

IPodGames

From wikiPodLinux

*The title of this article should be **iPod Games**. The initial letter is capitalized due to technical limitations.*

Table of contents

- 1 Overview
- 2 The Goal
- 3 The IPG File
- 4 Data File Formats
 - 4.1 Manifest.plist
 - 4.2 Manifest.p7b
 - 4.3 iPod Executable: .bin
 - 4.4 .bin.SINF file
 - 4.5 iTunesMetaData
 - 4.6 iTunesArtwork
 - 4.7 Resources/LANGUAGE CODE/Description.xml
- 5 Image Formats
 - 5.1 IDs 640 and 4097
 - 5.2 ID 2057
 - 5.3 ID 2048
 - 5.4 Image Converters
 - 5.4.1 Image Converter (Windows, Linux and Mac)
 - 5.4.2 Image Converter for Mac
 - 5.4.3 GUI Image Converter for Windows
- 6 Audio Formats
 - 6.1 .blob files
- 7 Known game GUID codes
 - 7.1 iPod 5th and 5.5th Generation Games
 - 7.2 iPod 6th Generation "Classic" Games
 - 7.3 iPod Nano 3rd Generation Games
- 8 Directories Generated On The iPod Upon Installation
 - 8.1 Games_RO/GUID
 - 8.2 GameStats_RO/GUID
 - 8.3 GameData_RW/GUID
- 9 Notes about Modifying Game Savefiles
 - 9.1 Texas Hold 'Em
- 10 Notes about specific games
 - 10.1 iQuiz
 - 10.2 PAC-MAN
- 11 Contributors
- 12 Information Source Links

Overview

This page is meant to document any reverse engineering of the iPod game packages (.ipg files) that function with the Apple-written iPod OS. This page does not discuss iPodLinux games, or podzilla games.

Since the contents of this page are the result of reverse engineering, there might be mistakes, and incorrect assumptions. Take it all with a grain or two of salt.

Sorry about the layout of this page. It is highly in flux. --BleuLlama

The Goal

The ultimate goal here is to enable people to write their own games for the iPod, for AppleOS. There is a huge homebrew community anxiously awaiting to do this.

We are NOT trying to crack the DRM on existing games. We are NOT trying to illegally spread purchased/protected games. Please do not ask to trade games with people. If you need some source material to work with, head to the iTunes Store, and buy a game or two.

The IPG File

The .ipg file is actually just a .zip file in disguise. If you change the name to have a .zip extension, you can unzip it and you will see a directory structure similar to the following: (the following is excerpted from a listing of "PAC-MAN 1.0.ipg")

```
audio/siren1.wav
audio/siren2.wav
audio/die.wav
Executables/Pacman_1_1_2563976.bin
Executables/Pacman_1_1_2563976.bin.sinf
Manifest.plist
Manifest.plist.p7b
PM_Logo.raw.lcd5
Resources/en/Description.xml
Resources/en/pacman.jpg
Resources/de/Description.xml
Resources/de/pacman.jpg
Resources/no/Description.xml
Resources/no/pacman.jpg
iTunesArtwork
iTunesMetaData
tex_ig.tga
tex_menu1.tga
```

Many files have been omitted, but this is the basic structure of the Pac-Man IPG zip archive.

Some of these files seem to be required, and exist in all games currently available. This is discussed below.

It would seem that the "Resources" is primarily used in the iTunes interface. They determine the instructions page, and other information displayed to the user in iTunes. The Resources directory contains a bunch of directories with two-character language coded subdirectories. Each of these have instructional information for iTunes in the associated language.

The "audio" directory, as well as the ".tga" files are used by the game binary at runtime as the graphics for the game. In fact, if you look at the pac-man tex_ig.tga, it contains all of the graphics for the game itself -- sprites, backgrounds, animations etc. tex_menu1.tga contains menu selection graphics, font lettering, and similar things. Other games contain m4a unprotected AAC files or WAV files which are background music, and in-game music cues.

The selection and names of these files are dependant on the game, and thus, none of them are required for a "minimal" game.

Data File Formats

Manifest.plist

This file appears to be scanned for by the iPod in the various game folders. If this file is removed from the game's folder, the game it's associated with disappears from the games menu.

This seems to be a manifest that explains a lot of the meta information about the game.

- **BuildIdentifier** - perhaps a unique identifier for this specific version and release of the game?
- **Files** - an Array of file data
 - **Digest** - a checksum of some kind (NOT: MD2, MD4, MD5, MDC2, RMD160, SHA, SHA1)
 - **Path** - the relative path to the file in the IPG heirarchy
 - **Size** - file size in bytes
 - **DRM** - contains a self-closing true or false tag, this denotes that the file is DRM protected. (only used for the .bin file)
 - **Verify** - verify the DRM maybe? (only used for the .bin file)
- **GUID** - the ID for the game. This is used to determine where the save/load/state data is stored. (see below)
- **name** - the name of the game
- **Version** - release version number
- **Platforms** - Information about the launchable binary
 - **BuildID** - some sort of unique identifier for this version and release of the game?
 - **ExecutablePath** - the path to the .bin file
 - **LaunchingArtwork** - the iPod-safe file. usually a .lcl5 file. (see ID 640 below)
 - This is displayed when the user selects the game on the iPod. It is the splash screen for the game, as it loads.
 - **PlatformID** - unknown, probably which iPod it runs on. "1" = 5th gen?
 - **PlatformVersion** - unknown, probably which revision of hardware it runs on. "1" = 5th gen?
 - **Size** - file size of the Executable referenced above.

Manifest.p7b

This seems to be another secondary manifest that deals with the FairPlay DRM certificate. It contains a lot of references to "Apple Computer Certificate Authority" as well as "Apple FairPlay Certificate Authority". It also contains the following related URLs:

- Apple Certificate Authority Terms (<https://www.apple.com/certificateauthority/terms.html>)
- Apple Certificate Authority Root CRL (<https://www.apple.com/certificateauthority/root.crl>)
- Apple Certificate Authority Signers (<https://www.apple.com/certificateauthority/casigners.html>)

This file seems to be different for every game, even if they are owned by the same user.

iPod Executable: .bin

Based on various people's information and research, this is not a Java, Small, or other virtual-machine based binary. It is most likely pure ARM asm, interfacing with an API provided by the RetailOS. This is the API we hope to figure out, and code against for our own games in the future.

The game executables are DRM-protected. It's unknown if this shares any similarity to the DRM used for the rest of iTunes Store-purchased content, or if it would be able to create our own executables that did not use DRM (just as it's possible to play .m4a files instead of .m4p files).

See this listing of files:

```
757544 Sep 19 00:42 Zuma_1_1_2563298.bin
757703 Sep 19 00:42 Zuma_1_1_2563298.bin.gz
1032 Sep 19 00:07 Zuma_1_1_2563298.bin.sinf
```

As you can see, the compressed version of the binary is even larger than the uncompressed version, indicating it's either compressed or encrypted already. the .sinf file, as has been noted elsewhere, contains information about the purchaser of the file. ("Signature Info")

Maybe someone could pull the executable for Brickout out of the firmware of the iPod, and put it into a .bin file, and see if that could be loaded...

The filename itself is composed of a few underscore-separated fields. These fields are defined in the Manifest.plist file.

- Name - "Zuma"
- PlatformID - "1"
- HardwareID - "1"
- BuildID - "2563298"

It is assumed that this is suggested by the Apple team to keep everything consistent. I would bet that as long as the filenames match between this and the content in the **ExecutablePath** and File listing in the Manifest.plist file, it would find it fine.

.bin.SINF file

This file seems to be related to the DRM on the iPod. Not much is known about it yet.

Field headers seem to be 4 bytes for block size, then a 4 byte identifier for that block. Since the 'sizes' might vary, the offset listed below is based on the Pacman sinf file I have for myself; yours may vary.

field ID	value
sinf	sinf header (size indicates filesize)
frma	unknown
schm	scheme ID?
user	unknown
key	unknown
tran	unknown
name	iTunes Store username
priv	private key?
sign	CA signing?

It is unknown if this file is customized by iTunes when the file is downloaded like other iTunes purchases, or if it happens on the server-side.

This file contains the purchaser's iTunes Store user name.

When only the name is changed, when attempting to put it back into iTunes you get an error, this may be because I zipped it incorrectly.

iTunesMetaData

This is a binary file that contains meta data about the game; copyright holder, game title, the email address/iTunes Store ID of the person it is registered to, and some other information.

iTunesArtwork

This is a 320x240 JPEG file that gets displayed as an icon in the iTunes 7 user interface. This is what the user clicks on to display the descriptive text/graphic about the game.

Resources/LANGUAGE CODE/Description.xml

This will display information to the right of the selected game icon. This probably can contain textual information, but for the initial group of games released, it simply points to a large (760x???) JPEG image which gets displayed in the left pane. It also defines, in an xhtml-esque/plist style, the layout of the right pane of the iTunes interface.

Image Formats

There are several not common image formats included in iPod games.

The header of these files seems to be always in the same format.

image header (little-endian byte order)

offset	field	size	value
0	image width	4	image width in pixels
4	image height	4	image height in pixels
8	type id	4	image type id (will be used later in this document as a label for the image format)
12	RGB format	4	seems to be always "565L" (for "raw.lcd5" files) or 0 (for other files)

IDs 640 and 4097

These two formats are very similar. File extensions are "raw.lcd5" (for ID 640) and "ipd" (for ID 4097).



"raw.lcd5" files are actually splash screens for iPod games and "ipd" files are used for example in the "Vortex" game.

The pixels of these images are stored as RGB565 (2 bytes per pixel).


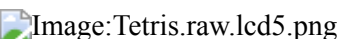
image data (big-endian byte order)

offset	field	size	value
16	RGB565 pixels	width*height*2	this is the whole image data in RGB565 format (2 bytes per pixel)

Here are some images converted from RGB565 "raw.lcd5" files and one image converted from a RGB565 "ipd" file:



 

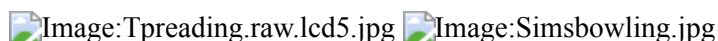
 

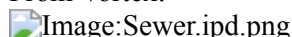
 



From Vortex:



ID 2057

This image format is used in "ipd" and "anm" files and stores the pixels in a different format than the other formats listed above.

It uses a RGBA color palette (4 bytes per pixel) and saves the pixels as indexes which point to a color stored in this palette.

palette data (big-endian byte order)

offset	field	size	value
16	RGBA palette	1024	this is the color palette which stores a maximum of 256 colors in RGBA format (4 bytes per pixel)

color index data (little-endian byte order)

offset	field	size	value
1040	color indexes	width*height	these are the color indexes (each 1 byte) which point to a color in the RGBA color palette

There are also images converted from this format available (both "anm" files):



ID 2048

This format contains an alpha mask and is also used in "ipd" files.

Every byte after the header represents the alpha value of one pixel.

alpha mask data

offset	field	size	value
16	alpha mask	width*height	these are the alpha bytes (1 byte per pixel) which form a mask

The following image is such a alpha mask from the game "Vortex":



Image Converters

Image Converter (Windows, Linux and Mac)

The commandline converter used to convert the above images can be obtained here:

- Download: iPodGame Image Converter (source + windows and linux binaries)
- Supports converting between the above four iPod game image formats, PNG and BMP.
- If an image is flipped upside down, flip it with a graphics program or use the --flip switch of the program.
- Integrated flipping function is implemented.
- There are still some endian problems on PPC Macs.

Image Converter for Mac

iPod Game Image to BMP for Mac OS X (Universal) (old version)

A CLI tool for converting three iPod game image formats to BMP.

GUI Image Converter for Windows

A GUI BMP --> LCD5 converter allowing new LCD5 images to be made can be found at:

LCD5 v1.1 BMP to LCD5 Converter (http://www.andrewcrook.co.uk/programs/LCD5_v1.1.exe)

This program uses the VB6 runtime files available from microsoft [1] (<http://support.microsoft.com/kb/290887/>)

Audio Formats

There are a few audio formats used in iPod games. WAV, AAC, m4a, and blob files. WAV and m4a files can be easily switched with another WAV or m4a file to change in-game sounds or music. WAV is used with sound effects, while m4a is usually used with music.

.blob files

.blob files can contain many different things. Some contain audio, others contain images. Audio .blob files can be played when imported as raw data in a program such as audacity. Sounds.blob within the Texas Hold 'em game appears to be a signed 16 bit PCM, little-endian, 16000 hz, stereo audio file.

Known game GUID codes

The **GUID** is defined in the manifest file for the game. The currently known GUIDs are as follows:

iPod 5th and 5.5th Generation Games

- **1C300** - Musika
- **11051** - SAT Prep 2008: Writing
- **11052** - SAT Prep 2008: Math
- **11002** - iQuiz 1.0
- **11050** - SAT Prep 2008: Reading
- **12345** - Vortex 1.0
- **12346** - Vortex Demo 1.0
- **14004** - Ms. Pac-Man 1.0

- **14005** - Ms. Pac-Man 1.0 Demo
- **1500C** - The Sims Bowling
- **1500E** - The Sims Pool
- **1B200** - Lost 1.0
- **33333** - Texas Hold 'Em
- **44444** - Zuma 1.0
- **50513** - Sudoku 1.0
- **50514** - Royal Solitaire
- **55555** - Bejeweled
- **66666** - Tetris 1.0
- **77777** - Mahjong
- **88888** - Mini Golf
- **99999** - Cubis 2 1.0
- **AAAAA** - Pac-Man 1.0

iPod 6th Generation "Classic" Games

None known yet.

iPod Nano 3rd Generation Games

- **11004** - iQuiz 1.1.0
- **11010** - Klondike 1.0.0
- **12347** - Vortex 1.1.0

Directories Generated On The iPod Upon Installation

When the game is copied over to the iPod, a few subdirectories are created within the =ipod_control= directory:

- Games_RO/GUID
- GameStats_WO/GUID
- GameData_RW/GUID

Once on the iPod, any game folder in Games_RO can be renamed and the executable will launch. Specifically, renaming any game folder except for Ms Pac-Man's(14004), Pac-Man's(AAAAA), and possibly Cubis 2's(Untested) to anything greater than a one word title will result in far slower and laggy gameplay and absence of music. Sound effects, however, still play.

Also, once saves are made, the name of the folder in Games_WO has exactly the same title as the folder of the game that created it. For example, if one renames one of the game folders in Games_RO to "just testing" (without the quotes), the saves that game will make will be in Games_WO/just testing.

Games_RO/GUID

This is where the contents of the .ipg zip file get copied into. The files are unmodified.

GameStats_RO/GUID

This is where stats for a particular game are stored. For Tetris, on my iPod, a subdirectory **en** contains the file **stats**.

There might be subfolders for audio, Executables and Resources. A lot of the game files in this directory can be replaced or edited to change the look of the game. For example, in Pac-Man, the splash screen PM_Logo.raw.lcd5, main game textures tex_ig.tga and iTunes game menu /Resources/pacman.jpg can all be edited and the binary will still execute.

GameData_RW/GUID

This is where save games are stored. When the user exits a game, and decides to 'save' their game, this data gets utilized. For Tetris, 'game.sav' and 'prefs.sav' are created. For the Vortex Demo, 'options', 'quicka', and 'stats' are created. For Texas Hold 'Em, a folder named 'en.lproj' is created, and inside is a file named 'data.txt'. For Zuma, 'Prefs' is created. For Ms. Pac-Man, a folder named save is created, and two files, 'suspend.dat' and 'ms_pac_man.dat' are placed inside. For Sudoku and Royal Solitaire, a file named 'savefile.dat' is created. Mini Golf creates 'jdmgp.sav'(Many files inside of the Mini Golf game folder begin with jdmg). Pac-Man creates 'pac_man.dat' and 'suspend.dat'. Bejeweled creates 'Prefs'.

The name of the game save folder in GameData_RW is based on what the game folder is named in Games_RO. For example, if the name of a folder in Games_RO is renamed BBBB, the save files the game inside creates will be stored in GameData_RW/BBBB. The same applies for GameStats_WO.

Notes about Modifying Game Savefiles

Texas Hold 'Em

Inside of it's GameData_RW folder, there is a folder named en.lproj. Opening this will reveal 'data.txt'. Upon opening this file in wordpad, one can see a kind of player registry:

```
FILE EXISTS v1.0
1
0,1
100,0,10,10,6
KURT,1000000
KILLER,500000
GREG,100000
JANE,50000
TERRY,10000
ADAM,5000
WENDY,1000
MIKE,1
0
3,2,2,2,2,2,2,1,1,1,1
255,255,255,255,255,255,255,255,255,255,255
0,0,0,0,0,0,0,0,0,0,0
0
0,0,0,0,0
0,0,0,0,0
0,0
```

The player 'MIKE' was created by the owner of the game. More users can be created by copying the entire segment after 'MIKE' and pasting after '0,0'. Changing the name changes the name of the account.

The ,1 after 'MIKE' appears to denote whether or not the player has an active game.

Notes about specific games

iQuiz

iQuiz allows for creation and playing of custom quizzes. Apple suggests that, in order to install a new quiz, one should: "Put the trivia pack folder and its contents in the iQuiz folder--it's in the iPod Games folder, which is in your iTunes folder." This method requires iTunes to synchronize with the iPod, thus copying the quizzes. There is another way to install quizzes.

Plug in your iPod, and browse to /iPod_Control/Games_RO/11002/UserTrivia/. If you've already put quizzes on your iPod through synchronization, there'll be a folder named "Packs" inside of UserTrivia. If not, then this folder may not exist. To put custom quizzes on it, create the "Packs" folder, and drag folders containing the new quiz's 'trivia.txt' into it. Once you do this, eject your iPod, your quiz should show up in iQuiz.

iQuiz is interesting because it provides a scripting language for the creation of quizzes. Here is an example of a quiz:

```

TITLE
iPodLinux Quiz

MENU TITLE COLOR
207, 148, 255

HIDDEN
NO

ASK
10

LOSE
3

WON MESSAGE
You won!

LOST MESSAGE
You lost!

MC
Which iPod does iPodLinux not support?
1st generation
2nd generation
3rd generation
shuffle
4

```

Scripting Terms

Terms	What it does
TITLE	On the line under this is what the quiz shows up as in the iQuiz quiz selection menu
MENU TITLE COLOR	In RGB, defines the color of the text that denotes the name of the information panes once the quiz has been won or lost.
QUESTION COLOR	In RGB, defines the color of the question text.
ANSWER COLOR	In RGB, defines the color of the text that the answers appear in.
SCORE COLOR	In RGB, defines the color of the text that the score appears in.
COUNT COLOR	In RGB, defines the color of the text that the question count appears in.
MENU BUTTON COLOR	In RGB, defines the color of the text that the button text in the postquiz menu appears in.
MENU TIME COLOR	In RGB, defines the color of the text that the time appears in.
HIDDEN	If the line under this is set to YES, the quiz won't show up. If NO, the quiz will.
ASK	The amount of questions to be asked.
LOSE	The amount of questions the player can miss before the game ends.
WON MESSAGE	The message to be displayed when the player wins.
LOSE MESSAGE	The message to be displayed when the player loses.
MC or TF	This means that the next question will be Multiple Choice or True and False, respectively.

Possible question types:

MC

```
MC
Which iPod does iPodLinux not support?
1st generation
2nd generation
3rd generation
shuffle
4
```

The MC tag has lines under it. The first line under MC contains the question to be asked. The next four lines contain the possible answers. The number after the last possible answer is the correct answer.

TF

```
TF
Apple was founded by Steve Wozniak and Bill Gates.
Actually, Apple was founded by Steve Wozniak and Steve Jobs.
FALSE
```

The line under TF is the question. The line under the question is the explanation of why the statement is false, and is displayed after the user answers. Under this is the answer.

If this was a true statement, then there would be no explanation line under the question, and the FALSE term would be replaced by TRUE.

PAC-MAN

After examining the ghost behavior, imagery, and various other things, it has been determined that this is not emulation at all, but rather a re-implementation of the game. Various bugs and eggs in the original are not found in this game, nor are some game behaviours. This is also evident in the promotional materials for the game; "a faithful rendition for your iPod". If you enjoy the original arcade classic, this is a nice alternative that you can carry in your pocket. If you **really** want 100% accuracy, then run ipodmame on your iPod instead.

Contributors

- BleuLlama 11:15, 14 Sep 2006 (CDT)
- Fxb 02:15, 10 Apr 2007 (GMT+1)
- Crook 12:54, 21 Sep 2006 (GMT)
- Etho 19:33, 26 Dec 2006 (GMT)

Information Source Links

These are websites where we've culled data from.

- {FPP} (<http://managepod.com>)
- Ben Sinclair's initial dissection (<http://www.bensinclair.com/article/whats-inside-an-ipod-game>)
- Dan Dicinson's thorough dissection (http://vjarmy.com/archives/2006/09/dissecting_ipod_games.php)
- DRM file Discussion at ipodwizard.net (<http://www.ipodwizard.net/showthread.php?t=15008>)

Retrieved from "http://ipodlinux.org/IPodGames"

Categories: Development

- This page was last modified 13:47, 14 Oct 2017.
- Content is available under GNU Free Documentation License 1.2.