Best Practices for Gaming Applications

Version 1.1a

Copyright
© Copyright 2003-2004 Tapwave, Inc. All Rights Reserved. Tapwave is a registered trademark of Tapwave, Inc. in the United States and/or other countries. The Palm logo, HotSync, Palm OS, Palm, Palm Powered, and the Palm Powered logo are registered trademarks of PalmSource, Inc., and its affiliates. X-Forge is a trademark of Fathammer, Ltd. Java is a registered trademark of Sun Microsystems, Inc. Windows is a registered trademark of Microsoft Corporation, Inc. All other brands are trademarks or registered trademarks of their respective owners.

1.Overview

The purpose of this document is to describe the development practices recommended by Tapwave. By following these recommendations, your application will provide the user with a common experience and will make effective and efficient use of unique Tapwave features.

2.Sound

- Respect the Master Sound Level system preference. Use `TwSndSetVolume` and `TwSndGetVolume` if you provide your own user interface for managing the master volume. See the *Tapwave Advanced Sound API Reference* document for more information.
- Support muting the game's sound without impacting the system's sound level/mute settings.

3.High Score

- Call the High Score API when a new high score has been achieved, or if you prefer you can call this API for every score, The API automatically rejects scores that are lower than the lowest high score and posts each application's **highest** score, of each score type, to Tapwave.com.
- Add support for tournament mode and decide whether your tournament is based on time, score, number of kills, etc. Alternately, you can use multiple score types to support several types of tournaments. The user should access the tournament functionality through a *Tournament* menu item. Be sure to disable cheats and turn off any other game features that could give a user an advantage over another user when tournament mode is enabled. See "High Score" in the *Tapwave Programmer's Reference* for more information.
- Register a description of your game and it's creator ID at the Tapwave Developer Zone. This is necessary to correctly map your game with its creator ID as reported by the high score conduit.

4.Controls

- A game must not override the behavior of the *Home*, *Power*, or *Bluetooth* buttons.
- The *Home* button should always return the user to the launcher (also called the *Home* screen). Pressing the Home button sends your application an `appStopEvent` and your application should respect the request to stop. If your game cannot automatically save the full current game state when quitting, then it's permissible to display an "Are you sure?" dialog first. See "Game Flow" later in this guide for more information.
- A game must pause when the device is powered-off during game-play.

- Game controls are split into two functional areas: Game Menus and Game Play.

- Control within the menus must be consistent across all games and follow these guidelines:

Action	**Result**
Tapping the screen or pressing the Function button as appropriate for your game.	Brings up the game menu.
Manipulating the analog controller or tapping the screen	Navigates through the game menu.
Pressing the analog controller when it is centered or tapping the screen	Selects the highlighted item in the game menu.

- Control within the game should be consistent across all games when possible. Offer the user the ability to either fully customize controls or select between preset control templates. Tapwave suggests mappings for several common game-play types; including: Driving, Sports (Offense), Sports (Defense) and First Person Shooter. Not all games will map nicely to the templates listed below, therefore developers should feel free to modify these if a specific game requires something different.

Driving Games

Analog Controller	Steer
Blue (up) Action Button	Turbo
Yellow (right) Action Button	Switch camera
Green (down) Action Button	Brake
Red (left) Action Button	Accelerate
Left Trigger	Select weapon (Offensive or Defensive)
Right Trigger	Fire weapon

Sports Games (Offensive)

Analog Controller	Move / Turn
Analog Controller (center)	Jump / Dive
Blue (up) Action Button	Icon Pass
Yellow (right) Action Button	Pivot
Green (down) Action Button	Pass
Red (left) Action Button	Shoot / Kick / Swing
Left Trigger	Guard
Right Trigger	Turbo
Function Button	Call a play

Sports Games (Defensive)

Analog Controller	Move / Turn
Analog Controller (center)	Jump / Dive
Blue (up) Action Button	Icon Switch
Yellow (right) Action Button	Steal / Intercept
Green (down) Action Button	Switch player
Red (left) Action Button	Block / Rebound / Catch

First Person Shooter

Analog Controller	Look
Analog Controller (center)	Crouch
Blue (up) Action Button	Move forward
Yellow (right) Action Button	Move right
Green (down) Action Button	Move backwards
Red (left) Action Button	Move left
Left Trigger	Shoot
Right Trigger	Select
Function Button	Jump

5.Game Flow

- A game should support the following loading sequence when launched:
    - Display a splash screen then display a progress indicator if loading the game takes longer than 2 seconds.
    - Display the main game menu.
- The portability and instant-on nature of Tapwave devices leads to different patterns of use than with PC or console games. Instead of sitting down for relatively long gaming sessions, portable devices are often used in short bursts for a few minutes at a time. Tapwave games should support this model, and as much as possible allow the user to quickly suspend a game and switch to a different application, then later resume the game and keep playing. This suggests that games should load as quickly as possible, and unnecessary delays for introductory animations or splash screens should be removed or limited to the very first time the application is launched. The user should not be forced to choose between finishing the current level of the game and switching to the address book to look up a phone number -- instead allow the user to pause, check the phone number, and return to the game.

- If you cannot save the state of your game then you must notify your user before quitting and ask if he really wants to quit. If he selects affirmative, simply quit the game. If he selects negative then the game should switch to “pause” mode so he can resume the game later when he is ready.

- We recommend using the following as a template for your confirmation form and that you set “No” as the defautl button:

- A game should provide the user with the option of quitting a game without saving the game state. (For instance, if he wants to begin a new game.)

6.Interacting with the OS/Hardware

- A game must call `EvtGetEvent` (or use an appropriate timeout value when calling TwAppRun) on a regular basis (e.g. once per frame, but never less than once per second) to ensure that the system operates normally (e.g.: responds to alarms, low power, ExgManager events, etc). Your game must pause game-play and work correctly under these conditions.
- A game should support rumble effects only when appropriate. Don't overuse the rumbler as it puts additional drain on the battery. A user preference to disable rumble effects should be offered.
- A game should support multi-player Bluetooth when possible. The game's interface should support options for hosting a Bluetooth game and for accepting an invitation to a hosted game.
- A game should support wide-screen landscape mode (450x320 when the status bar is displayed).
- A game should leave the status bar open as much as possible. The status bar provides a place for essential device information and navigation, and will be enhanced over time. Please allow the user to access it whenever it is reasonable to do so. For example, make sure the status bar is visible while the game is loading and from game menus. You can close the status bar if necessary during actual game play. When the user presses the pause button, however, make the status bar visible. (It can temporarily 'overlap' the game window if needed.)
- A game should manage data files according to the "Managing Game Data" section in the *High Performance Games* document.
- A game should implement DRM protection as defined in the *Digital Rights Management* document, including operating in demo mode when running on hardware to which the application is not locked.

7.Game Installation

- A game should support installing data files according to the "Installing Your Game" section in the High Performance Games document.
- A game's installation instructions must indicate which category the game will be placed into after installation. We recommend using the "Games" category.

7.References

PalmSource provides additional best practices documentation. Refer to the *Zen of Palm* and *Palm OS User Interface Guidelines*.