Best Practices: Preparing Games for the Windows Phone 7 Marketplace

The practices discussed here will help you have the most success possible in the marketplace with your Windows Phone 7 game. Merely meeting the expectations of millions of potential customers may not result in success, but exceeding expectations may result in a popular game that is purchased and played by many customers.

# Take Time to Polish Your Game

Because best practices is such a broad topic, it is hard to say what is the most important piece. It is also difficult to over emphasize how important polish is for a successful game. The best game ideas and the most stable code do not compare to a game with an extra level of polish.

Polish can be defined as putting in the extra effort to make your game look and feel its best. It also is:

* The difference between a basic menu with buttons that work and a basic menu with polish. It may take time to add a small animation flourish for each button press, sound effect, and styled buttons, but doing so makes a big difference.
* Smooth menu operation and intuitive controls.
* Smooth transitions among screens, modes, and levels.

If you get only one thing from this article, it should be this: *polish makes a good game great.*

# Build Intuitive and Fun Controls

Avoid simulating traditional controls such as thumbsticks. That type of control takes away useful space from the gameplay area and is not platform friendly. Use gestures such as **Tap**, **Pan**, **Flick**, **Pinch,** and **Stretch** for user input. Games that are engaging and naturally fun have controls that are natural to the platform.

| **Control** | **Description** |
| --- | --- |
| **Touch** | Touch control systems will feel natural to Windows Phone 7 users. Design games from the beginning to take full advantage of the touch screen. The touch screen is the primary way users interact with their phone, and users expect to interact with games in the same way. |
| **Back Button** | Although there are other buttons on the device, only the **Back** button is available to the game. Use the **Back** button for pausing and exiting the game. |
| **Gestures** | Design your gameplay to use touch gestures in natural ways. For example: allowing players to draw paths on the screen to direct gameplay, or allowing for group selection by stretching an on-screen rectangle around play pieces. Consider allowing navigation by dragging the landscape with a **Flick**, and allowing users to rotate by touching and rotating two fingers.  For example code, see the [Gestures Article](http://create.msdn.com/en-US/education/catalog/article/gestures). |

# Take Advantage of Device Sensors

All Windows Phone hardware has a standard suite of sensors: A-GPS, accelerometer, and a camera. Use these sensors to create a fun experience and richer gameplay.

|  |  |
| --- | --- |
| **Sensor** | **Description** |
| **A-GPS** | Use the A-GPS or the compass to create location or direction-aware games, bringing the user experience and the game closer to real life. |
| **Accelerometer** | Use the accelerometer as an alternative device to move objects. Think of interesting ways to use the tilt or motion of the device to affect gameplay. Your game, for example, might feature a way to change the view angle by changing the angle at which the phone is held, or use physics-aware code to provide a more realistic experience. |
| **Camera** | Use the camera and allow real images to become part of the game⎯or even the whole game! For example, take a picture and enable the user to modify and draw over it. |

# Support Changing Screen orientation

Windows Phone 7 supports three screen orientation views: portrait, landscape left, and landscape right. Portrait view is the default view for applications. The **Start** button is always presented in portrait view. In portrait view, the page is oriented vertically with the steering buttons displayed at the bottom of the phone.

In both landscape views, the **Status Bar** and **Application Bar** remain on the side of the screen that has the **Power** and **Start** buttons. Landscape left has the **Status Bar** on the left, and landscape right has the **Status Bar** on the right.

Routinely check current screen orientation and enable gameplay regardless of how the phone is held. See the [Screen Orientation](http://create.msdn.com/en-US/education/catalog/sample/orientation) Sample for a full example of monitoring changes between portrait and landscape orientations.

# Audio Tips

Audio can enrich an application and add needed polish. Playing audio should also dependent on user preference. Consider these tips when building audio into your game:

* Play sound effects at an average volume to avoid forcing the player to adjust the phone's volume during the game.
* Allow sound effects and background music to be turned on and off by users.
* Play directional sounds that reflect a location of the originating element on the screen.
* Pay attention to audio related requirements. For example, if a game overrides a user's current music choice, the game may fail certification. Games can give the user the option to change the currently playing music from within the game.

# If Your Game Is 3D, Take Advantage of Hardware Acceleration

Windows Phone 7 includes a hardware image scaler. You draw whatever size you prefer and the scaler adjusts the resulting image to fit the screen.

Resolution can range from 240 x 240 to a maximum of 800 x 480. For a portrait game, the resolution can be 480 x 800. If you choose a resolution that does not match the screen aspect ratio, the game automatically is letterboxed.

To use the scaler, set **graphics.PreferredBackBufferWidth** and **PreferredBackBufferHeight** in your game constructor. Touch input is scaled automatically to match your chosen back buffer resolution, and scaling is implemented by dedicated hardware so it does not cost any GPU.

XNA Game Studio 4.0 provides a number of prebuilt shaders designed specifically for Windows Phone. This section is not meant to imply that 2D rendering with **BasicEffect** occurs faster than using **SpriteBatch**. **SpriteBatch** also is GPU accelerated, meaning rendering with **BasicEffect** is not automatically faster.

| **Class** | **Description** |
| --- | --- |
| **BasicEffect** | Provides single texturing capabilities combined with configurable dynamic lighting. |
| **DualTextureEffect** | Provides blending capabilities between two source textures, and enables detail texturing or light mapping by using prebaked light maps. |
| **EnvironmentMappedEffect** | Uses the camera's view projection to compute the lookup into an environment texture map. Coupled with settings for Fresnel, this effect helps deliver high-quality shiny objects at a very low cost; it can represent a range of materials from fully-mirrored to glossy. |
| **AlphaTestEffect** | This general purpose alpha test effect employs a single texture that contains opacity values in the alpha channel to kill pixels being rendered. While basic 2D alpha blending may be free on the device, there are costs for the use of this shader effect. Consider appropriate uses such as for rendering foliage or compositing imposters. |
| **SkinnedEffect** | Enables use of hardware-accelerated skinning. While software skinning is possible on the platform, hardware skinning provides richer and more detailed meshes in cases where the work can be offloaded to the graphics processing unit (GPU). |

# Respond Correctly to Back Button Use

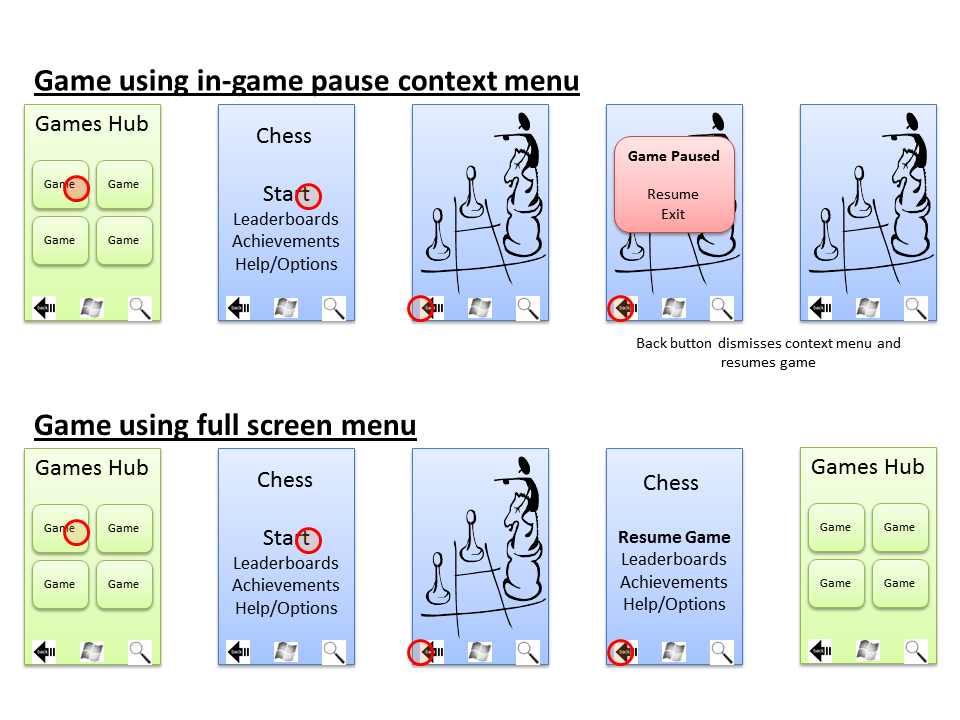
All Windows Phone 7 games must respond to use of the **Back** button. Windows Phone 7 consistently uses **Back** to move backward through the system UI. Games must implement this behavior as follows:

* **During gameplay**, the game can do one of the following:
  + Present a contextual pause menu (dialog). Pressing the **Back** button again while in the pause menu closes the menu and resumes the game.
  + Navigate the user to the prior menu screen. Pressing the **Back** button again should continue to return the user to the previous menu or page.

In addition to allowing the player to resume the game from the pause menu, games candisplay some, or all, of the standardized start screen options from this menu. It is a good practice in case the player exits the game to automatically save the game state while the pause menu is shown.

* **Outside of gameplay**, such as within the game's menu system, pressing **Back** must return to the previous menu or page.
* **At the game’s initial (start) screen**, pressing **Back** must exit the game.

The desired behavior is shown by the following diagram, which depicts the results of the player tapping menu items and pressing the **Back** button within a game. In the diagram, player actions are represented by a red circle.



# Diligently Save Your Game State

On a Windows Phone, a game might be exited at any time. An incoming call may interrupt gameplay, or the user might quit the game by using the **Home** or **Search** buttons to use other applications. We recommend saving your game's state whenever possible to protect the user's time investment in your game. We also recommend that you make a distinction between the automatically saved game state and the user's explicitly saved games. Automatically saved games should be viewed as a backup in case the game ends unexpectedly, but should not replace the user's ability to save the game at a chosen time or place.

If you implement automatic game saving, check for an automatically saved state when the game launches. If found, let the user choose to resume the game from the automatically saved state or from a specific manually saved game, if present. During the save process, we also recommend that you display a visual cue warning users not to press the **Search** or **Home** button because the action could cause the game to exit before the save is complete.

# A Few Final Things to Remember

* Make your trial mode great—showcase the best features of your game.
* Take advantage of all the sample code on this site and the help available in the forums.