

The Essential Guide to Flash Games

Building Interactive Entertainment with ActionScript 3.0

Jeff Fulton
Steve Fulton



The Essential Guide to Flash Games

Building Interactive Entertainment with ActionScript

Copyright © 2010 by Jeff Fulton and Steve Fulton

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-2614-7

ISBN-13 (electronic): 978-1-4302-2615-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please e-mail info@apress.com, or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>. You will need to answer questions pertaining to this book in order to successfully download the code.

Credits

President and Publisher: Paul Manning
Coordinating Editor: Kelly Moritz

Lead Editor: Ben Renow-Clarke
Copy Editor: Heather Lang

Technical Reviewer: Iain Lobb
Compositor: Mary Sudul

Editorial Board: Clay Andres, Steve Anglin, Mark Beckner,
Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan
Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper,
Frank Pohlmann, Ben Renow-Clarke,
Dominic Shakeshaft, Matt Wade, Tom Welsh
Indexer: BIM Indexing & Proofreading Services
Artist: April Milne
Cover Designer: Kurt Krames

For Jeanne, Jacob, Ryan, and Justin
–Jeff Fulton

For Dawn, Rachel, Daphnie, and Kaitlyn
–Steve Fulton

Contents at a Glance

About the Authors	xxv
About the Technical Reviewer.....	xxvii
Acknowledgments	xxviii
Preface	xxx
Layout Conventions	xxxiii
Part 1: The Basic Game Framework	1
Chapter 1: The Second Game Theory.....	3
Chapter 2: Creating an AS3 Game Framework	45
Chapter 3: Creating Super Click.....	99
Part 2: Building Games	135
Chapter 4: Laying the Groundwork for Flak Cannon	137
Chapter 5: Building the Flak Cannon Game Loop.....	173
Chapter 6: Laying the Groundwork for No Tanks!	207
Chapter 7: Creating the Full No Tanks! Game	253
Chapter 8: Creating the Color Drop Casual Puzzle Game	329
Chapter 9: Creating the Dice Battle Puzzle Game	365
Chapter 10: Blit Scrolling in a Tile-Based World	407
Chapter 11: Creating an Optimized Post-Retro Game	473
Chapter 12: Creating a Viral Game: Tunnel Panic	571
Index	621

Contents

About the Authors	xxv
About the Technical Reviewer	xxvii
Acknowledgments	xxviii
Introduction	xxx
Layout Conventions	xxxiii
Part 1: The Basic Game Framework	1
Chapter 1: The Second Game Theory	3
Making games is an iterative process	3
Understanding why you want to make games	5
Who this book is for	6
So we're not starting at the beginning?	6
Comparing AS3 and AS2.....	6
You can't stick with AS2 because it's dead	7
What you need to use this book	7
The craft of making games in the AS3 game framework	8
Object-oriented methodology.....	8
Encapsulation	8
Inheritance	9
Using object-oriented design to make games.....	9
Creating the basic game framework	11
State loop.....	11
Game timer	12
Event model.....	13
Testing the game	16
Your Second Game: Balloon Saw	17
Assets for this game	17
Graphics.....	17
Sounds.....	18
Settings.....	19
The Code	19

Balloon Saw gameplay	22
Breaking down the Balloon Saw code	23
Imports	23
Variables	24
Game constructor	24
Initializing the game loop	25
The playGame() function	26
Making the Balloons.....	27
Moving balloons	28
Testing collisions between the saw and balloons	28
Ending the game	30
Creating your third game: Pixel Shooter	30
Pixel Shooter game design	31
Pixel Shooter graphics	31
Pixel Shooter sounds	32
Pixel Shooter code	32
Pixel Shooter code highlights	38
New variables	38
Starting and restarting the player	38
Tracking multiple objects	39
Firing missiles	40
Detecting collisions on multiple types of objects	41
Creating explosions	42
Summary	43
Chapter 2: Creating an AS3 Game Framework	45
Exploring the Framework.....	45
The GameFrameWork.as class	46
The FrameWorkStates.as class.....	46
The BasicScreen class and SimpleBlitButton helper class.....	46
The ScoreBoard class and SideBySideScoreElement helper class	46
The Game class.....	47
The Custom event classes	47
The CustomEventButtonId.as class	47
The CustomEventLevelScreenUpdate.as class.....	47
The CustomEventScoreBoardUpdate.as class.....	47

The framework package structure	48
The source folder	48
The classes package	48
The projects package	48
The Flash IDE package structure	49
The Flex SDK package structure	50
The Main.as and StubGame.as files	50
Starting a project using the framework packages	50
Creating the stub game project in the Flash IDE	50
Creating the stub game project in Flash Develop	51
Creating game timers	52
Defining “frame timer tick”	52
State Machines	52
The FrameWorkStates.as class file	53
The state variables	54
The GameFrameWork.as class file	54
The class imports	60
The variable definitions	60
The state control variables	61
The background fill variables	61
The timer variables	61
The screen definition variables	62
The ScoreBoard variables	62
The Game object variable	62
The wait variables	63
The constructor function definition	63
The init function definition	63
The setApplicationBackGround function definition	63
The startTimer function definition	63
The runGame function definition	64
The switchSystemState function definition	64
The systemTitle function definition	65
The systemWaitForClose function definition	65
The okButtonClickListener function definition	65
The systemInstructions function definition	66
The systemGameOver function definition	66

The systemNewGame function definition	66
The systemNewLevel function definition	67
The systemLevelIn function definition.....	67
The systemWait function definition	68
The waitCompleteListener function definition	68
The systemGameplay() function definition	68
The custom event listener functions	68
The scoreBoardUpdateListener function definition for Main.as	68
The levelScreenUpdateListener function definition for Main.as.....	69
The gameOverListener function definition for Main.as	69
The newLevelListener function definition for Main.as.....	69
Framework classes beyond Main.....	70
The BasicScreen class	70
Class import and variable definition section for BasicScreen	72
Class constructor function definition	72
The createDisplayText function definition	73
The createOkButton function definition	74
The button listener function definitions	75
The SimpleBlitButton class	75
The class import and variable definition for the SimpleBlitButton class	77
The constructor definition for the SimpleBlitButton class.....	77
The changeBackgroundColor function definition	79
The CustomEventButtonId class	79
Class import and variable definition for the CustomEventButtonId class.....	80
The constructor definition for the CustomEventButtonId class	80
The clone function definition for the CustomEventButtonId class.....	81
The CustomLevelScreenUpdate class	81
The CustomEventScoreBoardUpdate class	82
The ScoreBoard class	83
Class import and variable definition for the ScoreBoard class.....	84
The createTextElement function definition.....	85
The update function definition	85
The SideBySideScoreElement class	86
The constructor definition.....	87
The setLabelText and setContentText definition functions	88
The Game class.....	88

A stub game class	89
Game objective	89
Game state screen flow	89
Game ScoreBoard constants	90
Creating the Main.as class	90
Class import and variable definitions for the Main class	92
Applying the constructor and init function definitions	92
Creating our game instance	93
Setting the application back ground	93
Creating the score board	93
Creating the title screen	93
Creating the instructions screen	94
Creating the game over screen	94
Creating the level in screen	95
Setting the initial state machine's state	95
Starting the game timer	95
The stub Game.as class	95
What we are doing in this class	97
Test it!	98
Summary	98
Chapter 3: Creating Super Click	99
Creating a Super Click game design specification	100
Defining the game basics	100
Calculating level progression	101
Defining the basic screens needed	101
Defining the game play variables	101
Defining the custom ScoreBoard elements	102
Describing the game play and flow	103
Evaluating the end of a level	103
Evaluating the end of a game	103
Defining the necessary events	103
Defining the game-specific custom classes	104
Creating Super Click	105
Starting the project	105
Creating the Super Click game project in the Flash IDE	105

Creating the Super Click game project in the Flash Develop	106
The Class file list.....	106
Creating the Main.as class	106
Importing classes and defining variables for Main.as	109
Creating our game instance	110
Setting the application background	110
Creating the Scoreboard.....	110
Creating the title screen	111
Creating the instructions screen	112
Creating the game over screen.....	112
Creating the Level In screen	112
Setting up the initial state machine	113
Starting the game timer.....	113
Creating the Game class for Super Click	113
Importing the classes and defining the variables for SuperClick.as	114
Defining the constructor and init function for SuperClick.as	116
Defining the newLevel function	117
Calling the runGame function definition	119
Defining the update function	120
Adding circles to the screen	121
Updating the on-screen circles	121
Updating the ScoreTextField instances	121
Optimizing the loops	121
Defining the removeCircle function	122
Defining the removeScoreText function.....	122
Defining the checkCollisions function	123
Fading out the circle.....	124
Defining the addToScore function	124
Defining the render function.....	125
Defining the checkForEndOfGame function	126
Defining the checkforEndLevel function	126
Defining the cleanUp function	126
The Circle Class	127
Defining the Circle class	127
Creating the Circle class in the package structure	128
Importing the classes and declaring variables for Circle.as.....	128

Defining the constructor and init function for Circle.as	129
Defining the update function for Circle.as	130
Defining the dispose function for Circle.as	130
Updating the clickedListener function definition for Circle.as	131
The ScoreTextField class	131
Defining the ScoreTextField class	131
Creating the ScoreTextField class in the package structure	131
Defining the update function for ScoreTextField	133
Defining the dispose function for ScoreTextField	133
Test it!	133
Summary	133
Part 2: Building Games	135
Chapter 4: Laying the Groundwork for Flak Cannon	137
A short history of Missile Command	138
Designing the game.....	139
Game development concepts in this chapter	139
Adding game assets to the library	140
Creating graphics using Spritelib GPL	140
Creating sounds using SFXR	143
Library differences in Flash and Flash Develop/Flex	144
Using graphic assets	145
Using sound	147
Creating a sound manager	147
Creating difficulty with settings	152
Difficulty settings	152
Creating sprites that move on a vector	155
Defining a sprite that moves between two points	155
Shot.....	155
BonusPlane.....	159
Ship	160
Creating objects that move on a continuous vector: Enemy.....	161
Creating animated sprites.....	166
Flak	166
Explosion	169
Summary	171

Chapter 5: Building the Flak Cannon Game Loop	173
Understanding the Flak Cannon game flow	173
Updating GameFramework.as	174
Defining Main.as	176
FlakCannon.as	179
Importing classes	179
Setting the FlakCannon.as properties and constructor	180
Starting a new game	182
Starting a new level	183
Handling mouse events and starting the crosshairs	184
Placing the ships	186
Handling newLevel events	187
Testing out the game	188
Creating the game loop	188
Checking for enemies	189
Checking for a bonus plane	192
Updating objects	194
Removing objects	196
Detecting basic bitmap collisions	196
Rendering objects	203
Ending a level	203
Ending the game	205
Playing the game!	205
Summary	205
Chapter 6: Laying the Groundwork for No Tanks!	207
The No Tanks! game design	207
Game development concepts in Chapters 6 and 7	208
Adding game assets to the library	209
Using Spritelib GPL	209
Using a tile sheet	209
Organizing game levels into tiles	210
Creating a level	211
Creating a level with Mappy	211
Sprites vs. tiles vs. background	212
Creating the tile sheet	212

Creating a game level with Mappy	216
Creating a two-layer map.....	217
Creating the background layer.....	217
Creating the sprite layer.....	218
Saving the level .FMP file	219
Exporting the level from Mappy	219
Exporting the background layer	219
Exporting the sprite layer	220
Setting up our project	221
Creating the No Tanks! game project in the Flash IDE	221
Creating the No Tanks! game project in Flash Develop	222
Describing our tile sheet in XML	223
Creating the TilesheetDataXML.as class.....	223
Tile attributes	225
How this code works	225
Using the tile sheet data in code	225
Create the GameDemo.as class.....	226
Reading the tile sheet data	228
The constants	228
The initTileSheetData function.....	229
Testing it out	230
Displaying the level data on the screen	230
Organizing your code.....	231
Adding the Library.as class for Flex projects only	231
Adding the library assets for the Flash IDE only.....	232
Using the library assets in Flex and the Flash IDE	232
Tile sheet blitting and sprites	233
Defining “blitting”	233
Defining “sprite” further	234
Understanding the differences between sprites and blitting	234
Bringing Flash into the mix	234
Understanding the difference between tile and sprite sheets	235
A full screen blit.....	235
Individual object blits.....	236
Combining types of blits.....	236
Testing blitting render speed	236

Testing the timeline-based method	236
Testing individual object blits	237
Testing the sprite full-screen blit	237
Making sense of these results	238
The TileSheet class	238
Understanding the attributes of a TileSheet	239
Reading the level data	240
The Level class	240
Creating the Level1 class	241
Updating the New GameDemo.as file	243
The class imports	243
The variable definitions	243
The constructor	246
The readBackGroundData function	246
The readSpriteData function	246
The drawLevelBackGround function	247
Organizing our game rendering	247
Summary	251
Chapter 7: Creating the Full No Tanks! Game	253
Tile-based movement theory	254
Tile jumping	254
Smooth tile-to-tile movement	254
The BlitSprite Class	255
Animating with the BlitSprite class	257
The TileByTileBlitSprite class	258
Moving the player character through the maze	259
The center tile trick	260
Adding the player (iteration 1)	260
Changing the class name for iteration 1	261
Adding the new framework classes to the import section	261
Defining the iteration 1 variables	261
The init function for iteration 1	262
The iteration 1 restartPlayer function	263
The readSpriteData function	263
Testing iteration 1	265

Moving the player using key logic (iteration 2)	266
Changing the class name for iteration 2	266
Adding a simple placeholder game timer.....	266
Switching player move state with the arrow keys	267
Adding the keypress logic.....	268
Testing iteration 2	268
Updating the move states for player movement (iteration 3)	269
Changing the class name for iteration 3	269
Adding the move state constants.....	269
Changing the runGame function for iteration 3.....	269
Updating the checkInput function	270
Adding the checkTile function.....	272
Adding the checkCenterTile function	274
Adding the switchMovement function	274
Testing iteration 3	275
Updating and rendering player movement (iteration 4)	276
Changing the class name for iteration 4	277
Updating the variable definitions for iteration 4.....	277
Updating the runGame function for iteration 4.....	277
Adding the update function.....	278
Adding the render function.....	280
Testing iteration 4	281
Adding and moving enemy tanks (iteration 5)	282
Changing the class name for iteration 5	282
Updating the variable definitions for iteration 5.....	282
Updating the init, newGame, and newLevel functions.....	283
Adding the setRegions function	283
Changing the readSpriteData function.....	285
Testing iteration 5	286
Allowing enemy tank movement with the AI (iteration 6)	286
Changing the class name for iteration 6	287
Making additions to the restartPlayer function.....	287
Modifying the update and render functions for enemy tanks	287
Adding new functions for enemy tank AI	290
Chasing the player	290
The chaseObject function	291

The checkLineOfSight function	296
The fireMissileAtPlayer stub function	299
Adding to the variable definitions	299
Testing iteration 6	299
Integrating into the framework	300
Integrating onto the Main.as framework class	300
The application background and game Sprite location	304
The ScoreBoard	304
The Screens	304
The Sounds	304
The New Function Overrides for the Sounds	304
Finishing up the Library class	305
Finalizing the Level.as and Level1.as file	306
Finishing up the NoTanks.as file	307
Changing the class name for NoTanks.as	307
Adding to the class import section for NoTanks.as	307
Adding the new NoTanks.as variables	308
Adding the new NoTanks.as init function	309
Creating the newGame function	309
Creating the newLevel function	310
Creating the restartPlayer function	311
Overwriting the runGame function	311
Adding the CheckInput function	312
Improving the update function	314
Adding the checkHitWall function	316
The checkCollisions function	316
Updating the render function for NoTanks.as	320
Checking for the end of a level or game	320
Creating functions to fire missiles	321
Changing the readBackGroundData function	323
Changing the readSpriteData function	323
Changing the checkLineOfSight function	324
Adding the createExplode function	324
Coding the object cleanup functions	325
Testing the final game	326
Extending the game	327

On to Color Drop	327
Chapter 8: Creating the Color Drop Casual Puzzle Game	329
Understanding the Evolution of Casual Games	329
Designing the game	331
Game development concepts in this chapter	332
Adding this game’s assets to the library	333
Adding sounds	333
Adding graphics	333
Defining the classes for Color Drop	333
Updating the Main class for Color Drop	334
Creating the Block class	336
Creating the CustomEventClickBlock class	340
Controlling difficulty with a level class	341
Creating the GameStates class	341
Implementing the basic game structure	342
Initializing the game	345
Adding the ColorDrop state machine	347
Adding blocks to the screen	350
Expanding update and render for ColorDrop	352
Waiting for user input	353
Using a nonrecursive function to test blocks	355
Removing blocks	358
Ending a level or the game	362
Test it!	364
Summary	364
Chapter 9: Creating the Dice Battle Puzzle Game	365
Getting started with intellectual property law	365
Considering copyright	365
Considering trademarks.....	366
Adding patents to the mix	367
Following the Golden Rule.....	368
Designing the Dice Battle game	369
Game development concepts in this chapter	370
Adding game assets to the library	371

Adding Graphics for Dice Battle.....	371
Adding sound for Dice Battle	372
Playing Soundtracks in Main.as	373
Revamping ScoreBoard.....	373
Playing SoundTracks.....	374
Updating the Main.as class.....	375
Creating AI difficulty with a class	379
Creating the Die class	379
Creating the CustomEventClickDie class	383
Creating the Character class	384
Adding the GameStates class	385
Setting up the game in Game.as	385
Creating a computer player	389
Taking turns using new game states	390
Capturing and scoring a player move	392
Creating the computer’s minimax-style AI	395
Analyzing the Dice Battle AI	395
Discussing the Dice Battle AI.....	396
Ending the level or game	401
Viewing the rest of the code for Dice Battle.....	401
Test it!	405
Summary	405
Chapter 10: Blit Scrolling in a Tile-Based World	407
Designing and getting started with Drive She Said.....	407
Game development concepts in this chapter	409
Defining new classes	409
Integrating with the game framework.....	409
Modifying the Library class	410
Adding the custom LevelInScreen Class	410
Transitioning between levels	410
Understanding free-form tile-based movement.....	410
Defining how free-form tile-based movement works.....	410
Using art-based MoveClip stage scrolling.....	411
GAS (GotoAndStop) tiles	411
Using tile-based blit scrolling	411

The World	411
The Camera	412
The Buffer	412
The Output Canvas	412
Creating the game world	413
Creating the Tile Sheet	413
Detecting collisions on the WALL tiles	414
Defining the game levels	415
Obtaining the level data	415
Applying basic car physics to our game	416
Moving forward and backward	416
Moving in a direction	417
Preparing to create our Drive She Said game	418
Creating the game project in the Flash IDE	419
Creating the game project in the Flash Develop	419
Updating the Main.as class for Drive She Said	420
Adding ScoreBoard elements	424
Modifying the screens	424
Modifying the sounds	424
Transitioning to the LevelInScreen	425
Updating the LevelInScreen text	426
Increasing the game Frame Rate	426
Creating the CustomEventHeartsNeeded.as class	427
Creating the Library.as class	427
Creating the new framework classes	429
Creating the BasicFrameTimer class	429
Creating the LookAheadPoint class	431
Creating the CarBlitSprite class	432
Creating the Camera2D class	434
Double (and triple) buffering	435
Creating the classes specific to Drive She Said	435
The TileSheeDataXML class	436
The Level.as class	437
The Level1.as Class	437
Iterating the Game class	439
Creating the Game class shell, variables, and constructor (iteration 1)	439

Keypress constants.....	442
The internal state machine.....	442
The tiles, display, and world.....	442
The camera.....	443
Car sounds.....	443
Creating a working class (hero?) in iteration 2	443
Setting up the game (iteration 3)	445
The init function.....	445
The newGame function.....	447
The initTileSheetData function.....	448
The newLevel function	449
The setUpWorld function	450
The restartPlayer function.....	452
Moving the car and game world together.....	453
The systemGamePlay function	457
Testing game iteration 3	457
Adding player input and the update / render loop (iteration 4).....	458
The update function	458
The checkInput function.....	461
The render functions.....	462
The drawCamera function.....	463
The drawPlayer function	464
Testing iteration 4	465
Detecting collisions and the end of a level or the game (iteration 5).....	466
The checkCollisions function	466
The rest of the game functions	469
Testing the final game.....	471
Extending the game.....	471
Summary	472
Chapter 11: Creating an Optimized Post-Retro Game	473
Understanding post-retro games	473
Defining post-retro	474
Exploring the features of post-retro games.....	474
Emerging post-retro game features	475
Tracing the history of post-retro games	475

What the post-retro genre means to developers	478
Designing Blaster Mines	478
Game development concepts in this chapter	480
Modifying the game framework	480
Checking stage access	481
Creating the New addToStage function	481
Adding pause and mute functionality to the framework	481
Adding the time-based step timer	482
Adding the runEnterFrame function	483
Optimizing using render profiling	485
Designing the FrameRateProfiler technical design	489
Monitoring frame rate and memory usage	493
Changing the game class	497
Getting Started with the Blaster Mines Project	498
Creating the Blaster Mines game project in the Flash IDE	498
Creating the Blaster Mines game project in Flash Develop	499
Creating the Main.as class for Blaster Mines	500
Implementing the pause and mute functionality	506
Adding the new constructor function	506
Adding the addToStage function	506
Implementing the time-based step timer for Blaster Mines	507
Customizing FrameRateProfiler for Blaster Mines	507
Creating the frameRateProfileComplete function	508
Creating the Library.as class	508
Modifying the SoundManager class	509
Optimizing with object pooling	510
Conserving processor execution time	510
Conserving memory	510
Implementing object pooling in our game	511
Creating the technical specifications for object pooling in Blaster Mines	511
Adding the private variables	511
Instantiating a particle in the pool	512
Making a particle active	512
Making a particle inactive	513
Optimizing with single-unit processing and memory conservation	513
Reusing global event objects	514

Optimizing with look-up tables.....	514
Creating the movement vector look-up table	515
Accessing the vectorRotationList look-up table	515
Optimizing screen-based blit scrolling with ScrollRect.....	516
Optimizing BitmapData reuse for the radar screen.....	517
Creating the new game classes	518
Designing the BlitArrayAsset class	518
Designing the BasicBlitArrayObject class	521
Designing the BasicBlitArrayParticle class	523
Designing the BasicBlitArrayProjectile class	525
Designing the BlitArrayPlayerFollowMouse class.....	526
Designing the MineManager class.....	531
Designing the ParticleManager class	533
Designing the ProjectileManager class.....	535
Designing the Mine class	537
Building the Blaster Mines class.....	538
Creating the Blaster Mines class shell	539
Adding the Blaster Mines game init functions	541
Adding the newGame and newLevel functions.....	545
Updating the game loop and internal state machine	552
Adding the update, autoShoot, render, and collision functions.....	554
Adding auxiliary functions	560
The full GameFrameWork class	562
Test it!	570
Summary	570
Chapter 12: Creating a Viral Game: Tunnel Panic	571
Defining viral web games.....	571
Distributing a viral web game	572
Using your own web site	572
Using social news sites.....	572
Twitter	572
Facebook	573
Uploading to portals.....	573
Social gaming sites	573
Selective portals.....	574

Making money from your viral game	575
Using in-page ads	575
Entering contests	575
Inserting in-game ads with Mochi Media	575
Obtaining licenses and sponsorships	576
Exclusive Licenses	577
Sponsorships	577
Nonexclusive licenses	577
API licenses	577
Working with Adobe Flash Platform Services	579
Securing your viral games	579
Using site locking	579
Encrypting your game	580
Marketing viral Flash games	580
Some Other Great Web Resources	581
Preparing to create our Tunnel Panic game	582
Creating the game project in the Flash IDE	582
Creating the game project in the Flash Develop	583
Preloading in the Flex SDK	584
Adding the compiler directive	584
Adding the Preloader class code	585
Preloading in the Flash IDE	587
Adding files to the library	587
Creating the timeline	588
Creating an asset holder MovieClip	589
Linking the assets	589
Putting the assets into the asset holder	590
Placing the asset holder on the main time line	591
Framework Changes for Flash IDE preloading	592
Adding in the new Flash IDE preloader state to the framework	592
Adding new variables for the preloader state	593
Defining the preloadScreen instance	593
Setting the preloader to be the first state in Main	593
Adding to the switchSystemState function	593
Adding the new systemPreload and addSounds functions	594
Adding Mochi preloader ads and leader boards	595

Importing the Mochi package.....	595
Changing the Main.as object type	595
Adding Mochi-related variables to the framework.....	595
Changing switchSystemState	596
Making Mochi ad-specific changes	596
Making the leader-board-specific changes	597
Creating our own viral game	598
Designing the Tunnel Panic game	599
Creating the PlayerSprite object	600
Creating the play field	601
Adding the obstacles	601
Animating the player ship’s exhaust	601
Using dirty rect erase	602
Increasing the game difficulty	603
Ending the game.....	604
Creating the Main.as for Tunnel Panic	605
Changing Game.as for Tunnel Panic.....	605
Changing Main.as for Tunnel Panic.....	605
Creating the Library.as class for Flex SDK only	609
Adding to the Flash IDE Library.....	609
Coding the TunnelPanic.as class.....	610
Test it!	619
Summary	619
Index	621

About the Authors



Jeff Fulton has been making and playing computer and video games as a hobby for over 30 years. In his early years, Jeff dreamed of programming games for a living, but he never considered a career as a professional game developer. Rather, he fashioned himself as a writer and filmmaker. After creating a handful of no-budget college films on video with friends, he discovered that he was just as drawn to the technical side of the process as he was to the creative side. The Atari ST computer that he and his brother Steve had used to create animations and title sequences for these films turned out to be the inspiration that pushed him toward professional game development. In

1991, Jeff read a series of game development articles by the great Llamasoft game developer Jeff Minter, in the back of ST Action magazine. This series of articles planted the seed of his future career. He was drawn to the colorful language, the exciting stories, and the wonderfully post-retro games that Minter described and created. With the understanding that game development was not just a creative pursuit, Jeff dropped all of his film classes at the university and took as many technical and business classes as he could find. After college, Jeff dove into game coding and development in his spare time while working during the day coding systems in Perl and C++ for a variety of business applications. When the web boom hit in the late 1990s, Jeff jumped at the chance to parley his skills into a job making kids' web sites and games for a large multinational corporation. In 2006, with over 200 games and sites under his belt, he and his brother Steve (also a game and site developer) started their own web site, www.8bitrocket.com. The goals of the site are to celebrate web and retro style games and to teach others the methods they had gathered from their considerable experience in game design, coding, and development.

Jeff's all time favorite games include Super Breakout (Atari 2600), River Raid (Atari 2600), Dragon Stomper (Atari 2600), Rally Speedway (Atari 800), Fort Apocalypse (Atari 800), Mule (Atari 800), Asteroids (Arcade), Ms. Pac-man (Arcade), Star Castle (Arcade), Phantasie (Atari ST), Megaroids (Atari ST), Oids (Atari ST), Anco Player Manager (Atari ST), Galaga (Arcade), Food Fight (Atari 7800), Tempest 2000 (Jaguar), Wolfenstein 3D (DOS), Medal Of Honor (PS1), Point Blank 2 (PS1), Duke Nuke 'Em 3D (DOS), System Shock 2 (PC), Tony Hawk Pro Skater (PS1), Half Life (PC), New Star Soccer (PC / Mac), and Baldur's Gate: Dark Alliance (PS2).



A self-professed Atari nerd, **Steve Fulton** has wanted to make games as long as he can remember. While dabbling in both film school and rock journalism, Steve found his footing as a C++/Assembly language programmer in the early 1990s. However, after working several the client-server software companies that were crushed under the weight of the burgeoning World Wide Web, Steve saw the light in 1995 and started developing in HTML/Java and Perl/CGI. This shift led to very early (before the Web explosion) jobs creating web sites and interactive applications like chat, word searches, and simple games that utilized only HTML and server-side scripts. After developing web sites for small clients and large corporations alike for most of the 1990s, Steve's path led to developing customer-facing web sites for one of the world's largest entertainment companies. For the past ten years, he has worked on web based games and entertainment using a variety of technologies including Flash. Along with dozens of high-profile, high-traffic web sites, and web-based communities, Steve has designed, developed, or programmed hundreds of Flash applications and web sites, including dozens of single-player and multiplayer web-based games played over 1 billion times. Along with his brother Jeff, Steve runs the popular and influential Flash and retro game development and news site www.8bitrocket.com. The site is updated frequently with news, tutorials, demonstration games, experiments, and musing about Flash and the viral web game world.

Steve's favorite games of all-time are Breakout (Atari 2600), Asteroids (coin operated), Star Castle (coin operated) River Raid (Atari 2600), Castle Wolfenstein (Apple IIe), Galaga (coin operated), Time Pilot (coin operated), Star Wars (coin operated), MULE (Atari 800), Ultima IV (Atari 800), Temple Of Apshai (Atari 800), Dungeom Master (Atari ST), Oids (Atari ST), Food Fight (Atari 7800), Wizards's Crown (Atari ST), Anco Player Manager (Atari ST), Machine Bride of Pinbot (pinball coin operated) Dune 2 (PC), Fallout (PC), Desert Strike (Genesis), Tempest 2000 (Jaguar), Zolar Mercenary (Lynx), Roller Coaster Tycoon (PC), Baldur's Gate: Dark Alliance (PS2), Knights Of The Old Republic (PC), Final Fantasy 1 and 2(GBA), Wii Sports Resort (Wii), Mario And Luigi: Partners In Time (DS), Pinball Hall Of Fame: Williams Collection (Wii), Pac-Man Championship Edition (Xbox Live Arcade), Puzzle Quest (DS), Bookworm Adventures(PC), and Dragon Age (PC).

About the Technical Reviewer



Iain Lobb is a freelance Flash games developer and designer, and a nine-year veteran of London's hectic digital agency scene. Until early 2009, Iain was head of interactive at the award-winning studio Bloc, where he cocreated some of the best-loved games on the web, including ZW0K!, Stackopolis, Pop Pirates, and Meta4orce. Along the way, his work has picked up awards and nominations from the likes of BAFTA, the FWA, Cannes Lions, and NewGrounds, with Stackopolis winning the 2006 Webby award for best game.

Iain now runs his own game company, Dull Dude, where he develops games for clients, as well as creating original game ideas and characters.

Acknowledgments

Writing a book, especially a technical one, requires a team much larger than the author, or authors, in this case.

First, thanks to the incredible team at Apress / friends of ED: Ben who gave us the chance to prove ourselves; Kelly for putting up with the temperamental twins; Heather for making it read much better than it should; Iain Lobb for helping us to standardize and improve every aspect of our code. You guys rock!

We'd also like to thank the teachers and mentors that helped us find our way including Mr. Hughes and Ms. Brown from Foster A. Begg Middle School; Mr. Scott, Mr. Lang, Mr. Fredricks, Mr. Holland, Mr. Sumpter, and Ms. Staich from Mira Costa High School; Dr. Gessford and Dr. Godfrey from Long Beach State; Dave Robinson, Laurie Shammel, Myron Bowman, Tim Cashin, and Shel Klee from TXS; Mike Gitchel and Joe Loo from Investors Business Daily; and John Watson, Bruce Williams, and Tim Locke from the wild corporate world.

We want to thank the game developers, game journalists, and authors who have instructed and inspired us over the years including Nolan Bushnell, Al Alcorn, David Crane, Alan Miller, Ed Logg, Dan(i) Bunten, Ihor Wolosenko, David Heller, Chris Crawford, Bill Budge, Rob Fulop, Lord British, Winston Douglas Wood, all the guys from FTL, the Bitmap Brothers, Jeff Minter, Jeffery Stanton, Chris Sawyer, Katz/Kunkel/Worley, Steve Levy, Steven L. Kent, Tom Chick, Jeff Green, Johnny L. Wilson, Andrew Bub, Simon Carless, Ari Feldman, Jobe Makar, Gary Rosenzweig, Keith Peters, and Colin Moock. Plus, we thank all at Flash Game License, Mochi Media, Adobe, and Macromedia (R.I.P.)—especially Jonathan Gay, the inventor of Flash.

We also like to acknowledge the fellow game heads who have contributed to our love of video games including Carrie Lennihan, Alex Mortensen, Eric Barth, John and Richard Campi, Kenny Brown, Mike Jackson, Greg Dyer, Scott Johnson, Mike Foti, Evan Pershing, Jonas Sills, Wesley Crews, Brandon Crist, James Ku, Ian Legler, John Little, Dan Cady, Chris Cutler, Scott Delamater, Scott Jeppesen, Alan Donnelly, Mark "Icky Dime" Grossnickl, Richard "Mr. Atari ST" Davey, Richard "Squize" Myles, Oliver "nGfx" Sons, Dave "Retro Shoot" Munsie, Julian "LongAnimals" Scott, Tony "The Symbol" Pa, and everyone else on The Board, plus Ace the Super Villian and all of our friends at www.8bitrocket.com.

We'd also like to thank the rock bands without which we would not have made it through high school, college, and beyond: The Alarm & Mike Peters, Slade, Stiff Little Fingers, The Damned, The Business, Icicle Works, The Smithereens, The Hoodoo Gurus, Soul Asylum, Cactus World News, Minor Threat, Midnight Oil, CH3, Fugazi, The Gear Daddies, The Dead Milkmen, The Mr. T Experience, Naked Raygun, The Nils, All, The Descendents, TSOL, The Long Ryders, Drivin' N Cryin', Green Day, Big Drill Car, U2, Big Country, The Who, Social Distortion, The Wonderstuff, The Sweet, The Equals, Love, The Replacements, Husker Du, The Shoes, Grant Hart, The Goo Goo Dolls, Pete Drogé, Tom Petty, Material Issue, Weezer, Pearl Jam, The White Stripes, Daft Punk, LCD Soundsystem, 8 Bit Weapon, The Mooney Suzuki, the Gas Light Anthem, The Brady 6, Pain, Snow Pink, and many more. We'd also like to thank the filmmakers who have enlightened us, including John Hughes, Wes Anderson, Allen Moyle, Paul Feig, Judd Apatow, Mike White, Amy Heckerling, and Jason Reitman, plus Sid and Marty Croft. As well, we'd like to mention the authors who have written the books we love the most, including Donald J. Sobol, Robert A.

Arthur, Sue Townsend, Edward Packard, R.A. Montgomery, Paul Zindel, Judy Blume, Frank Portman, Nick Hornby, D.B. Weiss, Doug Stumpf, Steve Almond, Joe Meno, Mark Haddon, Sara Gruen, Chuck Klosterman, Dr. Seuss, Richard Scarry, and Bill Pete.

Finally, we'd like to thank our sister, Carol, for trying to teach us math on her chalkboard when we were just four years old, and our other sister Mari who was instrumental in helping us acquire our own Atari 2600 for Christmas in 1981. Thanks to Dad for purchasing the Atari 800 as a 1983 Christmas present, without which today we can only imagine that Steve would be the guitarist and I would be the bass player in a struggling retro punk band making terrible movies. Thanks to Mom for tolerating us reading the book *Dr. C. Wacko's Miracle Guide to Designing and Programming Your Own Atari Computer Arcade Games* before, during, and after church and for waiting long hours in the car as we copied public domain libraries from SBACE user group software library. Finally, love and thanks go to our wives, Dawn and Jeanne, for putting up with this process and supporting us unconditionally along the way and to our children, Rachel, Daphnie, Ryan, Kaitlyn, and Justin, for being the reason we work as hard as we do.

Preface

We are twin brothers who were born right at the beginning of the 1970s just about the same time the first video games were being created and marketed by people like Nolan Bushnell at Atari and Ralph Baer for Magnavox. While we did not know of these video game advances at the time, something exciting was obviously in the air in those years. As far back as we can remember, we have wanted to make our own games. We grew up just like most suburban kids of the 1970s—riding bikes, playing guns and ditch 'em at the school yard, and staying out all day only to come home when the street lights came on. There was never a lot of extra money in the household, so that meant we had to find creative ways to entertain ourselves. At a very early age, we started designing games to help fill the days.

First came sports contests. We spent many days conceiving two-player versions of nearly every sport imaginable on the 100-foot driveway that adorned our 1950s tract house. Not too long after, we graduated to experimenting with our dad's surplus batteries, wires, lights, motors, and potentiometers; we were trying to make anything electronic. Through trial and error, we made electric gadgets with blinking lights, switches, and running motors, and even crude pinball machines. Soon, almost any household item had the potential to become an interactive toy or game. We spent many days creating animated flip-books out of every paperback we could find. There was one full summer in the Fulton household during which you could not read any soft-cover book without being distracted by cartoons of exploding Tie-Fighters, flying arrows and text rearranging itself running up the side of every page. Toys did not escape this frenzy either. For example, an Etch-A-Sketch became our first "hand-held game development platform." By using scotch tape to create tracks and a digital watch, we created our own simple racing games and other activities.

These uber-analog game designs might have gone on ad infinitum, but something else was on the horizon. It seemed that just as soon as we had discovered a way to create new games out of our old toys, new experiences many times more interesting suddenly arrived on the scene. First came Star Wars in 1977, which basically made everything else irrelevant. All we wanted to do after seeing that movie was recreate Star Wars, think Star Wars, and be Star Wars. Then, a year or so later, the Space Invaders coin-operated game arrived in the arcades, and we were able to actually play Star Wars—or at least a reasonable facsimile.

By 1980, if we were not in an arcade playing Asteroids or Missile Command, we were designing our own pixelated arcade games on graph paper our dad brought home from Hughes Aircraft. Soon after, we taught ourselves BASIC and, in stolen minutes on a borrowed Apple IIe computer, starting writing our own primitive text-based games. By 1984, we had our own second-hand Atari 800 computer and had taught ourselves enough BASIC programming skills to try and create more elaborate games. We made sentence generators, game show simulations, a poker game, a horse racing contest, and even a small role-playing game. By the age of 14, we had both simultaneously discovered what we wanted do when we grew up—program computer games. We felt like we were naturals at programming and nothing could stop us.

However, things do not always turn out the way you plan them. In fact, those early successes led to more difficult times ahead. It was one thing to make a little game in BASIC, but it was another thing entirely to try to create a fast-action arcade game that was fun to play. As we moved from an 8-bit Atari 800 to a 16-bit Atari ST, programming only became more difficult. Balancing success in high school and college classes with learning lower level computer languages became a very time-intensive process. We managed

to finish a game in 1989 with the STOS game creator for the ST. We made our first finished, compiled game—Zamboozal Poker Dice, complete with animated sprites and digitized sound. But the joy was short-lived. STOS, while powerful, was a shortcut that could not replace the solid programming skills we had not yet developed. As college got underway in earnest, our hobby programming projects fell by the wayside and soon were mostly forgotten. We never made a second game with STOS.

Eleven years later, after many false starts making games that rarely saw the light of day, plus detours dabbling in both indie filmmaking and music, we found ourselves in a completely different situation. In that time, we had both paid some dues in the development world, working for many years developing software in Perl, C, C++, and Assembly language. At that moment, in the year 2000, we were working together at a major corporation that manufactured products for kids, developing marketing web sites in HTML and ASP. With the need to create web-based games and activities growing daily, we were introduced to Flash 5 and its new programming language named ActionScript. With ActionScript, Flash had gone from a simple animation tool with some embedded timeline events into a real client application development platform. At first, though, we had a bit of consternation about the coding of ActionScript. Most of Flash 5 was based on timeline interactions with MovieClips. To us, this felt like STOS all over again. It was a quick way to create simple games but still a shortcut around a solid grounding in software development. Still, without a better answer, we dove into Flash 5 and set off to help define the next generation of interactive games on the World Wide Web.

The games we helped make in Flash 5 were pretty crude by today's standards, but cool for the time. We made puzzle games, click-fests, dress-up games, customization activities, and even a few action contests. With the advent of Flash MX, we moved to building entire web sites using the technology. Games now included streaming audio, loadable assets, and backend integration to save customizations beyond a single user session.

However, it was with Flash MX 2004 that things really started to take-off. ActionScript 2 provided a better programming language that was only hinted at in previous versions of Flash. Flash games could be designed like real software, using design patterns and object-oriented methodologies. The games we designed and built became ever more sophisticated with features like particle effects, parallax scrolling, and customizable levels. By the time Flash 8 was released, we were using raw bitmap data, tile sheets, and more complex physics and creating multiplayer games with technologies like Electrotanks' Electrosever 3.

Even though we were able to create some really nice games, things did not get really cool until Flash CS3 was released in 2007. CS3 included a completely rewritten programming language named ActionScript 3.0, which ran many times faster than ActionScript 2. All of a sudden, things like 3D effects and true fast-action arcade games (just like the ones we wanted to make as kids) were finally within our grasp. With ActionScript 3.0, we were able to bury our concerns about Flash being a shortcut. By carefully using ActionScript 3.0, not only could we create solid products based on sound development techniques, but we could write code that was portable to other platforms if need be.

Ten years, and hundreds of Flash games and projects later, we look back and wonder how things would have turned out if we had ActionScript 3.0 available to us back in 1989. Would we have made the same choices we made to get where we are today? How many more games would we have made?

PREFACE

Flash is a great tool, but you only get out what you put into it. In this book, we show how we make games in ActionScript 3.0 using techniques that we believe strike a supportable balance between Flash technology and sound coding practices. We love Flash and the freedom it has given us to exercise our creativity. If we seem overly enthusiastic about Flash and ActionScript 3.0, it is because it helped us make our childhood dreams of making video games come true. Our hope is that we can do the same for you.

Steve Fulton and Jeff Fulton

Layout Conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout.

Important words or concepts are normally highlighted on the first appearance in **bold type**.

Code is presented in `fixed-width font`.

New or changed code is normally presented in **bold fixed-width font**.

Pseudo-code and variable input are written in *italic fixed-width font*.

Menu commands are written in the form **Menu ▶ Submenu ▶ Submenu**.

Where I want to draw your attention to something, I've highlighted it like this:

Ahem, don't say I didn't warn you.

Sometimes code won't fit on a single line in a book. Where this happens, I use an arrow like this: ➡.

This is a very, very long section of code that should be written all on the same ➡
line without a break.