Table of Contents

About CatHide

Introduction

Welcome to CatHide! CatHide is a tool for creating cross-platform mobile games. You'll be able to write all of your code, and build for all of the supported target platforms, right from inside CatHide. No more messing around with Xcode, various flavors of Eclipse, text editors, command line tools, or any of the other technologies that you previously had to mix and match in order to get your product on multiple platforms.

The manual will tell you every thing you need to know in order to use CatHide as efficiently as possible. In addition to using this manual, we encourage you to check our website http://www.CatHide.com frequently and follow along with any new tutorials that might be posted. Those tutorials will give you a code level look at creating games, unlike this manual, which will focus on using the CatHide software itself.

Cocos2d-x

CatHide is based on the popular cross-platform library, Cocos2d-x. We made a few changes to the library to aid in making CatHide's goal of one-click publishing to various platforms easier. We will run down some of those differences in this section so you'll know how CatHide might effect your existing knowledge of Cocos2d-x.

Sounds Effects and Music

Various platforms targeted by Cocos2d-x have varying support for different audio formats. We've made sure that all platforms that CatHide supports have support for mp3 audio for music, and wav audio for sound effects. You should use these formats when creating your game to insure its compatibility with any platforms that CatHide may add support for in the future.

Retina (or HD) mode

One more change that we've made to Cocos2d-x is to allow so-called “Retina mode” on all available platforms, instead of just iOS. When the first Retina display became available on iOS devices, the resolution was twice as much as previous models, up to 640x960 from 320x480. In order to support older, non-retina devices, developers create graphics assets in both sizes.

In addition, development switched from working with pixels to working with points. There are 320x480 points on an iPhone, regardless of whether its display has 320x480 pixels, or 640x960 pixels. This allows for ease of development across the product line without having to recalculate graphics for each resolution.

CatHide, in order to make one code base work in all platforms, works in the same 320x480 space that iPhones do. Well, sort of. We'll get into that further in the next section. Other platforms have moved away from the 320x480 graphics size as well, but Cocos2d-x's default behavior is simply to stretch the graphics to fit. We find that this is not as appealing as it could be. So, if you enable Retina mode on a CatHide app, it will use the doubled resolution images anytime the display size is greater than 1.5 times 320x480. On devices with resolutions such as 480x800, this results in a slightly scaled down graphic. But in our tests we firmly believe that a scaled down graphic looks much more appealing than one that is scaled up.

We've also added what we call “Super Retina' mode, which will allow retina iPads to use a further increased graphics asset resolution. To keep the aspect ratio of our “magic number”, those should be 1440 x 2280.

Every Cocos2d-x scene that is created automatically has the code to enable retina and super retina display already added. The option, however, is turned off. If you want to add support for retina or super retina mode, simply change the argument to the appropriate function from 'false' to 'true'.

Image Bleed

Standard Cocos2d-x relies on letter boxing to handle different aspect ratios. We've modified the version used in CatHide to use image bleed instead. What this means is that, for static backgrounds, you create an image that is 360x570 for standard definition images, 720x1140 for high definition images, and 1440x2280 for super high resolution images. The 320x480 (or 640x960, 1280x1920) pixel area in the center of that image is guaranteed to be visible on all currently supported devices. The space outside of that zone will be displayed as needed on different aspect ratios. Because of this, nothing critical to the function of your game should be put in those areas. The first tutorial on our website, about creating a Pong clone, gives a much more thorough explanation of how to use this “magic image size” to insure that your game looks great on any aspect ratio currently used in the mobile market.

Platform Specifics

The section will discuss aspects of CatHide that pertain to each of the specific platforms. It will lay out any requirements that there might be and any special instructions that might apply to one platform or another.

One thing that applies to all platforms, is that you should test your app on an actual device running that platform, or at the very least, one of the platform's official simulators. Although the goal of CatHide is to make all apps created with it behave the same across all supported platforms, there may be some differences that errors you have made, or things that we have missed, have caused. For example, some platforms do better at loading images if you pass the wrong format than others do. Some platforms also require special permissions be requested before performing certain tasks.

CatHideSim

CatHide requires Xcode to build and run your apps on the CatHide Simulator. In addition to Xcode, you must have downloaded the Xcode command line tools. In order to do that on Xcode 4.3 or greater, open up the Xcode preferences and navigate to the 'Downloads' tab. From there you will see the option to install the command line tools. Previous versions of Xcode install the command line tools in the expected place to begin with so no action is necessary with those versions.

The CatHide Simulator supports many different resolutions. If one of the simulators will not fit on your screen at its native resolution, it will be scaled to fit so you will still be able to view the entire device. This will allow you better visibility of your app while still allowing you to view its appearance at the appropriate aspect ratio.

Android

There are a number of issues that need to be addressed with Android. The first is that apps built with

CatHide will only run on devices which have Android 2.2 or better. You will also need to use NDK R7 or newer in order to build your apps.

When creating your app, you can use preprocessor defines to create different behaviors for regular Android devices, Kindle Fires, and Nook devices. Use *CH_ANDROID* for standard Android devices, *CH_NOOK* for Nook devices, and *CH_FIRE* for Kindle Fire devices. However, if you run the app on the Android emulator, it will compiled as a regular Android app. This is the case even if you are running on the Nook emulator. A future version of CatHide may remove this limitation.

By default, Cocos2d-x doesn't do anything when you press the Back button on an Android device, so you'll need to implement this functionality yourself. Although, the first scene of every CatHide Android based project is coded by default to exit the app when the user presses the back button. It is important to note that Nook devices handle the back button differently. Every scene created by CatHide's templates is already pre-coded to handle Nook's back button need correctly. Use should use the preprocessor defines listed above to make sure that your back button code does not interfere with the code for Nook devices.

Also on the subject of the Back button, it appears that Android emulators running versions of Android earlier than 2.3 do not respond to Back button presses. We've tested it on actual devices running those earlier versions and it works fine. This is another reason why you should try to test your apps on actual devices whenever possible.

Also, it seems that, while HD images will be loaded by the emulator, the quality of the graphics still look poor. This can give the illusion that the wrong images were loaded. The image quality is not degraded on an actual device however.

If you are using the Cocosdension audio library, you will need to preload sound effects before playing them the first time on Android. If the sounds are not preloaded, then they will only play the second time that they are requested to.

iOS

Here we'll discuss some of the distinctions that need to be made about developing for iOS with CatHide. First, make sure that you have Xcode and the iPhone SDKs installed, as they are required for building iOS apps. CatHide does not support Armv6 processors since Apple dropped support for them in Xcode 4.5. Make sure that you have developer certificates installed and provisioning profiles for each app that you create, as those are required for code signing. More information about code signing is available on the Apple developer website.

Next we'll talk about “Super Retina” mode. iOS is currently the only platform that it is supported on. If we are developing an iPad app, we want the 'normal' sized graphics (360x570 for fullscreen) to scale up with proper image bleed if no other assets are provided. If 'large' assets (720x1140) are provided, you can turn on retina mode and benefit from those higher resolution assets. With the iPad now supporting a retina mode of its own, we've added the “Super Retina” mode to indicate that you have provided assets in the 'xlarge' folder to be used with those devices.

It is important to note that the extra large graphics will only be included in your app if it is built for iPad only. This is to keep file sizes down for iPhone and iPod users. You can the target for building in the iOS tab of the project settings dialog. If you set iPhone and iPad individually, instead of selection Universal, then two binaries will be built. One for iPhone without the XLarge images, and one for iPad with them.

One other thing that is worth noting is that the iPad simulator can run pretty slowly when simulating

CatHide apps. You can increase the speed a little by temporarily turning of retina and super retina mode. But it is best to test on an actual device if you have one handy.

Playbook

CatHide requires version 2.0 or greater of the Blackberry NDK to compile your apps successfully. At the current moment, BB10 devices aren't supported, but support is planned for them in a future update.

Also keep in mind that the Blackberry Playbook simulator takes a bit longer than some of the other simulators to receive the upload and start the app. If it appears as though nothing is working, give it a little more time to let the process complete.

WebOS

There are a couple of things to note about WebOS support. First, CatHide cannot build for Pixi devices. Those devices use a slower processor, and while Cocos2d-x may work with them, we've decided to leave out support to avoid shipping extra versions of the library.

To build for WebOS, you will need to have both the WebOS PDK and the WebOS SDK installed on your computer. You can obtain these from the WebOS developer site.

CatHide has not been tested on a HP Veer. The resolution isn't one that the CatHide simulator currently supports. While there is no reason why the software shouldn't create apps that work fine on the Veer, please keep in mind that the aspect ratio of the device is non-standard and any software built on it will have to be extensively tested on an actual device.

At the current time, CatHide is using an older version of Cocos2d-x for WebOS support. The only major feature missing at this time is notifications. In the future we will update the WebOS port of Cocos2d-x to be in sync with the other platforms.

Handling Graphics

There are a lot of graphics that go into creating an app. Not only the graphics of the app itself, but the icons and the splash screen. This section will cover how CatHide handles those assets.

Icons

There are two different ways you can handle the creation of icons in CatHide. The first way is the most simple, but take a little control away from you regarding the quality of the icons. That is to simply let CatHide handle the scaling for you. This will save you from having to create many different sizes for the various platforms, and the various differences within each platform.

To let CatHide handle the graphics for you, you can simply replace the default icon that is added when you create a new project, with your own 512x512 pixel icon. CatHide will automatically scale this icon to the sizes required by all of the supported platforms.

If you'd like more control over your icon, you'll need to create files of the appropriate size and name them so that CatHide knows which icon goes with which platform. A list of icons by platform is given below, with the size that is required. Keep in mind that you should still create the 512x512 icon and replace the default icon. If you do not, and CatHide cannot find the icon that it needs, it will scale the default down. This could result in your app having the CatHide icon instead of yours. Also keep in mind that the filenames given below are case sensitive, so be sure to capitalize the I in icon.

Android Icons and Sizes

96x96 Icon-xhdpi.png

72x72 Icon-hdpi.png

48x48 Icon-mdpi.png

36x36 Icon-ldpi.png

iOS Icons and Sizes

57x57 Icon.png

72x72 Icon-72.png

114x114 Icon@2x.png

144x144 Icon-72@2x.png

Playbook Icons and Sizes

86x86 Icon-86.png

WebOS Icons and Sizes

64x64 Icon-64.png

After you create the icon files above, you will need to import them into your project. Follow the instruction in the section of this manual on Importing Images to do so. Make sure that you import them into the 'Icons' folder. Alternatively, you can drag and drop the files into the 'Icons' folder of the appropriate project in Project Explorer.

Splash Screens

By default, CatHide creates a splash screen for your game. This is separate from the splash screen created by iOS with the default.png file. You can disable this functionality in the project settings dialog, however, you cannot delete the splashscreen.cpp, and splashscreen.h files or you will get an error on building. Please note that the trial version does not allow disabling of the splash screen.

CatHide looks for the splash screen in an image named Splashscreen.png. You'll need to replace the default image with your own if you want to use a custom splash screen. Please note that the trial version does not allow changing the splash screen.

By default, the splash screen is only set to standard definition. CatHide only provides a default splash screen for standard definition. If you are supplying your own splash screen and want to provide a high resolution version, you'll need to enable retina mode in SplashScreen.cpp. To do this, open SplashScreen.cpp and find the line that says enableRetinaDisplay(false); and set it to true instead. In addition to enabling retina mode, you'll need to insure that there is a Splashscreen.png image in the Large folder of your projects graphics files. For more information on graphics folders, see the section below.

App Graphics

There are three subfolders in the Graphics folder of a CatHide project, Normal, Large, and XLarge. The normal folder contains graphics that are designed for devices that are of lower resolution, such as the first iPhones, Android devices, and WebOS phones. The large folder is for newer, high resolution graphics assets, such as those that would be used by the Retina display on more recent iPhones. The last, XLarge, is for graphics that are to be used by super high resolution devices, such as the Retina iPad.

You should always have graphics in the Normal folder, and add graphics to the other two if you wish to support higher resolution devices. In order to support those devices, enable retina display in your code.

Do not add any extensions to the graphics in the Large and XLarge folders. CatHide will do that for you automatically. The images should be named identically to their counterpart in the Normal folder.

Please see the section on retina graphics above for more information on which sizes to support and how to enable retina or super retina mode.

Registering CatHide

The trial version of CatHide will work for 30 days before you have to register. The only limitation in that 30 day period is that the “Made With CatHide” splash screen will always be used in place of your own splash screen and you cannot disable the splash screen. Once you have registered CatHide, you will receive a registration number via email. You will need an active internet connection to authorize the software.

Authorizing CatHide

A limited number of computers is allowed to use the same registration code. To keep track of how many copies of the application are currently using the code, it must be registered with our online server. To register your copy, simply open the Help Menu and click on Authorize. From here, you will be asked to enter your registration number. Once you do, and click the Ok button, CatHide will connect to the server and make sure that your registration number is valid and has not went over the install count limit. In order to keep your install count accurate, it is important to deauthorize the software if you no longer need it on a computer.

If your copy of CatHide is registered, you can see your registration number at any time in the About dialog, accessible from the CatHide Menu.

Deauthorizing CatHide

The procedure to deauthorize a copy of CatHide is the nearly identical to the procedure to authorize it. Once a copy of CatHide has been authorized, the Authorize option in the Help Menu changes to Deauthorize. To deauthorize a copy of the software, simply click on that menu item and then click Ok in the dialog that pops up. Assuming you have a valid network connection, the software will connect to our servers and deauthorize the software.

If you deauthorize the software, any time remaining in your trial period will be gone. But, you will be free to authorize a copy of the software on a different computer without having to worry about going over your authorization limit.

CatHide Menus

File Menu

The File Menu is where you create and work with new CatHide projects and project files. This section of the manual will go over each of the file menu options and all of the dialogs contained therein.

New Project

This is the first option that you will probably use when starting out with CatHide. It creates a new CatHide project. When a project is created, a new Cocos2d-x scene is created with the project name to serve as your first scene. A splash screen scene is also created. We'll talk more about the splash screen in a later section. When you click on this menu item, you are presented with a dialog that has a number of options. We'll go through the options one by one.

Project Name

This is the name of the project file. It should not contain any whitespace. Do not worry, though, there are separate settings for the display name, where you can use whitespace as appropriate to your project's name. We will discuss those in a later section.

Bundle Identifier

This is the bundle identifier that will be used for your project. Most mobile app stores use this to uniquely identify your app. It should follow the reverse domain name structure, e.g. com.yourcompany.yourapp. There is a default value already plugged in to remind you of the structure that should be used. Always replace this value with one unique to your company and app.

Create In

Here is where you specify the directory that you want to create the new app in. CatHide will create a new subdirectory within the given directory to hold the new project. The directory will have the same name as the project name given in the first field.

Select an Orientation

Here is where you tell CatHide whether your app is primarily used in landscape or portrait mode. One of these options must be selected.

Select a Physics Engine

Cocos2d-x, and therefore CatHide, comes with support for two physics engines built in. Those are Box2D and Chipmunk. If your game will be making use of one of these engines, select it here so CatHide will know to include it at compile time. You will still need to add in the appropriate header files to any source files that need to use the engine.

CocosDenshion

Click the checkbox in this section to include support for Cocos2d-x's audio engine, CocosDension. At this time, CatHide only supports the very basic features in SimpleAudioEngine. This is because not all Cocos2d-x platforms support the full feature set at this time.

New File

This menu item is used for creating new files or classes within a project. You have the option of creating generic c++ files, Cocos2d-x classes, or plain text files. Source files will be created in the Source folder of the project. Text files will be created in the Assets folder of the project. The options presented in the New File dialog are explained in detail below.

File Name

The first option you see in the dialog log will prompt you to enter name for the file. CatHide will automatically append the appropriate suffix to the file name unless you are creating a plain text file. If you are creating a plain text file, you'll need to provide your own extension.

File Type

This is where you select the type of file(s) that you would like to create. Unless you are creating a plain text file, you will most likely want to create a full class. When you do this, CatHide will create both the source file and the header file for the class. A generic C++ source file will be completely empty. If you create a Cocos2d-x class, then a basic scene will also be created. You can also choose to create source files and header files separately, although this feature will be less commonly useful.

Select the Project

The drop down box is used to select which of the open projects that you would like to add the file(s) to. By default, whichever project is currently active will be selected. The currently active project is the one highlighted in the Project Explorer. More information on the Project Explorer is given in a later section.

Open a File or Project

This menu item pulls up a standard file open dialog. If a CatHide project file is selected, that project will be loaded into the Project Explorer. Any other source or text file will be opened in the Editor. Please note that files opened in this way are not added to any open projects. It is most often easier to open files that belong to an already open project by using the Project Explorer. More details on the Project Explorer are in a later section.

Recent Files and Projects

These two menu items store the most recently opened files and projects, respectively. You can use these submenus to quickly open a file without having to use the open file dialog.

Import Files

This is the menu option to use when you want to import an existing file into a CatHide project. Please note that there is a separate menu item for importing images. If you import a file that is not already in the appropriate subdirectory of your project, it will be copied to the correct directory.

Select Files

Use this section to select the files that you want to add to the project. Clicking the 'Add' button will bring up a standard file open dialog that you can use to select the file to add. If you accidentally add a file that you did not intend to, or change your mind about adding a file, you can highlight it and press the 'Remove' button.

Select Project

This drop down box will allow you to select which of the open projects you would like to import the file to. By default the currently active project is selected. The currently active project is the one that is highlighted in the Project Explorer. We will cover the Project Explorer in more depth in a later section.

Import Images

This menu option allows you to import images into your project. It is very simliar to the Import Files menu option listed above. You should read those instructions for details on using this dialog. There is one additional drop down box in the Import Images dialog that does not exist in Import Files however. That is the one in which you select the folder to import the files into. The options are Normal, Large, X-Large, and Icon. More information about these folders is available in the Project Explorer documentation in a later section.

Close Projects

There are two options for closing projects in the File Menu. One for closing the current project, and another for closing all projects. The current project will be listed in the menu item so you can be sure which one will be closed. Closing a project removes it from the Project Explorer so that you can no longer work with it within CatHide. If a project is not open, then any changes to the Notepad data for any files associated with it will not get saved. More information about the Notepad is available in a later section.

Save Files

There are 5 entries related to saved files in the File menu. They should all be familiar to you as they are fairly standard across programs. We'll go over them here though, just in case.

Save

This is a simple save function. It will save the work you have done so far on a file using the file's current filename.

Save As

This will allow you to save the changes to the current file using a different file name. After the file has been saved to a new filename using this option, the save function will use the new filename.

Save a Copy

This option will save the changes to the current file using a different file name. Unlike the Save As option, however, the file will keep its original name and calling Save on any future changes to the file will modify its original file, not the copy that was created.

Save All

This save all of the open files using their current filenames.

Revert To Saved

This will reload the current file from disk. Any changes that have been made since the file was last saved to disk will be lost.

Close Files

There are three options for closing files in the File Menu. One for closing the current file, one for closing all files, and another for closing all of the files except the current file. The current file will be listed in the menu item where applicable so you can be sure which file will be closed.

Edit Menu

Most of the Edit Menu in CatHide is fairly standard. We won't waste your time talking about what functions such as undo, redo, cut, copy, and paste do. We are pretty sure that you are already familiar with those functions. There are a few non-standard items in the Edit Menu, however. We will take a look at some of them now.

Copy To / Paste From

Many times when you are coding you may want to copy several non-contiguous blocks of code from one location to another. CatHide makes this easier by giving you five additional clipboards that you can copy to and paste from. You can use the submenus in Copy To and Paste From to select which clipboard slot to perform the operation in. Alternatively, to save time you can use the keyboard shortcuts or the Editor's context menu. We'll discuss the Editor and its context menu in a later section.

(Un)comment Selection

Sometimes it is nice to be able to quickly comment or uncomment a block of code. With CatHide, you can just select the block of text with your mouse and then choose this option from the Edit menu. It is also available in the context menu for the Editor by right clicking on your document.

Go To Line

This menu item will pop up a little dialog that will allow you to enter a line number. Upon accepting the line number that you have entered, the editor will place the text cursor on that line in the current document. The document will be scrolled if necessary to put the chosen line in the viewport.

Go To Cursor

This menu item will scroll the document so that the location of the text cursor is in the viewport.

Deploy Menu

This menu is where you will find the option to deploy your app to the various app stores. After the deployment process is complete, a Finder window will pop-up that contains all of the binaries that you requested to be built.

Building an app from the deploy menu always creates a fresh build, so any problems that may be caused by not started with a clean build are not a problem. Of course, this means that compiling can take a little bit longer as each source file will be rebuilt every time. Since deployment is only meant to be used when you are finished with an app, compile time shouldn't be a major problem.

Deploy

When you select this option, you'll be given a list of the four major platforms (Android, iOS, WebOS, and Blackberry). Choose which of the platforms you wish to deploy to. The Project Settings dialog allows you to refine which sub-platforms you wish to target.

In order to deploy, you'll need to have all of the code signing information setup correctly in the Project Settings dialog. See the instructions for that dialog for more information. You'll be asked to provide a password for the Android keystore and the key alias that you have set to sign the app with. You'll also be asked to provide the password to your Blackberry signing key.

Project Settings

The Project Settings have been given their own section of the manual. Please refer to that section for help in setting up your project for the various platforms that are available.

SDK Settings

This is a shortcut to the SDK Settings tab in the main preferences dialog.

Run Menu

The Run Menu is CatHide is what you will use for testing your app. From this menu, you can run the app in CatHide's built-in simulator, CatHideSim, or from a variety of platform specific devices and emulators. The options in this menu are explained below.

Run in CatHideSim

This option presents a submenu with a variety of popular mobile resolutions within it. Selecting one of those resolutions will build and run your app in the CatHide Simulator with that resolution. More information about CatHideSim can be in a later section of the manual.

Debug

This option will open up a Terminal and start the GDB debugger. It will use the most recently launched

CatHide Simulator as the binary for debugging. If changes have been made to the app since it was last ran, it will be rebuilt.

Clean

This option will delete all of the temporary build files that are created when you build your projects to run in the various simulators and emulators. For most platforms these files are stored so that the entire project doesn't have to be rebuilt each time you want to make changes.

Run on Android

This option will bring up a submenu with three choices for running your app on a native Android platform. They are explained below.

Start Emulator

This option simply brings up the Android emulator launcher to allow you to launch an emulator. It doesn't know if there is already an instance of the launcher running, so clicking it a second time will start another instance, rather than bringing the current instance to the front.

Run on Emulator

If there is already an Android emulator running, this option will install and run your application on that emulator. At the present time, CatHide can only work with one emulator at a time. If there is more than one emulator running, it will launch the app in whichever one the Android Debug Bridge defaults to. The running emulator will not be brought to the front when the app is launched, so be sure to bring it to the front yourself.

Run on Device

If there is a device connected to the computer, and detectable by the Android Debug Bridge, this option will install and run your app on that device.

Run on iOS

This menu provides several options for testing your app on iOS.

Run on iPhone or iPad Device

Selecting these options will build your app using the development profile that is setup in the project settings. In order to build for the device, a valid singing id and provisioning profile must be given. For more information on obtaining and installing developer profiles, visit the official Apple developer site.

CatHide cannot actually launch your app on an iOS device. Instead, it will open the finished build in Finder. To run it on your device, simply drag and drop it into the 'Applications' section of iTunes and sync your iOS device. If the app is already installed on your target device, and you have not incremented the version number since the last time it was installed, you'll need to delete it from the device before iTunes will install the new build.

Run on iPhone or iPad Simulator

These two options both build the app for testing on a simulator and open the app on the simulator for the chosen device. They will be opened using the most current iOS version available to the simulator. It is important to note that the iOS Simulator will not come to the front when the app is launched if it is already open. So be sure to bring it to the front yourself after the app is successfully launched.

Run on Blackberry

This option will run your app on the Blackberry Playbook simulator, assuming that one is running. You will need to setup the IP address of the simulator in the Project Settings dialog, which is accessible at the bottom of the Run menu. Please note that the Playbook Simulator seems to respond much slower than the others, so it may take longer for your application to install and load on this platform.

Running on an actual Blackberry Playbook device is not supported at this time, but is planned for a future release.

Run on WebOS

This option will allow you to run your app on a WebOS device, provided that one is attached to your computer. At this time, running CatHide apps on the WebOS emulator is not supported, because CatHide relies on OpenGL, which is not supported on the WebOS emulator.

Project Settings

The Project Settings have been given their own section of the manual. Please refer to that section for help in setting up your project for the various platforms that are available.

SDK Settings

This is a shortcut to the SDK Settings tab in the main preferences dialog.

Preferences

The CatHide preferences dialog is available from the application menu. There are two separate tabs In the dialog. One is for the appearance of CatHide and the other is for SDK settings. Both of them are described in detail below.

Appearance Settings

These settings allow you to control how CatHide looks. Let's take a look at the various options that are available to you.

Syntax Highlighting

CatHide ships with five different color schemes that can be used within the editor. The drop down box in this section of the preferences will allow you to choose between them. To the left of the drop down is a small sample of what the color scheme looks like, and below the drop down a description of the scheme.

Outline

There is only one option currently available for the Outline. This option allows you to show the class name of the function in the outline. By default, Outline hides this to make things more clean and easy to read.

Console

By default, the Console will replace the full paths in the build output with a shortened version to make the output easier to read. The parts replaced will have $BUILDDIR in their place. If you would rather see the full filenames, you can check the box in this section.

SDK Settings

This is the section where you tell CatHide where it can find the various SDKs that are needed to build for all of the supported platforms. On first startup, CatHide will try to locate some of these by itself. Others you will need to fill in for it. If you do not have one or more of the SDKs, and do not plan on building for that platform, simply leave the relevant fields blank.

Toolbar

The CatHide toolbar provides quick access to many of the most commonly used menu items. In this section, we will take a look at each of the items, from left to right, and explain what they do.

New Project

This toolbar item features a document with the CatHide logo embedded in it, and a plus sign in the top right corner. It is used for creating new projects.

New File

This toolbar item, which looks like a text document with a plus sign in the upper right corner, will bring up the New File dialog when pressed.

Open File or Project

A green arrow pointing to the left, away from a document, makes up this toolbar item, which opens the Open File or Project dialog.

Save File

This toolbar item features an icon much the opposite of the open file item. It has a red arrow pointing to the right, toward the center of a document. Pressing this item will save the current document.

Save All

This item is simliar in appearance to the Save File toolbar button. It differs only in that it consists of a stack of three documents instead of a single document. Pressing this will save all of the file which are currently open in the Editor.

Revert To Saved

Pressing this toolbar item, which consists of a green arrow pointing down onto a document, will revert the currently active document to its saved version. Any changes made between the time the document was last saved, and the time this button is used will be lost.

Run

This toolbar action, which looks like the Play button on an audio interface, will run the currently active project using the most recently used CatHide Simulator. If changes have been made to the source since the last time the project was ran, it will build the project first.

Stop Building

This toolbar item, which looks like the stop button on an audio interface, can be used to stop any build process once it has been started. This toolbar item does not have an equivalent menu item.

Debug

Pressing this toolbar item, which looks like a command line prompt, will open up a terminal and start GDB so that you can debug the currently active project. Debugging only works for projects running in the CatHide Simulator. The most recently used CatHide Simulator will be the one that gets ran.

Stop CatHideSim

This stop-sign looking toolbar item will stop the CatHide Simulator if it is running. The simulator will immediately exit with no confirmation dialog.

Deploy

This toolbar item looks like a gear. It will bring up the deployment dialog, which will allow you to build your submission ready binaries of the current application. For more information on building for deployment, see the Deploy Menu documentation above.

Project Settings

The Project Settings dialog is accessible from both the Run menu and the Deploy menu. It is where you can adjust options for your project, as well as where you setup various platform specific options for the project.

Platform Independent Settings

At the top half of the Project Settings dialog are all of the settings that are platform independent. Changing these settings will affect all platforms. The settings are explained below.

Display Name

This is the name that will be displayed for your app when it is displayed on a device. Unlike the project name you gave when creating the project, this field is allowed to contain spaces. Remember that you must keep your display name short to insure that it fits in the limited space available under the icon on mobile devices. Twelve characters or less is a good goal.

Bundle Identifier

Almost every mobile store requires your app to have a bundle identifier using reverse domain formatting. This is the field that you will enter that identifier into. It should be already populated with the value that you gave when creating the project. You can change the identifier at any time, but be aware that if you have already submitted an app to the store, you might not be able to update that app with a different identifier.

Orientation

This is where you tell the program whether your app is designed to function in landscape or portrait mode. It is important to note that if you are not using the trial version, and have not replaced the default splash screen that CatHide creates for you, then the splash screen will not change if you change this option. The default CatHide splash screen will always be in the orientation that you selected when creating the project. Your custom splash screen, of course, will be in whichever orientation you created it in.

Physics Engine

This option tell CatHide which physics engine, if any, you will be using. You will still need to add the appropriate header files to any source files that use that physics engine. This option merely tells CatHide which library to link into the final build.

Checkboxes

There are several checkboxes in the section under Physics Engine. These allow you tell CatHide whether or not you will be using the Cocosdenshion audio library, and whether or not you want the splash screen to show. Please not that trial version users cannot disable the splash screen.

Android Settings

This section is where you will set the Android settings that are specific to your app, including information that CatHide needs to properly sign the app.

Version

This is the version of the app that you want customers to see. For example, version 1.1 or version 1.0.1.

Version Code

The version code an integer representing the build number of your app. Every time you submit an update to the Google Play store, this number should increment by one.

Targets

This set of checkboxes is where you tell CatHide which of the Android stores you would like it to build for. At the current time, there is no default difference between Android (Google Play), and Kindle Fire builds. For that reason, Kindle Fire is unselected by default and you can submit the regular Android build to Amazon. You can use preprocessor defines to make distinctions between standard Android builds and Kindle Fire builds. If you do so, you will need to enable Kindle Fire as a separate build in this area. More information on using preprocessor defines is available in the Platform Specifics section earlier in this manual.

Keystore and Key

These are both related to signing your app for distribution on the various markets. Every app must be signed with a key, which is stored in a keystore. If you have already created a keystore for signing apps, you can load it by clicking on the button labeld 'Load Keystore' underneath the Keystore section. If not, you will need to click the button labeled 'Create Keystore' to create a new keystore.

The same goes for the key itself. Each app should use a separate key from other apps, but you can reuse the keystores.

Remember, DO NOT LOSE OR DELETE the files that you create during this process. If you do you will no longer be able to update your apps on the markets. Be sure to make a backup of the keystore file and keep it in a safe place.

Manifest Permissions

This section is where you add any permissions that your app will need to ask of the Android devices that it runs on. Type the permissions into the text box exactly as you they would appear in the Android manifest. More information on Android permissions is available in the official Android developer documentation on the Android developer website.

iOS Settings

These settings are for building and deploying for iOS.

Version

This setting controls the version number that is shown to the user.

Targets

This will allow you to choose whether you want to build an iPhone application, and iPad application, or a universal application. You can choose to build both an iPhone and iPad application and two separate files will be created. If universal is selected then the other two will be automatically unchecked. Images in the XLarge folder will only be included on iPad only apps.

You can use preprocessor defines to make distinctions at compile time regarding which platforms are supported. More information on using preprocessor defines is available in the Platform Specfics section earlier in this manual.

Signing IDs

These are the fields where you put your development and distribution signing IDs that you want to sign your app with. You'll need to enter these fields exactly as they appear in your Keychain utility, including any numbers that appear after your ID. A development ID is required for testing an app on your own device, and a distribution ID is used for building the app for the app store or ad hoc distribution. More information on creating signing IDs can be found on the official Apple developer website.

iPad Bundle ID

iPad apps need their own bundle ID. So if you are creating an app that is meant to be deployed on iPad, enter that bundle ID here. Universal apps and iPhone only apps use the bundle ID from the platform independent section at the top of the dialog. If iPad is not one of the selected targets, this field will be grayed out.

Provisioning Profiles

These fields allow you to select the provisioning profiles that you would like to use for building your app for distribution or development. The profile on the left is for iPhones and universal binaries, the profile on the right is for iPad specific builds. If iPad is not one of the selected targets, these fields will be grayed out. CatHide only displays profile that are correctly installed on your system and that haven't expired. Please see the official Apple developer website for more information on creating, installing, and using provisioning profiles.

If you have installed a provisioning profile after opening the project settings for the first time in a CatHide session, you can click on the Refresh button to reload the data.

Playbook Settings

These settings are for building and deploying to the Blackberry Playbook.

Version

This is the version of the app that you want customers to see. For example, version 1.1 or version 1.0.1.

Version Code

The version code an integer representing the build number of your app. Every time you submit an update to Blackberry App World, this number should increment by one.

Signing ID

This is the name that you gave to RIM when requesting your code signing keys. More information about requesting and setting up code signing for Blackberry development is available on the official Blackberry developer portal. If this ID does not match the key provided by Blackberry, the final build will not be signed correctly.

Simulator IP

In order to deploy a development build to the Blackberry Playbook simulator, CatHide must know the IP address of the Simulator. To find the IP address, first launch the simulator, then click on the icon of a little person in the upper right corner. The IP address of the simulator will be displayed. While you are there, you can make sure that development mode is turned on.

Simulator Pass

This is the password that you set for your Blackberry Playbook simulator. If you have not set a password for the simulator, simply leave this field blank. The password will be stored in your setting, but it will be encrypted for security.

Device IP, Device Pass, Debug Token

These options all relate to testing an app on a physical Blackberry Playbook device. This is not supported in the initial release of CatHide, although it is a feature planned for a future update.

Permissions

This is where you set which permissions your app will need to request from the Playbook device. Enter the permission exactly as you you in your manifest file. More information on Blackberry Playbook permissions can be found on the official Blackberry developer portal.

Category

The category that you select here is not a direct relation to the category that your app will be in on App World. In fact, there are only two options here, games and media. If your app is one of the two, select it. Otherwise leave it marked Uncategorized.

WebOS Settings

The WebOS settings dialog is where you tell CatHide things that it needs to know in order to build your app for WebOS. The details are listed below.

Version

This is where you give the version number of your app. WebOS requires a three part version number, such as 1.0.0 when submitting.

Vendor

This is the vendor name. It should be either your own full name, or your company's name.

Memory Usage

WebOS requires that you provide an estimate of how much memory your app will use. Estimating usage relies on running the app on a device and then using a device shell to monitor the memory usage of the app over a period of time. The official documentation on the WebOS developer site gives full instructions on how to do this. Alternatively, and less accurately, you can run the app in a CatHide simulator that matches the resolution of your target device, and monitor the usage using the unix 'top' command, or a process management program like Activity Monitor.

CatHide Main Window

Project Explorer

The Project Explorer is the tree view on the left side of the main window. From the project explorer, you can see the projects that you have open and the files contained in that project. Highlighting a project, or the file of a project, in the Project Explorer will make it the currently active project for functions such as those in the Run menu.

If a file is highlighted, then the context menu (available by right clicking) will contain an option to remove the file. If this option is selected, a dialog will pop up to confirm that you wish to continue with the action. The dialog will also give you the option to delete the file from disk, as well as just removing it from the project. Please remember that the SplashScreen.cpp and SplashScreen.h files cannot be deleted. You can, however, disable the splash screen in the project settings unless you are using the trial version of CatHide. For more information on the splash screen, read the Graphics Considerations section earlier in this manual.

Every item in the Project Explorer has a context menu that can be accessed by right-clicking. Other than the remove file feature mentioned above, the only option available in context menus that isn't available in the menu bar is the option to display the file or folder in Finder. For information on the other context menu items, read the section on the menu bar above.

Color Coding

Files listed in the project explorer can be one of three different colors. A black filename is the normal state. A filename that is printed in red indicates that, for some reason, the file was not found on the disk. A purple file indicates that the file still has some items marks as todo that have not been completed. For more information on todo items, refer to the Notepad section below.

Drag and Drop

You can add files to your project simply by dragging and dropping them into the project folder under the project that you wish to add them to. Source, plain text, and mp3 or wav audio files can be dropped anywhere under the project and will automatically placed in the right location.

Graphics files must be dropped into the appropriate size subfolder (Normal, Large, Xlarge, or Icons). If you try to drop an image in the Graphics folder instead of one of its subfolders, you will be given an error message. For more information on graphics sizes, read the Graphics Considerations section earlier in this manual.

Editor

The Editor is where you will do all of the editing of your source files, and if you'd like, any text files that are associated with your project. At the current time, it does not have some of the features of more advanced IDEs, such as code completion or auto-formatting. It does however have syntax highlighting, as any good code editor should. The color scheme can be adjusted from the Preferences Dialog.

The Editor also features line numbering on the left hand side to make referencing the code much easier. The line number bar has a couple of other features that we think you'll find helpful. First, it will highlight any lines that have changed since the file was last saved in red, making it easier to track your changes. Second, it will highlight any files that have changed since the file was last opened, whether they have been saved or not, in green.

Open files will be accessible through tabs on the top of the editor. If you have more open files than will fit in the tab bar, then a scroller will appear in the upper right corner which will allow you to scroll through the open tabs. The tab of the currently active file will feature darker text and will appear slightly larger than the other tabs.

The CatHide title bar will also change according to which file is currently active in the editor. In the title bar, you will not only be able to see the name of the file that you are editing, but also the name of the project that it is associated with. This is convenient if you have two projects which have files of the same name. In particular, this could happen a lot if you are editing the splashscreen.cpp or splashscreen.h files, which every CatHide project has.

If the Editor has a file open a file from outside a CatHide project, and you then import that file into a CatHide project, the file is copied to the source directory of the project that it was imported to. When this happens, the file that is opened in the Editor will then point to the newly created copy. This allows you to continue editing your already open file and have any saved changed affect the file that the project will be reading from instead of the original file.

The Editor interacts nicely with some of the other features of CatHide. Functions created in the Editor will show up in the Outline. The outline can also make use of #pragma marks. Comments marked as 'todo' will show up in the todo tab of the Notepad. More information about how the Editor interacts with the Outline or the Notepad can be found in the section of the manual dedicated to those features.

Editor Context Menu

By right clicking on the editor, you will be able to bring up its context menu. The context menu gives you quick access to a few of the functions that are available in the menu bar within the Edit Menu. Please see the section on the Edit Menu for a detailed explanation of these functions.

Console

The Console is located at the bottom of the CatHide main window. It provides you with information about the build process. The Console is divided into three different tabs. Each tab provides a specific amount of data. All three tabs share the following color coding scheme:

Red text indicates an error.

Orange text indicates a warning.

Green text indicates a CatHide message.

White text is general output.

Compiler Output Tab

The compiler output tab is where all of the data that is output by the compiler during the build process will go. This section will also contain information from any other tools required to build and package the app.

App Output Tab

If you are building for CatHideSim, then this is where the output from your app will go. It is here that you will be able to read debugging messages from CCLog calls, or regular C/C++ output calls.

If you are not building for CatHideSim, then this area will simply tell you when your app is launched on the relevant emulator or device and when the build is successful.

Build Issues Tab

This tab will take focus if there is an error in the build process that prevents the process from completing. It will show only the errors and warnings from the compiler output tab to make it easier for you to see what went wrong. You can double click on any error that contains a file name and a line number to be taken to the source of the error in the Editor.

Remember, this is just an abbreviated version of the compiler output, so it is certainly possible, and sometimes maybe helpful, to switch to the compiler output tab in order to see the full output.

Outline

The Outline section of the CatHide screen appears in the upper right corner of the screen. Its function is to provide you with a listing of all of the functions the given project file, to make navigating the file easier. You can double click on any of the functions to have the cursor jump to that function. You can think of the Outline as sort of an index into your file.

By default, the Outline hides the class name in front of the function. This is because, for the most part, all functions in a file are from the same class. And at the very least, there aren't usually two functions with the same name belonging to two separate classes in the same file. Should you want to turn this behavior off, you can do so from the Preferences dialog in the Appearance Settings tab.

Pragma Marks

In order to make your Outline even more easy to navigate, it supports the use of #pragma marks. For example, before the group of functions in your source file that handle initialization of some data set, you could place, on a line by itself:

```
#pragma mark Initialization Functions
```

By doing this, the Outliner will create a new heading at the start of that section, labeled “Initialization Functions”. As files get larger, this can be a very convenient way to keep your outline easy to navigate and all of the functions in a file easy to find.

Notepad

The Notepad is the set of tabs in the lower right hand corner of the main window. The section of the interface allows you to take notes for the current project, the current file, or to keep track of a todo list relating to the current file. It is important to note that if the project that a given file belongs to is not open, then the data typed in this space will not be saved. If the project associated with the file cannot be found, you will be told within the editor itself.

Scratch Tab

The Scratch tab is where you put any notes that you want to make about the project as a whole. This section is for keeping notes on the project associated with the currently active file in the Editor, and not necessarily the currently active project. The name of the project that the Scratch tab will be associated with will be displayed in the header for this tab.

Notes Tab

This tab is for keeping track of notes that you would like to keep for the currently active file. Since the notes are saved in the project file, the project associated with the file must be open in order to take notes on it. This is for general notes that you might want to take. For code specific todo lists, you may want to use the Todo tab.

Todo Tab

This tab is different from the other tabs in that you don't write to it directly. Instead, if you place a comment in your source file consisting of a single line comment, followed by a space and the word 'todo', an entry will be created in here labeled with whatever comes after the 'todo'. For example, if your source contains a comment such as:

```
// todo Add more options here
```

Then you will see 'Add more options here' in the todo list of your Todo tab for that file. Double clicking on any of the todo items will take you to the line of code that contains the comment.

Any files that still have todo items remaining in them will be highlighted in purple in the Project Explorer for ease of tracking. More information on the Project Explorer is available in its section of the manual.

CatHideSim

CatHideSim is the name of CatHide's Simulator. It will allow you to test your apps without having to rely on the native simulators and emulators of any specific platform. It also allows you to test your app across multiple resolutions. This is especially beneficial because CatHideSim does not have to boot up the way that platform specific simulators and emulators do. So it is much quicker to switch resolutions using CatHideSim than any other method.

Input

Input on CatHideSim works similar to any other simulator out there. The mouse is used to simulate touch events and the keyboard is used to simulate the keyboard of a mobile device.

One thing CatHideSim has that other simulators do not have, is support for accelerometer simulation. You will need to have a joystick with an analog controller, but if you do you can use that to simulate accelerometer events. This makes testing your accelerometer driven games much easier and removes the need to test them on an actual device during the initial creation process. Of course, you should always test the final build of your apps on the all platforms that they are intended to be sold for.

Screenshots

CatHideSim has a feature built in that allows you to take screenshots right from within the simulator. Since most apps created with CatHide will look the same on all platforms, as long as the aspect ratio is the same, this is a great way to quickly take screenshots for your apps across multiple aspect ratios.

Access to the screenshot function can be had through the Tools menu of CaHideSim, or through the keyboard shortcut, Command-S. When you activate the screenshot function, the app will automatically pause. There is also a separate pause function (see below) that might help you get just the right shot.

You will be asked to select a location to save the screenshot and a filename to save it under. CatHideSim saves screenshots in PNG format. If the simulator is scaled to fit on your screen, it will zoom to full size in order to take the screenshot. You will know the screenshot has been captured when you hear the camera click.

Pause/Unpause

Getting the perfect screenshot is much easier if you can pause the app in progress. CatHideSim allows you to do just that. Simply press Command-P to pause the app, and press it again to unpause it. Or, if you'd prefer, you can access the command from the Tools menu.

Copyright Notices

The CatHide software is copyrighted by Imperial Penguin. Imperial Penguin is not associated, affiliated, endorsed, or sponsored by any of the mobile companies or mobile app stores mentioned in this manual or in the CatHide software. The Apple, Android, WebOS, Kindle, Nook, and Blackberry trademarks are property of their respective owners. Cocos2D is a registered trademark of Ricardo Quesada. Neither Imperial Penguin nor any of the products offered on this site are associated, affiliated, endorsed, or sponsored by Ricardo Quesada.

CatHide makes use of the Cocos2d-x library. Cocos2d-x is an open source library that Imperial Penguin has no legal claims to. Links to the source code of Cocos2d-x that includes the changes made for use in CatHide are available from the downloads section of our website, http://www.cathide.com.

The font used in CatHide is Deja Vu Sans Mono. It is available under a free license from the the official website at http://dejavu-fonts.org.

CatHide makes use of the free software iPhoneSim for running applications on the simulator. That software can be found on its Github repository at https://github.com/landonf/iphonesim