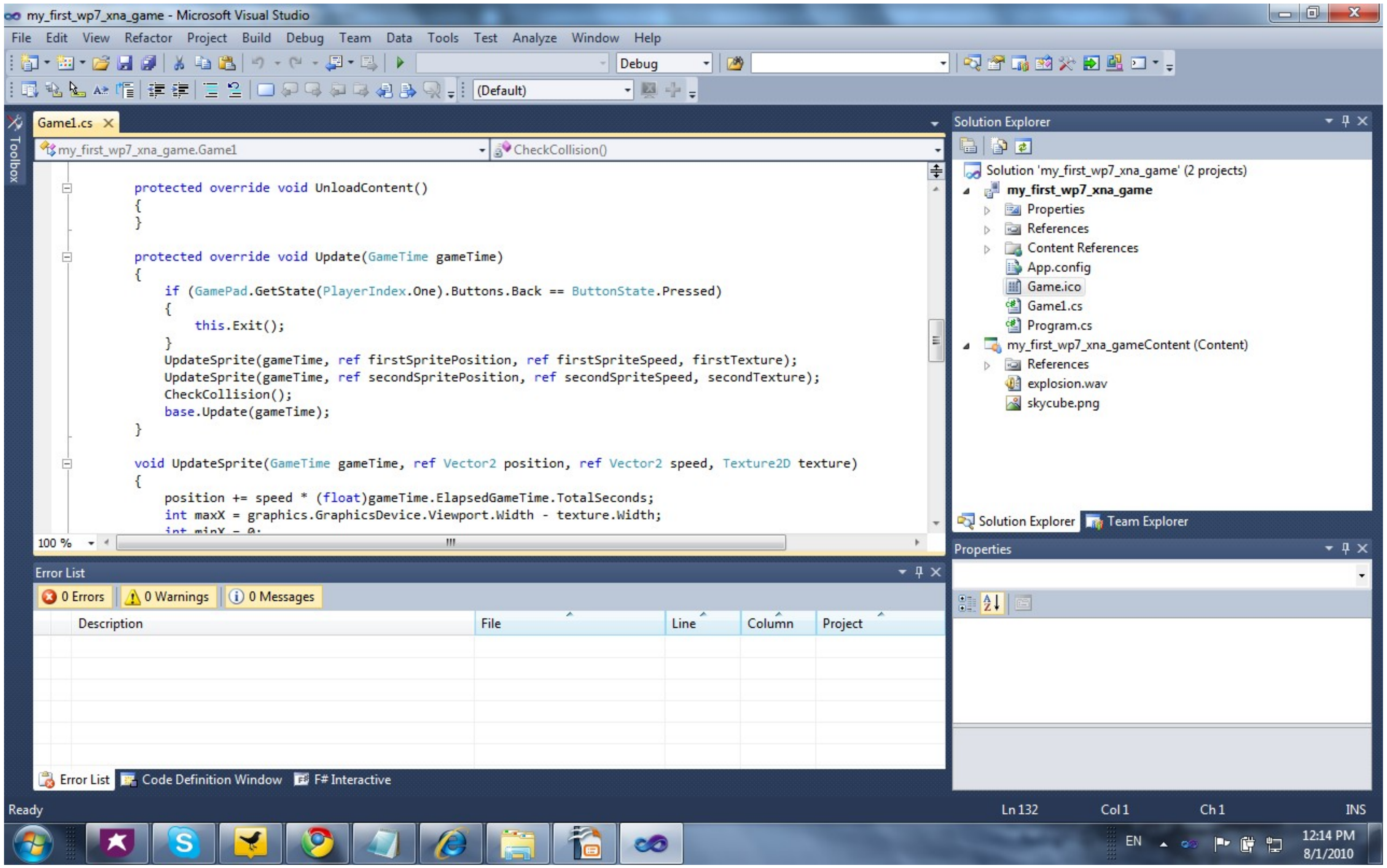
© 2010 Haim Michael

## The Game Resources

- The separation between the code and the resources ease the development process.
- The resources include image files, sound files and any other file the code uses.
- The resources are loaded into the execution of our code by calling the `Content.Load()` method.

© 2010 Haim Michael

# Simple Game



```
        protected override void UnloadContent()
        {
        }

        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            {
                this.Exit();
            }
            UpdateSprite(gameTime, ref firstSpritePosition, ref firstSpriteSpeed, firstTexture);
            UpdateSprite(gameTime, ref secondSpritePosition, ref secondSpriteSpeed, secondTexture);
            CheckCollision();
            base.Update(gameTime);
        }

        void UpdateSprite(GameTime gameTime, ref Vector2 position, ref Vector2 speed, Texture2D texture)
        {
            position += speed * (float)gameTime.ElapsedGameTime.TotalSeconds;
            int maxX = graphics.GraphicsDevice.Viewport.Width - texture.Width;
            int minX = 0;
```

© 2010 Haim Michael

## Simple Game

```
namespace my_first_wp7_xna_game
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch batch;
        Texture2D firstTexture;
        Texture2D secondTexture;
        Vector2 firstSpritePosition;
        Vector2 secondSpritePosition;
        Vector2 firstSpriteSpeed = new Vector2(40.0f, 20.0f);
        Vector2 secondSpriteSpeed = new Vector2(80.0f, 80.0f);
        int firstSpriteHeight;
        int firstSpriteWidth;
        int secondSpriteHeight;
        int secondSpriteWidth;
        SoundEffect sound;
        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
            TargetElapsedTime = TimeSpan.FromTicks(333333);
            graphics.PreferredBackBufferWidth = 480;
            graphics.PreferredBackBufferHeight = 800;
        }
```

© 2010 Haim Michael

# Simple Game

```
protected override void Initialize()
{
    base.Initialize();
}

protected override void LoadContent()
{
    batch = new SpriteBatch(GraphicsDevice);
    firstTexture = Content.Load<Texture2D>("skycube");
    secondTexture = Content.Load<Texture2D>("skycube");
    sound = Content.Load<SoundEffect>("explosion");
    firstSpritePosition.X = 0;
    firstSpritePosition.Y = 0;
    secondSpritePosition.X = graphics.GraphicsDevice.Viewport.Width
         - firstTexture.Width;
    secondSpritePosition.Y = graphics.GraphicsDevice.Viewport.Height
         - secondTexture.Height;
    firstSpriteHeight = firstTexture.Bounds.Height;
    firstSpriteWidth = firstTexture.Bounds.Width;
    secondSpriteHeight = secondTexture.Bounds.Height;
    secondSpriteWidth = secondTexture.Bounds.Width;
}
```

© 2010 Haim Michael

## Simple Game

```
protected override void UnloadContent()
{
}

protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
         ButtonState.Pressed)
    {
        this.Exit();
    }
    UpdateSprite(gameTime, ref firstSpritePosition,
         ref firstSpriteSpeed, firstTexture);
    UpdateSprite(gameTime, ref secondSpritePosition,
         ref secondSpriteSpeed, secondTexture);
    CheckCollision();
    base.Update(gameTime);
}
```

© 2010 Haim Michael

# Simple Game

```
void UpdateSprite(GameTime gameTime, ref Vector2 position,
     ref Vector2 speed, Texture2D texture)
{
    position += speed * (float)gameTime.ElapsedGameTime.TotalSeconds;
    int maxX = graphics.GraphicsDevice.Viewport.Width - texture.Width;
    int minX = 0;
    int maxY = graphics.GraphicsDevice.Viewport.Height - texture.Height;
    int minY = texture.Height/2;
    if (position.X > maxX) {
        speed.X *= -1;
        position.X = maxX;
    }
    else if (position.X < minX) {
        speed.X *= -1;
        position.X = minX;
    }
    if (position.Y > maxY) {
        speed.Y *= -1;
        position.Y = maxY;
    }
    else if (position.Y < minY) {
        speed.Y *= -1;
        position.Y = minY;
    }
}
```

© 2010 Haim Michael

# Simple Game

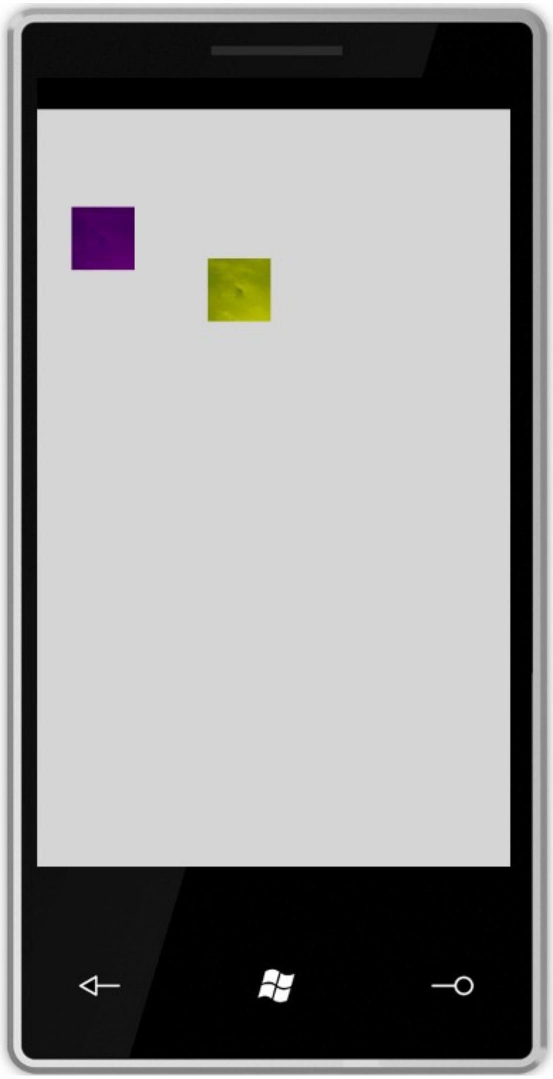
```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear(Color.LightGray);
    batch.Begin(SpriteSortMode.BackToFront, BlendState.Opaque);
    batch.Draw(firstTexture, firstSpritePosition, Color.Yellow);
    batch.End();
    batch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
    batch.Draw(secondTexture, secondSpritePosition, Color.Purple);
    batch.End();
    base.Draw(gameTime);
}
```

© 2010 Haim Michael

# Simple Game

```
        void CheckCollision()
        {
            BoundingBox firstBoundingBox = new BoundingBox(
             new Vector3(firstSpritePosition.X - (firstSpriteWidth / 2),
             firstSpritePosition.Y - (firstSpriteHeight / 2), 0),
             new Vector3(firstSpritePosition.X + (firstSpriteWidth / 2),
             firstSpritePosition.Y + (firstSpriteHeight / 2), 0));
            BoundingBox secondBoundingBox = new BoundingBox(
             new Vector3(secondSpritePosition.X - (secondSpriteWidth / 2),
             secondSpritePosition.Y - (secondSpriteHeight / 2), 0),
             new Vector3(secondSpritePosition.X + (secondSpriteWidth / 2),
             secondSpritePosition.Y + (secondSpriteHeight / 2), 0));
            if (firstBoundingBox.Intersects(secondBoundingBox)){
                 sound.Play();
            }
        }
    }
}
```

## Simple Game



© 2010 Haim Michael

## The `TouchPanel` Static Class

- The multi touch screen can detect up to four simultaneous fingers.
- We handle the touch events through the `Update` method. The `TouchPanel` static class provides us with methods we can use to obtain input.

## The `TouchPanelCapabilities` Class

- Calling the `TouchPanel.GetCapabilities` method we get a `TouchPanelCapabilities` object through which we can get information about the multi touch device.
- The `TouchPanelCapabilities` object has two properties:

  `IsConnected`

  This property returns true if the touch panel is available.

  `MaximumTouchCount`

  This property returns the maximum number of touch locations that can be tracked by the touch pad device.

© 2010 Haim Michael

# The `TouchCollection` Class

- Calling the `GetState()` static method defined in `TouchPanel` we get a `TouchCollection` object.
- The `TouchCollection` object is a collection of zero or more `TouchLocation` object.

© 2010 Haim Michael

## The `TouchLocation` Class

- Each `TouchLocation` object has the following properties:

  `State`

  This property is of the TouchLocationState enumeration type. Its possible values are Pressed, Moved and Released.

  `Position`

  This property is of the Vector2 type. It indicates the finger position.

  `Id`

  This id identifies a specific finger.

  `Pressure`

  This property returns the recorded pressure in G force.

© 2010 Haim Michael

# The `TouchLocation` Class

- When we don't touch the screen the `TouchCollection` will be empty.
- When the first finger touches the screen the `TouchCollection` will contain a single `TouchLocation` object with a `State` equals to `Pressed`.
- Each subsequent call to `TouchPanel.GetState` will return a `TouchCollection` with a `TouchLocation` object that its `State` is `Moved` even if the finger doesn't really move.

© 2010 Haim Michael

## The `TouchLocation` Class

- When the finger is lifted from the screen the State of the `TouchLocation` object is changed into `Released`.
- Each subsequent call to `TouchPanel.GetState` will return an empty collection.
- When tapping the screen fast enough we might get a `TouchLocation` object with a `State` equals to `Pressed` followed with a `TouchLocation` object with a `State` equals to `Released` without having any `Moved` state.

© 2010 Haim Michael

## Tracking Particular Fingers

- We can use the `Id` property in order to track particular fingers.
- We can easily track each finger by using a `Dictionary` object.

© 2010 Haim Michael

## Tracking Specific Finger Changes

- When getting a `TouchLocation` object we can call the `TryGetPreviousLocation` method on it.

```
...
TouchLocation previousTouchLocation;
bool success = touchLocation.TryGetPreviousLocation(
    out previousTouchLocation);
...
```

- Through calling this we can obtain the previous location and calculate the difference.

## Tracking Specific Finger Changes

- If the user has just touched the screen then the  method `TryGetPreviousLocation` method will return false, and the `State` of the `TouchLocation` object that describes the previous location will be `Invalid`.

© 2010 Haim Michael

## The `Update` Method Code

- We should place the code that checks the touch screen within the Update method.

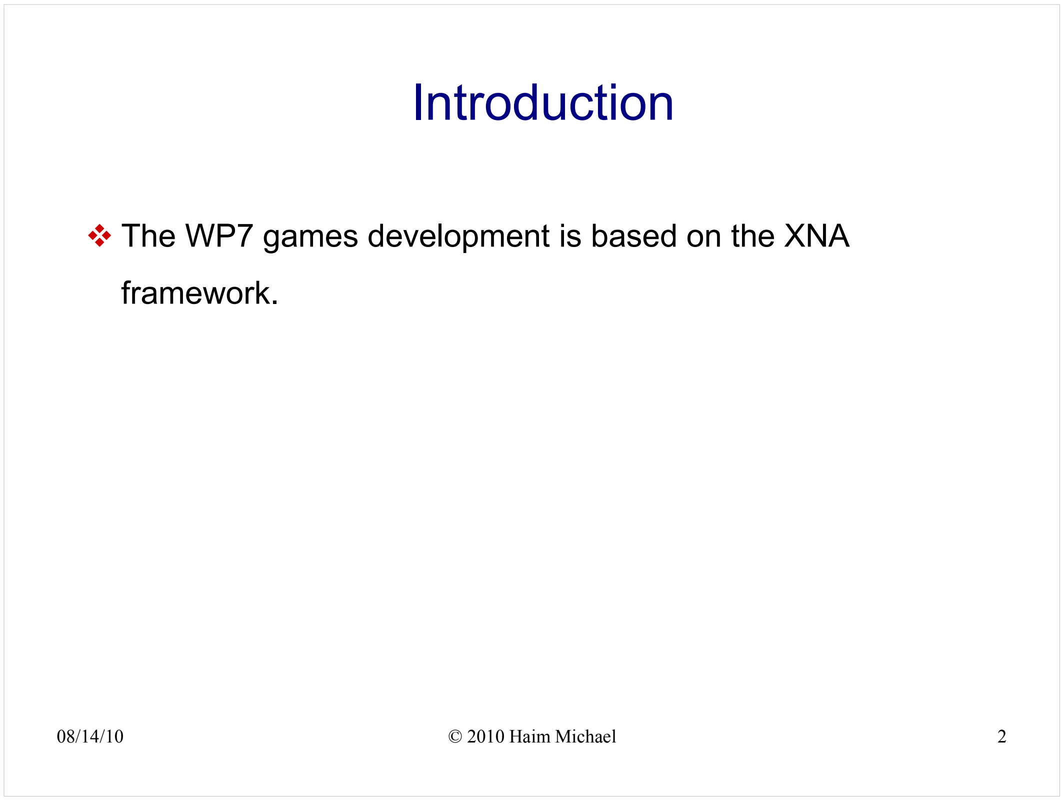
© 2010 Haim Michael

## The Update Method Code

```
protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();
    TouchCollection touchLocations = TouchPanel.GetState();
    foreach (TouchLocation touchLocation in touchLocations)
    {
        if (touchLocation.State == TouchLocationState.Pressed)
        {
            Vector2 touchPosition = touchLocation.Position;
            ...
        }
    }
    base.Update(gameTime);
}
```

© 2010 Haim Michael

# Games Development



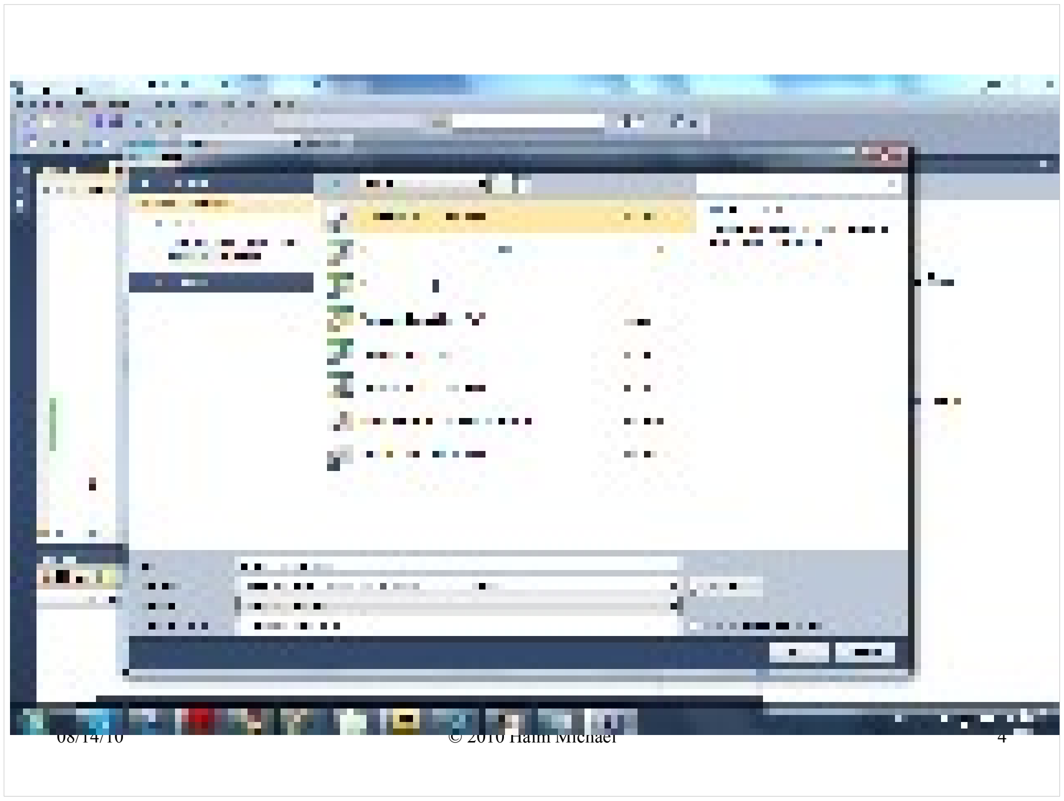
## Introduction

❖ The WP7 games development is based on the XNA framework.

08/14/10 © 2010 Haim Michael 2

## The XNA Game Studio Template

- When creating a new project we should select the XNA Game Studio Windows Phone Game template.

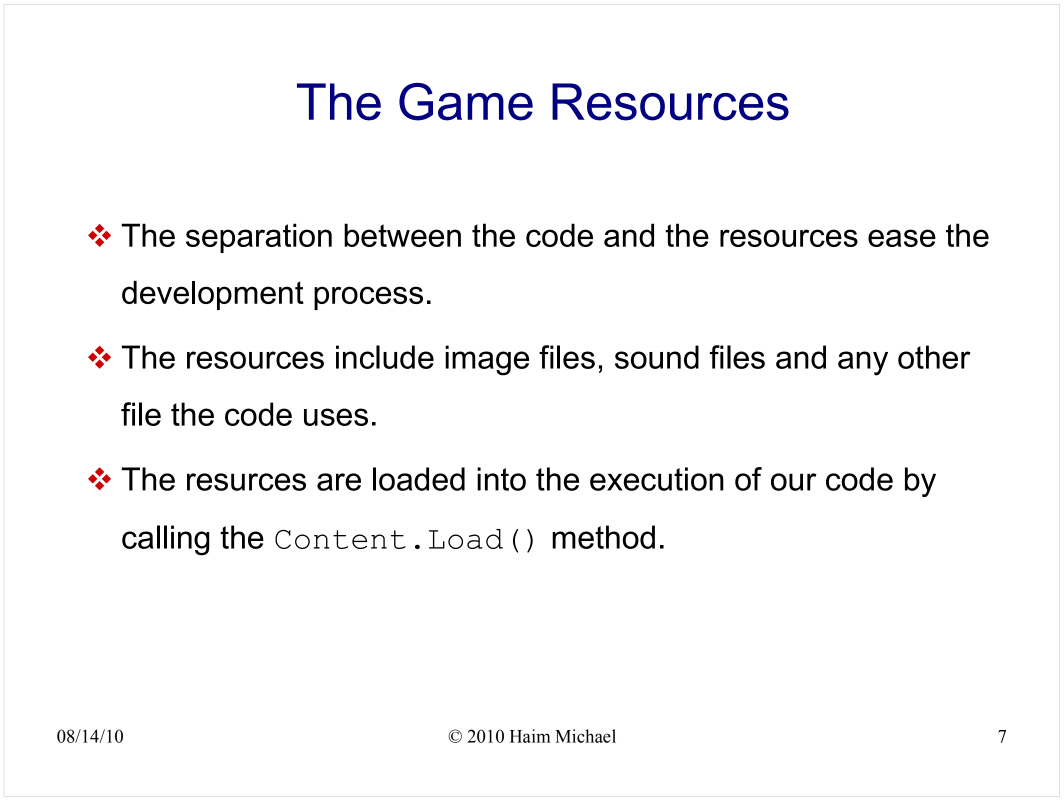


## The `Texture2D` Class

❖ Each object of this type represents a 2D grid of texels. A texel is the smallest unit that can be stored by the graphics processing unit (GPU). Each texel includes the color and the transparanecy values.

## The `Vector2` Class

- Each object of this type represents a 2D vector. In order to create a `Vector2` object we need to specify two numeric values of the `double` type.

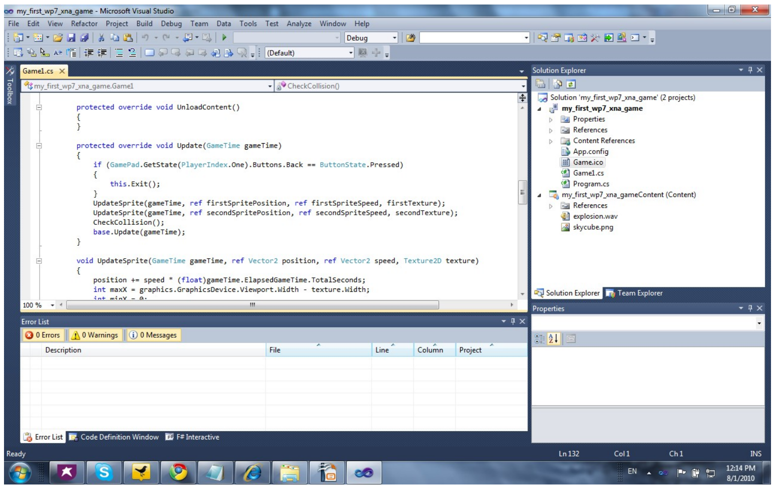


## The Game Resources

- ❖ The separation between the code and the resources ease the development process.
- ❖ The resources include image files, sound files and any other file the code uses.
- ❖ The resources are loaded into the execution of our code by calling the `Content.Load()` method.

08/14/10 © 2010 Haim Michael 7

## Simple Game

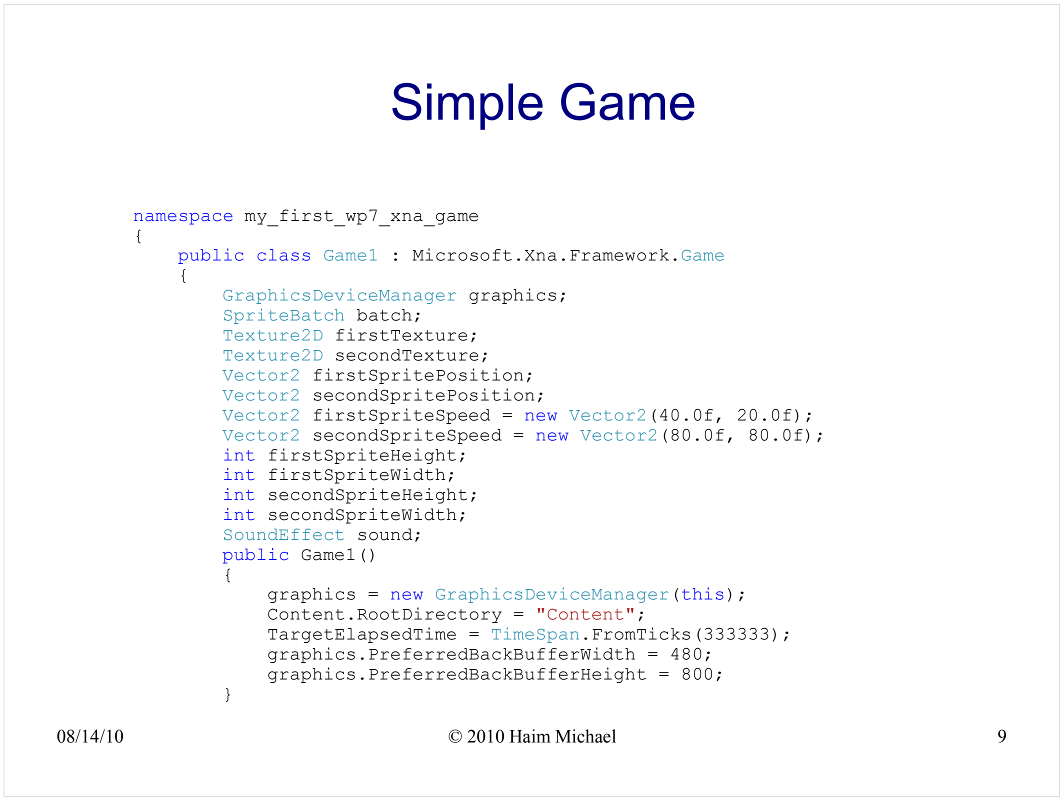


```
        protected override void UnloadContent()
        {
        }

        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            {
                this.Exit();
            }
            UpdateSprite(gameTime, ref firstSpritePosition, ref firstSpriteSpeed, firstTexture);
            UpdateSprite(gameTime, ref secondSpritePosition, ref secondSpriteSpeed, secondTexture);
            CheckCollision();
            base.Update(gameTime);
        }

        void UpdateSprite(GameTime gameTime, ref Vector2 position, ref Vector2 speed, Texture2D texture)
        {
            position += speed * (float)gameTime.ElapsedGameTime.TotalSeconds;
            int maxX = graphics.GraphicsDevice.Viewport.Width - texture.Width;
```

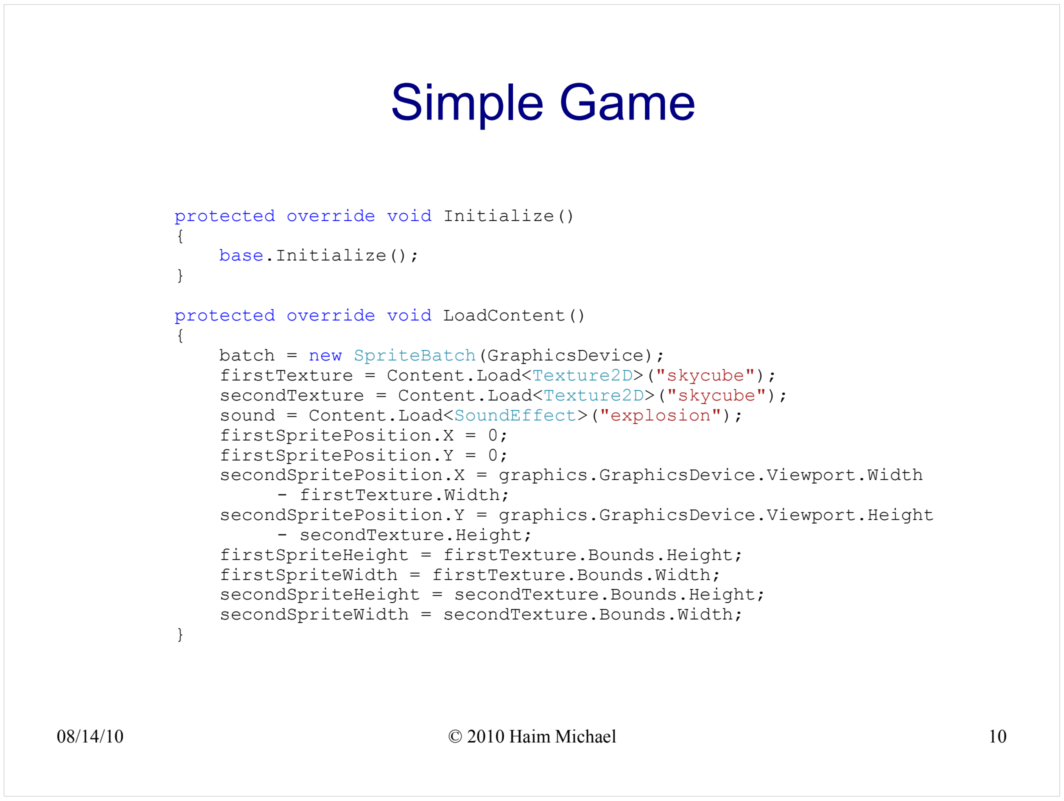
```
Solution 'my_first_wp7_xna_game' (2 projects)
  my_first_wp7_xna_game
    Properties
    References
    Content References
    App.config
    Game.ico
    Game1.cs
    Program.cs
  my_first_wp7_xna_gameContent (Content)
    References
    explosion.wav
    skycube.png
```



## Simple Game

```
namespace my_first_wp7_xna_game
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch batch;
        Texture2D firstTexture;
        Texture2D secondTexture;
        Vector2 firstSpritePosition;
        Vector2 secondSpritePosition;
        Vector2 firstSpriteSpeed = new Vector2(40.0f, 20.0f);
        Vector2 secondSpriteSpeed = new Vector2(80.0f, 80.0f);
        int firstSpriteHeight;
        int firstSpriteWidth;
        int secondSpriteHeight;
        int secondSpriteWidth;
        SoundEffect sound;
        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
            TargetElapsedTime = TimeSpan.FromTicks(333333);
            graphics.PreferredBackBufferWidth = 480;
            graphics.PreferredBackBufferHeight = 800;
        }
```

08/14/10 © 2010 Haim Michael 9

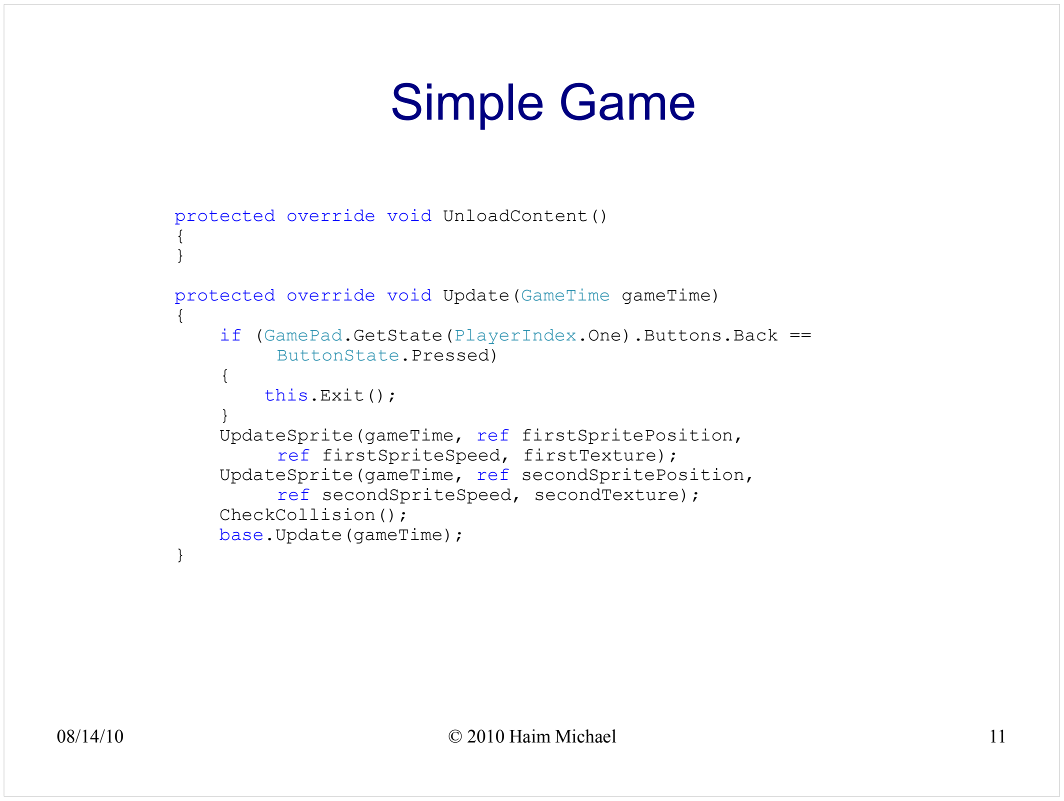


## Simple Game

```
protected override void Initialize()
{
    base.Initialize();
}

protected override void LoadContent()
{
    batch = new SpriteBatch(GraphicsDevice);
    firstTexture = Content.Load<Texture2D>("skycube");
    secondTexture = Content.Load<Texture2D>("skycube");
    sound = Content.Load<SoundEffect>("explosion");
    firstSpritePosition.X = 0;
    firstSpritePosition.Y = 0;
    secondSpritePosition.X = graphics.GraphicsDevice.Viewport.Width
         - firstTexture.Width;
    secondSpritePosition.Y = graphics.GraphicsDevice.Viewport.Height
         - secondTexture.Height;
    firstSpriteHeight = firstTexture.Bounds.Height;
    firstSpriteWidth = firstTexture.Bounds.Width;
    secondSpriteHeight = secondTexture.Bounds.Height;
    secondSpriteWidth = secondTexture.Bounds.Width;
}
```

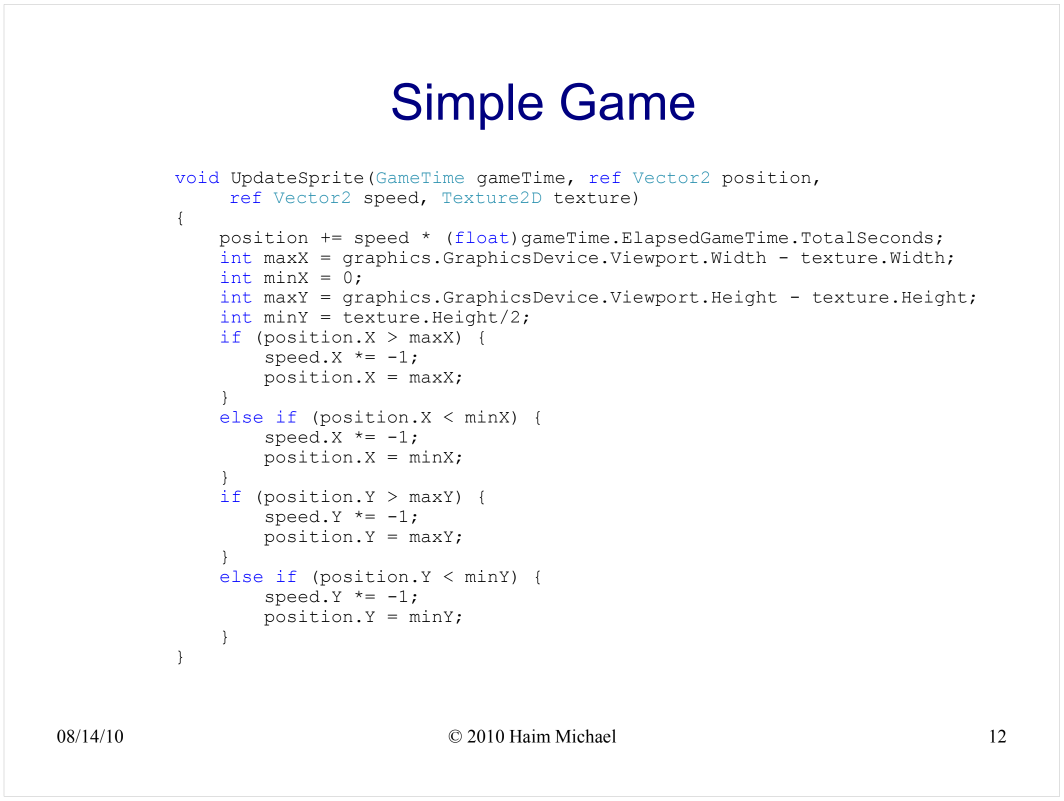
08/14/10 © 2010 Haim Michael 10



## Simple Game

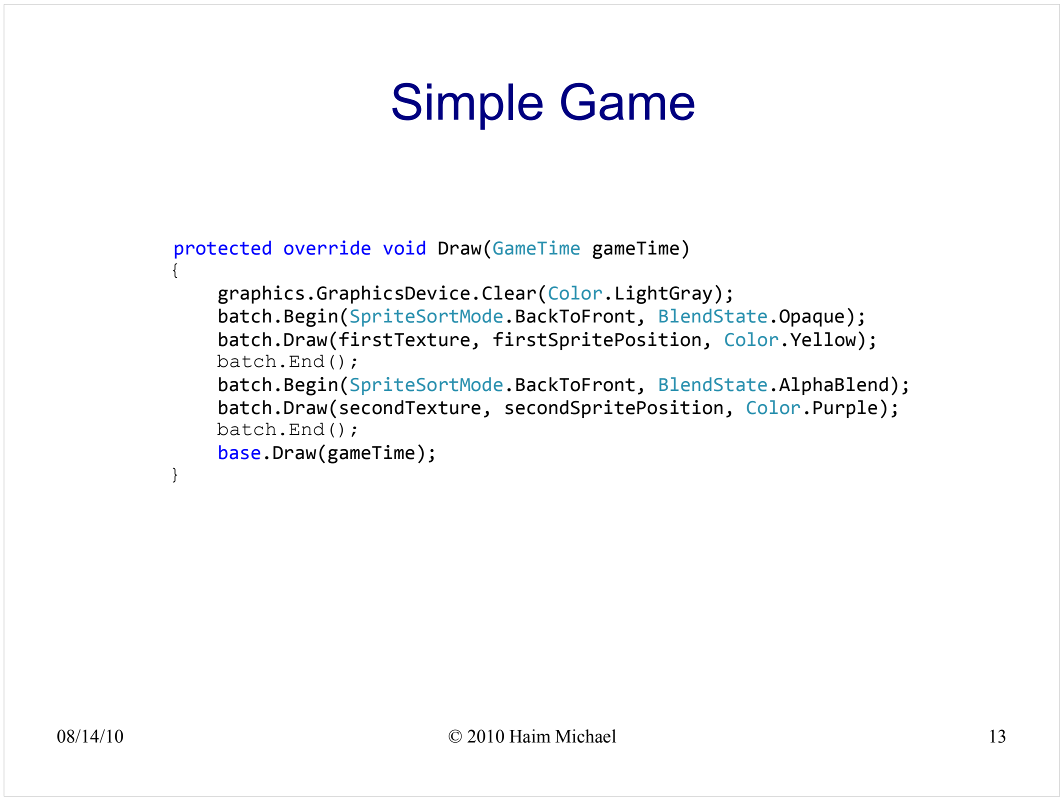
```
protected override void UnloadContent()
{
}

protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
         ButtonState.Pressed)
    {
        this.Exit();
    }
    UpdateSprite(gameTime, ref firstSpritePosition,
         ref firstSpriteSpeed, firstTexture);
    UpdateSprite(gameTime, ref secondSpritePosition,
         ref secondSpriteSpeed, secondTexture);
    CheckCollision();
    base.Update(gameTime);
}
```



## Simple Game

```
void UpdateSprite(GameTime gameTime, ref Vector2 position,
     ref Vector2 speed, Texture2D texture)
{
    position += speed * (float)gameTime.ElapsedGameTime.TotalSeconds;
    int maxX = graphics.GraphicsDevice.Viewport.Width - texture.Width;
    int minX = 0;
    int maxY = graphics.GraphicsDevice.Viewport.Height - texture.Height;
    int minY = texture.Height/2;
    if (position.X > maxX) {
        speed.X *= -1;
        position.X = maxX;
    }
    else if (position.X < minX) {
        speed.X *= -1;
        position.X = minX;
    }
    if (position.Y > maxY) {
        speed.Y *= -1;
        position.Y = maxY;
    }
    else if (position.Y < minY) {
        speed.Y *= -1;
        position.Y = minY;
    }
}
```



## Simple Game

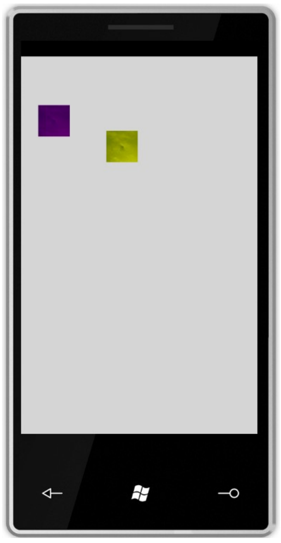
```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear(Color.LightGray);
    batch.Begin(SpriteSortMode.BackToFront, BlendState.Opaque);
    batch.Draw(firstTexture, firstSpritePosition, Color.Yellow);
    batch.End();
    batch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
    batch.Draw(secondTexture, secondSpritePosition, Color.Purple);
    batch.End();
    base.Draw(gameTime);
}
```

08/14/10 © 2010 Haim Michael 13

## Simple Game

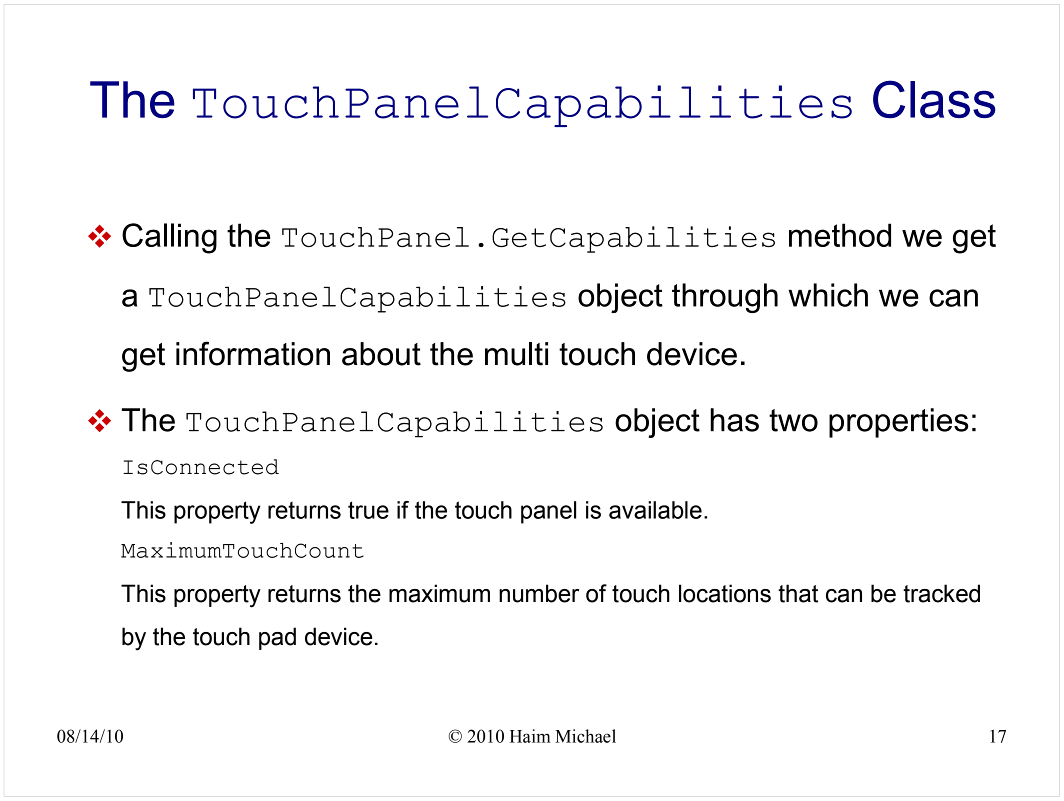
```
            void CheckCollision()
            {
                BoundingBox firstBoundingBox = new BoundingBox(
                 new Vector3(firstSpritePosition.X - (firstSpriteWidth / 2),
                 firstSpritePosition.Y - (firstSpriteHeight / 2), 0),
                 new Vector3(firstSpritePosition.X + (firstSpriteWidth / 2),
                 firstSpritePosition.Y + (firstSpriteHeight / 2), 0));
                BoundingBox secondBoundingBox = new BoundingBox(
                 new Vector3(secondSpritePosition.X - (secondSpriteWidth / 2),
                 secondSpritePosition.Y - (secondSpriteHeight / 2), 0),
                 new Vector3(secondSpritePosition.X + (secondSpriteWidth / 2),
                 secondSpritePosition.Y + (secondSpriteHeight / 2), 0));
                if (firstBoundingBox.Intersects(secondBoundingBox)){
                     sound.Play();
                }
            }
        }
    }
```

## Simple Game



## The `TouchPanel` Static Class

- The multi touch screen can detect up to four simultaneous fingers.
- We handle the touch events through the `Update` method. The `TouchPanel` static class provides us with methods we can use to obtain input.



## The `TouchPanelCapabilities` Class

- Calling the `TouchPanel.GetCapabilities` method we get a `TouchPanelCapabilities` object through which we can get information about the multi touch device.
- The `TouchPanelCapabilities` object has two properties:

  `IsConnected`

  This property returns true if the touch panel is available.

  `MaximumTouchCount`

  This property returns the maximum number of touch locations that can be tracked by the touch pad device.

08/14/10 © 2010 Haim Michael 17

## The `TouchCollection` Class

- Calling the `GetState()` static method defined in `TouchPanel` we get a `TouchCollection` object.
- The `TouchCollection` object is a collection of zero or more `TouchLocation` object.

## The `TouchLocation` Class

- Each `TouchLocation` object has the following properties:

  `State`

  This property is of the TouchLocationState enumeration type. Its possible values are Pressed, Moved and Released.

  `Position`

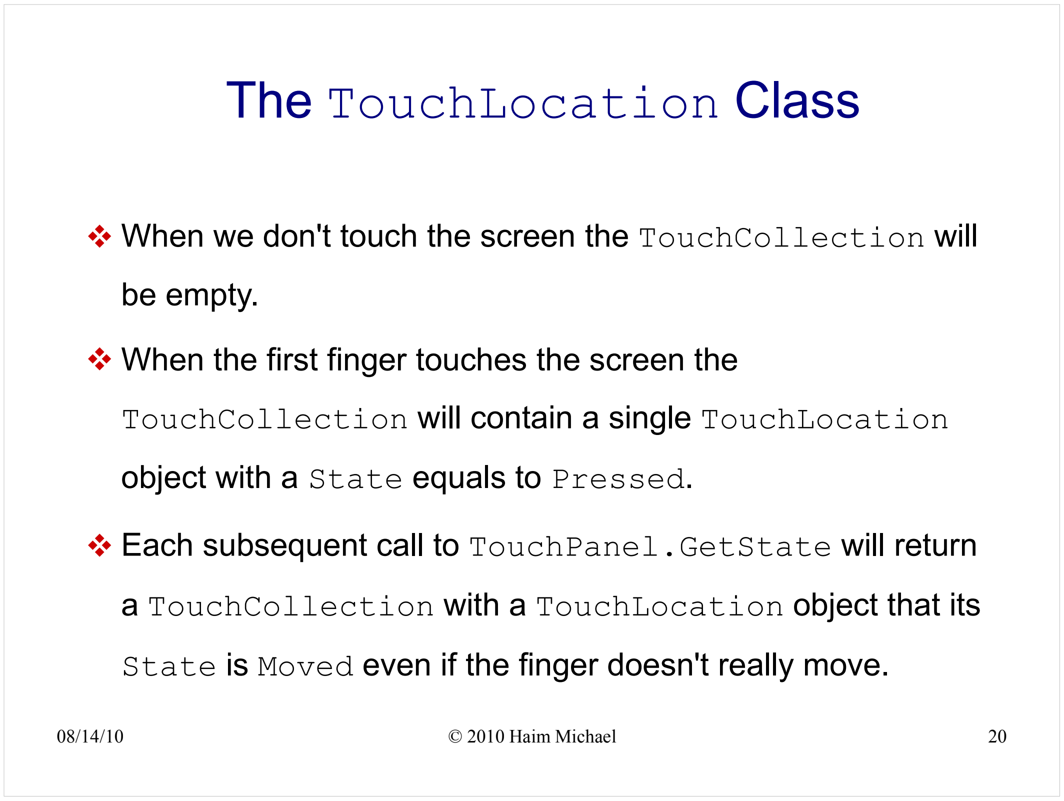
  This property is of the Vector2 type. It indicates the finger position.

  `Id`

  This id identifies a specific finger.

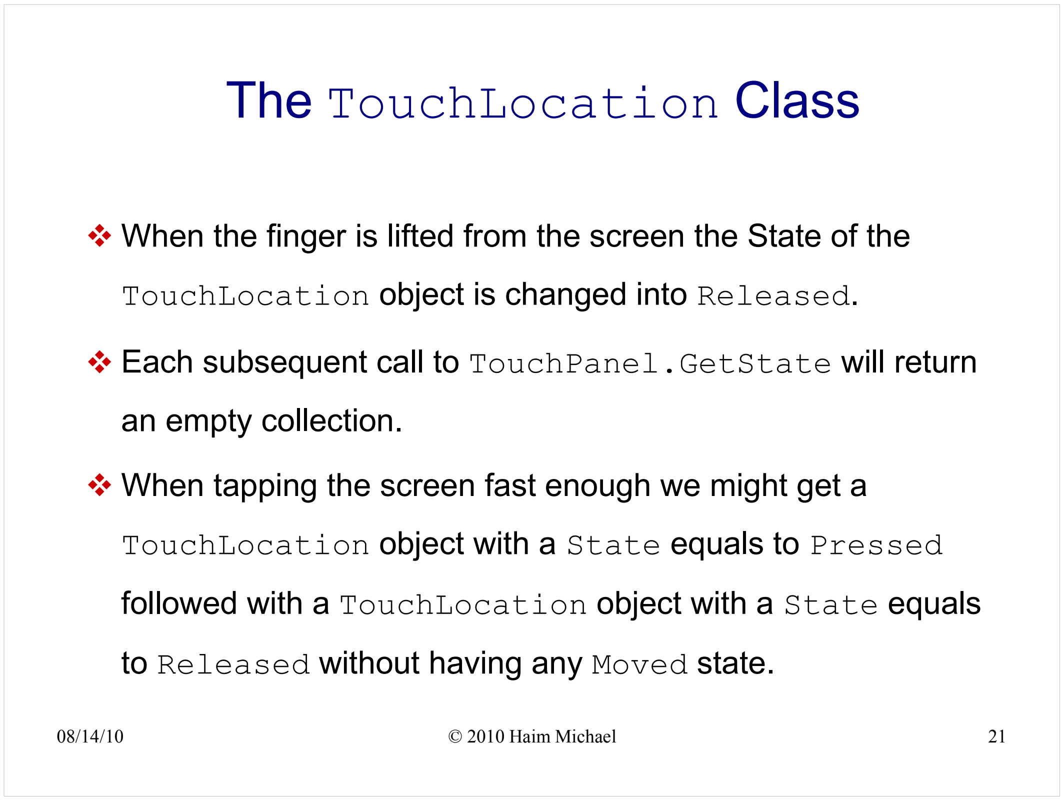
  `Pressure`

  This property returns the recorded pressure in G force.



## The `TouchLocation` Class

- When we don't touch the screen the `TouchCollection` will be empty.
- When the first finger touches the screen the `TouchCollection` will contain a single `TouchLocation` object with a `State` equals to `Pressed`.
- Each subsequent call to `TouchPanel.GetState` will return a `TouchCollection` with a `TouchLocation` object that its `State` is `Moved` even if the finger doesn't really move.



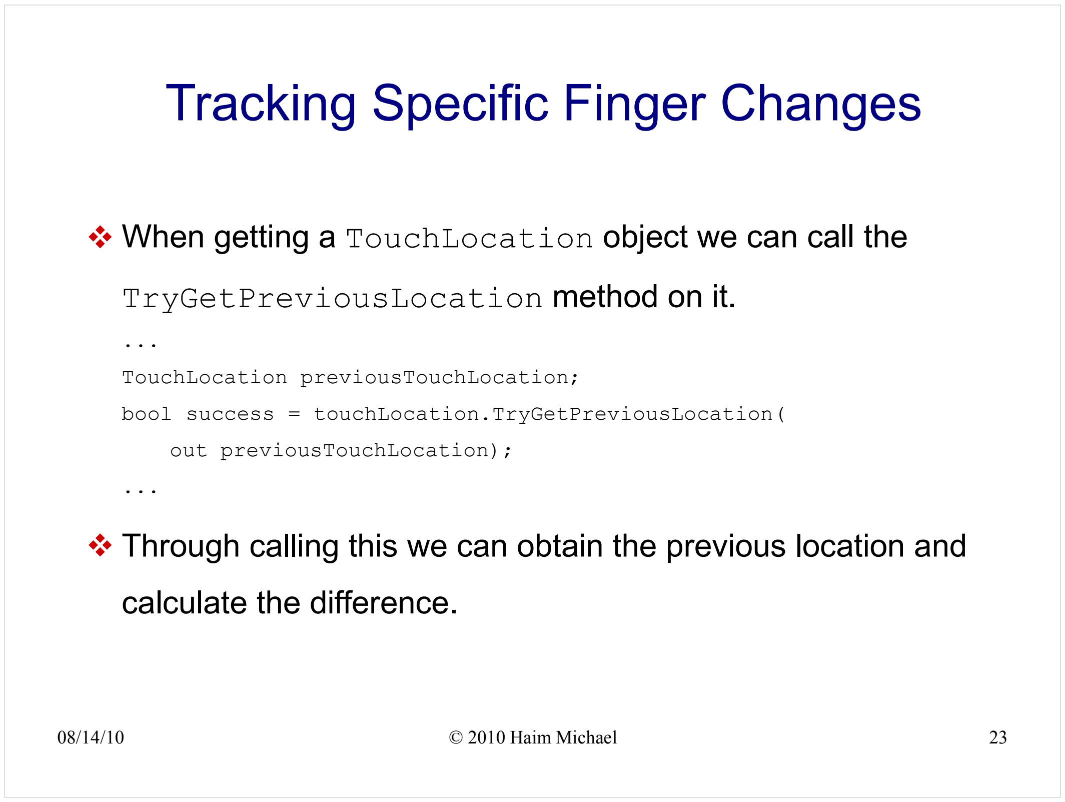
## The `TouchLocation` Class

- When the finger is lifted from the screen the State of the `TouchLocation` object is changed into `Released`.
- Each subsequent call to `TouchPanel.GetState` will return an empty collection.
- When tapping the screen fast enough we might get a `TouchLocation` object with a `State` equals to `Pressed` followed with a `TouchLocation` object with a `State` equals to `Released` without having any `Moved` state.

08/14/10 © 2010 Haim Michael 21

## Tracking Particular Fingers

- We can use the `Id` property in order to track particular fingers.
- We can easily track each finger by using a `Dictionary` object.



## Tracking Specific Finger Changes

- When getting a `TouchLocation` object we can call the `TryGetPreviousLocation` method on it.

```
...
TouchLocation previousTouchLocation;
bool success = touchLocation.TryGetPreviousLocation(
    out previousTouchLocation);
...
```

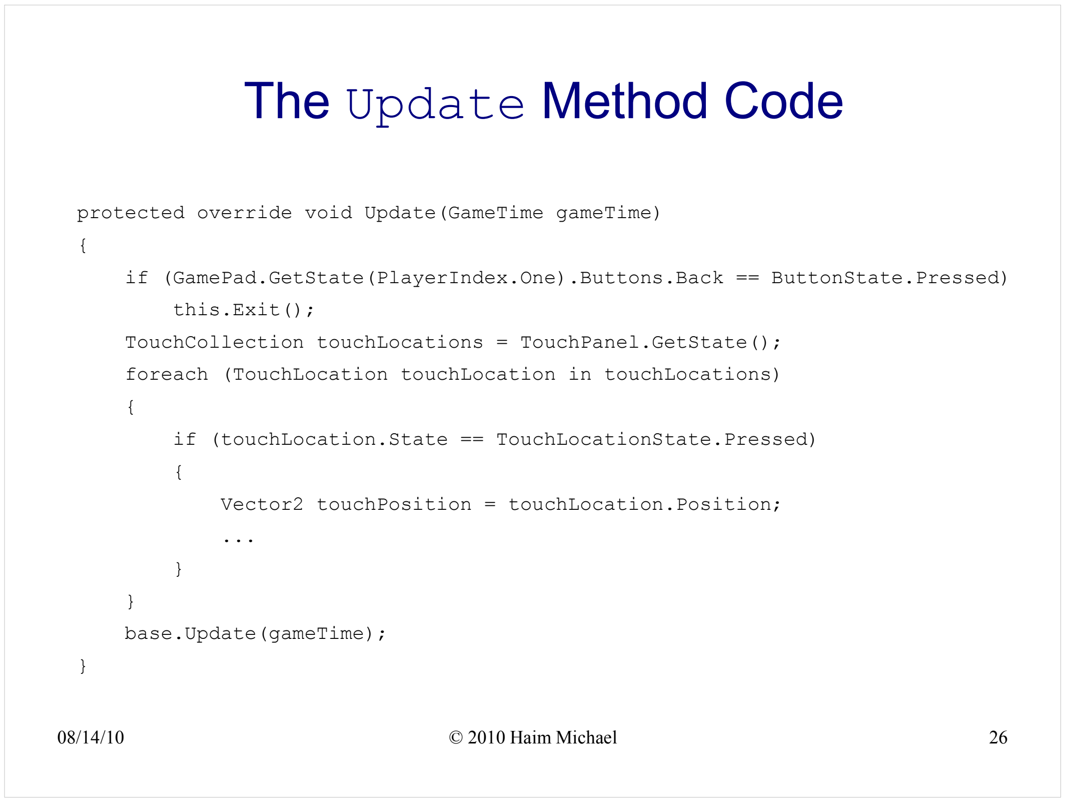
- Through calling this we can obtain the previous location and calculate the difference.

## Tracking Specific Finger Changes

- If the user has just touched the screen then the method `TryGetPreviousLocation` method will return false, and the `State` of the `TouchLocation` object that describes the previous location will be `Invalid`.

## The `Update` Method Code

❖ We should place the code that checks the touch screen within the Update method.



## The `Update` Method Code

```
protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();
    TouchCollection touchLocations = TouchPanel.GetState();
    foreach (TouchLocation touchLocation in touchLocations)
    {
        if (touchLocation.State == TouchLocationState.Pressed)
        {
            Vector2 touchPosition = touchLocation.Position;
            ...
        }
    }
    base.Update(gameTime);
}
```