# Mobile Game System Code Manual

## Table Of Contents



## 1. Light Strobe

### ESRender

**Main OpenGL rendering object.**

**- (id)init**
*initialization opengl color frame buffer and timer.*

**-(void)render**
*setting OpenGL context and preparing for rendering - building viewport and clearing previous frame.*

***- (void)resetTime;***
*Reset time. When application going background or when rendering stopped.*

***- (void)bindRenderBuffer;***
*Binding OpenGL render buffer.*

***- (int) getElapsedTime;***
*Return elapsed time between two draw callbacks. need for time synchronization.*

***- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;***
*when screen size changed. needed at initialization stage*

***- (void)setupPerspective;***
*setup Perspective projection for any 3d stuff*

***- (void)setupOrtho;***
*setup orthogonal projection need for any 2d stuff*

***- (CGPoint)touchResizeForDevice:(CGPoint)touchLocation;***
*resize touch point for specific device. ViewPort have fixed size (3.2x4.8 units) and we no need to load any different files for ipad or iphone we just scaling this.*

***- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *) acceleration;***
*here is entry point for accelerometer callbacks. here you can send accelerometr info to any other objects*

***- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
***- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
***- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*) view;***
***- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
*here is entry point for any touches. kepp in mind we should resize touches before sned it to any objects.*

## MViewPort

**- (id)init**

*initialization point of all light strobe screens: MStrobeFire, MStrobeKolo, MStrobeRose, MStrobeBlueFire, MstrobeLightning.*

**- (void)initRecorder**
*initialization point of recording. we need this for level measurement*

**-(void)ledLightOn**
*enabling/disabling LED light*

**-(void)upgradeComplete**
*invoke only when upgrade IAP completed*

**-(void)upgradeAction**
*start upgrade action*

**-(void)render**
*main rendering method*

### StoreObserver

Main Object for processing In App Purchase
**- (void) failedTransaction: (SKPaymentTransaction *)transaction**
*when transaction is error*

**- (void) completeTransaction: (SKPaymentTransaction *)transaction**
*when transaction is complete*

### MStrobe

**-(void)render**
*method for rendering 2d stuff*

**-(void)renderPersp**
*method for rendering 3d stuff*

# 2. Road Racing

### ESRender

**Main OpenGL rendering object.**

**- (id)init**
*initialization opengl color frame buffer and timer.*

**-(void)render**
*setting OpenGL context and preparing for rendering - building viewport and clearing previous frame.*

***- (void)resetTime;***
*Reset time. When application going background or when rendering stopped.*

***- (void)bindRenderBuffer;***
*Binding OpenGL render buffer.*

***- (int) getElapsedTime;***
*Return elapsed time between two draw callbacks. need for time synchronization.*

***- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;***
*when screen size changed. needed at initialization stage*

***- (void)setupPerspective;***
*setup Perspective projection for any 3d stuff*

***- (void)setupOrtho;***
*setup orthogonal projection need for any 2d stuff*

***- (CGPoint)touchResizeForDevice:(CGPoint)touchLocation;***
*resize touch point for specific device. ViewPort have fixed size (3.2x4.8 units) and we no need to load any different files for ipad or iphone we just scaling this.*

***- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *) acceleration;***
*here is entry point for accelerometer callbacks. here you can send accelerometr info to any other objects*

***- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***

***- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
***- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*) view;***
***- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
*here is entry point for any touches. kepp in mind we should resize touches before sned it to any objects.*

### MViewPort

**-(void)init**
*main initialization point*

**-(void)resetTime**
*reset time when player exit from pause mode or when application become active*

**-(void)setScaleFactor:(float)_scaleFactor**
*set scale factor for retina and non retina displays*

**-(void)loadMultiplayer**
*show game center match maker screen*

**-(void)waitingOpponentTimeOut**
*invoke in opponent connection timeout case*

**-(void)render**
main rendering method

### GameField

**-(void)init**
*main initialization point for all game items - buttons, player, background and etc.*

**-(void)emailAction**
*show email composer screen*

**-(void)gameCenterAction**
*show GameCenter leaderboards screen*

**-(void)multiplayerAction**
*start multiplayer game*

**-(void)singleplayerAction**
*start single player game*

### StoreObserver

Main Object for processing In App Purchase
**- (void) failedTransaction: (SKPaymentTransaction *)transaction**
*when transaction is error*

**- (void) completeTransaction: (SKPaymentTransaction *)transaction**
*when transaction is complete*

# 3. Space Blast - Android

### Description

Space Blast Android port need AndroidOS 2.2 and above.
To Run this Project you need to install Android SDK first ( http://developer.android.com/sdk/index.html ) and then Eclipse Classic IDE (http://www.eclipse.org/downloads/ )

Then you need to import SpaceBlast project in to your Workspace Directory
1. File->Import->General->Existing Projects into Workspace, Next
2. Select root directory of SpaceBlast project
3. Projects->Select All
4. UNCHECK both "Copy projects into workspace" and "Add project to working sets"
5. Finish

Now you ready to build and modify your project.

### Folders Structure

**assets** - folder with all graphics assets: animation atlases and animation information files (plist)
**bin** - folder with compiled binary files (you can use spaceblast.apk for uploading on your device and testing it on real hardware)
**gen** - folder with auto generated source files (do not touch this, because will be overridden automatically at building phase )
**res** - Android OS specific resource files
**res/drawable*** - set of Icon files for low DPI, hi DPI and mid DPI screens.
**res/layout** - main layout file
**raw** - music mp3 files
**src** - project source code
**src/com** - project specific source code
**src/com/badlogic -** other useful libraries
**src/com/basecamp/spaceblast** - main source code folder for this project. Game and Menu Logic, sprite drawing and other game stuff.
**src/org** - shared libraries (libraries like Cocos2d)

# 4. Space Blast - Cocos2d

### Folder Structure

**Space.xcodeproj** - this is MAIN xCode file. just install ios SDK, xcode and simply double click on this file to open it. then you will able to run this App on your device or at simulator on your mac.
**Space -** main folder with all app resources. source code and graphic assets.

**Space/Resources** - graphic assets and music
**Spaces/theora_vorbis** - folder with theora lib source code - http://www.theora.org/ (we need this for video playing as gamefield background)
**BCFAds** - this is lib for BestCoolFun Ads network - http://www.bcfads.com/

## Main Code Objects

**SBTitle** - Ttile Cocos2d Object called in MainMenu for drawing game Logo/Title, start and upgrade buttons, ship selection and other decorations

**SSGame** - Main game object. With all in game logic, GameOver controller, ship rendering code and other decorations.

**SSPlay** - InBetween special object needed for SSGame.

**SSPause** - This object needs for drawing and manipulating of Pause Menu. On The GameField you can push pause button and you will see this Pause Menu.

**SSOver** - GameOver screen. displayed right after ship crashed.

**SSVideo -** Object for playing fullscreen video as game background

# 5. Space Blast - OpenGL

### ESRender

**Main OpenGL rendering object.**

**- (id)init**
*initialization opengl color frame buffer and timer.*

**-(void)render**
*setting OpenGL context and preparing for rendering - building viewport and clearing previous frame.*

***- (void)resetTime;***
*Reset time. When application going background or when rendering stopped.*

***- (void)bindRenderBuffer;***
*Binding OpenGL render buffer.*

***- (int) getElapsedTime;***
*Return elapsed time between two draw callbacks. need for time synchronization.*

***- (BOOL)resizeFromLayer:(CAEAGLLayer *)layer;***
*when screen size changed. needed at initialization stage*

***- (void)setupPerspective;***
*setup Perspective projection for any 3d stuff*

***- (void)setupOrtho;***
*setup orthogonal projection need for any 2d stuff*

***- (CGPoint)touchResizeForDevice:(CGPoint)touchLocation;***
*resize touch point for specific device. ViewPort have fixed size (3.2x4.8 units) and we no need to load any different files for ipad or iphone we just scaling this.*

***- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *) acceleration;***
*here is entry point for accelerometer callbacks. here you can send accelerometr info to any other objects*

***- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
***- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
***- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*) view;***
***- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event view:(UIView*)view;***
*here is entry point for any touches. kepp in mind we should resize touches before sned it to any objects.*

### MViewPort

**-(void)init**
*main initialization point*

**-(void)resetTime**
*reset time when player exit from pause mode or when application become active*

**-(void)setScaleFactor:(float)_scaleFactor**
*set scale factor for retina and non retina displays*

**-(void)loadMultiplayer**
*show game center match maker screen*

**-(void)waitingOpponentTimeOut**
*invoke in opponent connection timeout case*

**-(void)render**
main rendering method

### GameField

**-(void)init**
*main initialization point for all game items - buttons, player, background and etc.*

**-(void)emailAction**
*show email composer screen*

**-(void)gameCenterAction**
*show GameCenter leaderboards screen*

**-(void)multiplayerAction**
*start multiplayer game*

**-(void)singleplayerAction**
*start single player game*

### StoreObserver

Main Object for processing In App Purchase
**- (void) failedTransaction: (SKPaymentTransaction *)transaction**
*when transaction is error*

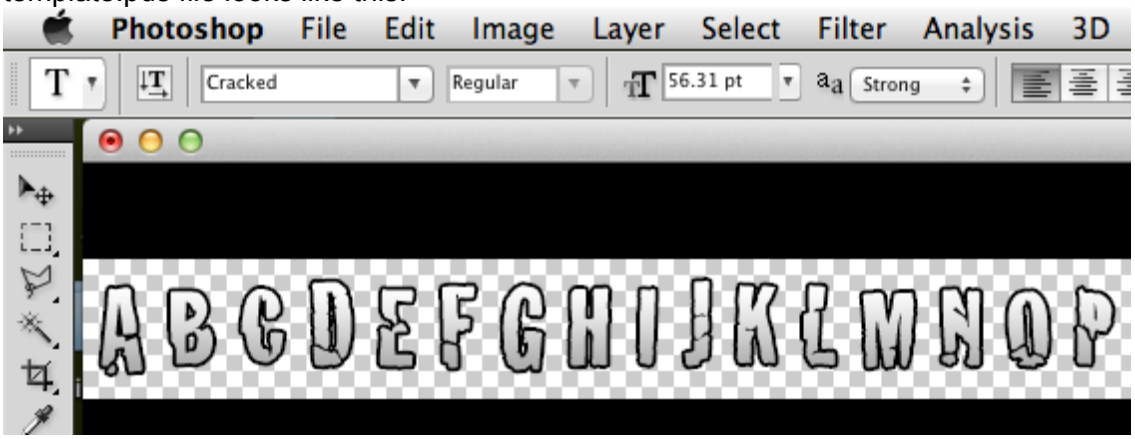
**- (void) completeTransaction: (SKPaymentTransaction *)transaction**
*when transaction is complete*

# 6. How To Build New Fonts In Games

To build bitmaps fonts you need easy script **sfont.py** and a template Photoshop file.

Here you can download script and template file.

template.pds file looks like this:



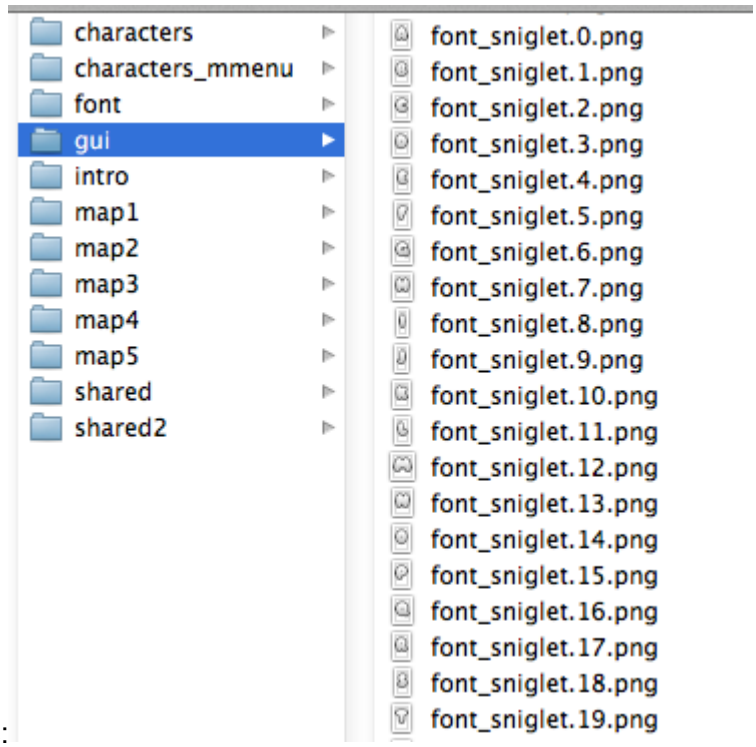

You can change font, gradient direction and anything you want. Just keep enough free space between characters.

Here is really good site with free TTF fonts (100% free for commercial use) -

www.fontsquirrel.com

The **sfont.py** script just slice this one image to bunch of small per character images. Script run over whole image and detect free spaces between characters and split it.

like this:




Run this command to split template image.

*python sfont.py -i ~/Documents/MonsterJump/sourceGraphics/animation/font/font_sniglet.png -o ~/Documents/MonsterJump/sourceGraphics/animation/font_sniglet*

**where:**
**-i** full path to input template image
**-o** full path to output sequence

Now you can add a bunch of files to your GUI (Graphical User Interface) folder and build atlases like you did for regular sprites.

For font initialization in code you should add this:

```
font = [[MFont alloc] initWithName:@"font_sniglet"];
font.scl = CGSizeMake(0.005, 0.005);
font.space = -0.04;
```

then for drawing:

```
font.loc = CGPointMake(1.4, 0.3);
[font print:(char*)"TAP TO START" align:MFontAlignCenter];
```

or

```
char c[32];
sprintf(c, “total coins: %d”, coinsVariable);
[font print:c];
```

# 7. DES File:  What they are and how to create them

In Open GL ES (this is the name of the low level library for drawing 2d or 3d graphics on your iPhone) you can properly load images only in a specific size - this size is called the “power of two” (read more)

*EXAMPLE: 512pixels x 512pixels = $2_9$ pixels x $2_9$ pixels or 1024 x 1024 =  $2_{10}$ x $2_{10}$*

But in real life is really hard to keep all your graphics materials in **power of two** sizes (POT). In modern OpenGL versions you can find a way to load non-POT  images but this is not effective and will eat more memory while holding your textures.

To solve this issue you can put all you small images in to one really big image (sometimes called a resource image, atlas, sprite sheet, sprite list, etc).  We call this an **Atlas**.

So what does DES mean?

A DES file is very tiny plain text file. In this file you can find information about the atlas file including: where to find atlas (resource) image AND describe the position of original image on atlas image.

Basically, this DES file tells us where certain images are in the big ATLAS image.

As example if you have a folder called *MainMenu*  and two files *button.png* and *levelIcon.png* after atlas making process you’ll have two DES files button.des and levelIcon.des and two big images MainMenu.png and MainMenu_hi.png *( *_hi.png is for Retina Displays and IPad )*

**PLEASE NOTE:  This is our IN HOUSE custom made system.  We don’t have some super pretty software that looks great to make atlases :)**

To build Atlases you need **mkatlas.py** - easy python script for making Sprite Atlases.

**You can download it here**

*(Read more about sprite atlases if you want, here: Sprites in Wiki )*

Here’s how to use the PYTHON SCRIPT link above.

### Installing additional software

**1. Python 2.5 +** read more
First of all you need python 2.5 and higher is you working on Mac OSX Snow Leopard (10.6.x) and higher you are good and no need to install python again.

**2. Python Image Library (PIL)** read more
This is python module for image manipulation.  The script requires this module to work.
Here is a good tutorial about installing PIL on your MAC:

http://passingcuriosity.com/2009/installing-pil-on-mac-os-x-leopard/

**3. Install argparse pytohn module**
Follow this instruction to install **argparse** module
http://pypi.python.org/pypi/argparse

## Building your first DES file

First of all you need to open the **Terminal App** on Mac OSX (Applications > Utilities > Terminal)

Move to folder where mksprite.py script located

Then run command:

*python mkatlas.py -i ~/Documents/appermind/MonsterJump/sourceGraphics/animation/shared -s 1.0 --spacing 4 -z 2048 -w 2048*

**where:**
**-i** full path to input folder (with list of PNG files)
**-s** scale of output images
**--spacing** spacing between each sprite in atlas
**-z** maximal width of output atlas
**-w** maximal height of output atlas

## Building your first Animated DES file

This is super easy. To build DES files with animation you only need to keep proper name convention.

It should be like this:

[file-name].[frame_number].png

example:

enemy_death.001.png
enemy_death.002.png
enemy_death.003.png
etc.