


An introduction to making games with

Cocos2d & Chipmunk Physics

By Jim Range

Heyalda™

About Heyalda™

- Established in Oct 2009
- Published 39 games in the Apple App Store
- Games downloaded millions of times
- Several Heyalda games have reached the #1 ranking in the racing category and the top 10 of all games in the Apple App Store in countries such as France, Japan, Italy, Sweden, Norway and others.

About Jim

- Spent the last twelve years transforming from an electrical/firmware engineer to an ethical hacker that became an enterprise solution development consultant that also makes video games.
- Clients include Siemens, SAP, Fannie Mae, Choice Point, HCSC, Safeway, A.G. Edwards, Ingersoll Rand, Chrysler, Symantec, and several other health care and financial services companies.

Presentation Outline

Detailed tutorials on Cocos2d and Chipmunk can be found at http://heyalda.com/blog

Overview of Cocos2d

What is Cocos2d iPhone/Mac?

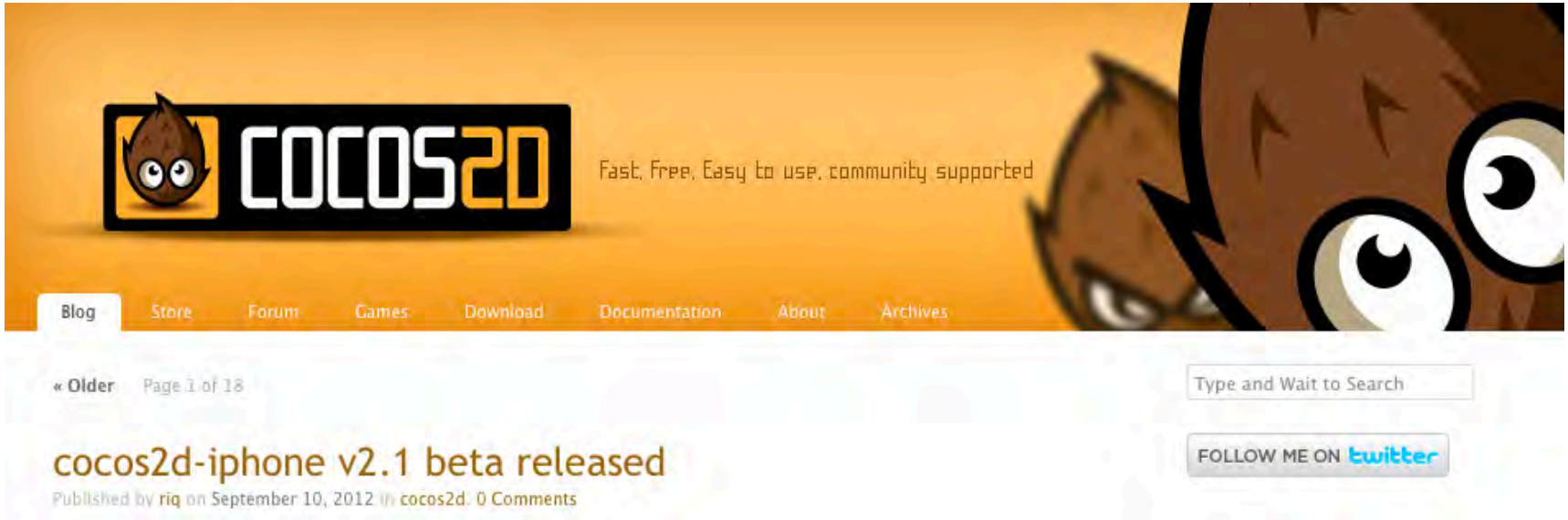
- Flexible open source Objective-C 2D sprite based game engine for iPhone/iPod/iPad and Mac http://www.cocos2d-iphone.org
- Built on OpenGL ES 1.1 (Cocos2d 1.x), OpenGL ES 2.0 (Cocos2d 2.0)
- CocosDension for audio
- Chipmunk and Box2d physics
- Lots of built in functionality that simplifies loading and using game assets.

Check out: http://heyalda.com/Getting-Started-with-Cocos2d-and-Chipmunk-Part1/

Also, for tons of iOS and Cocos2d tutorials, check out: http://www.raywenderlich.com/

Getting Cocos2d

- Download from http://www.cocos2d-iphone.org/
- Run the install-templates.sh script to install the templates into Xcode.
- Check out http://www.learn-cocos2d.com/ and Kobold2d for an alternative to directly using Cocos2d.



Important Cocos2d Classes

CCDirector – Subclass of UIViewController that is a singleton object that manages setting up OpenGL ES, the game loop, and presenting scenes.

CCNode – The base class of most viewable objects and includes properties such as position, rotation, scale, skew, etc.

Important Cocos2d Classes

CCScene – Subclass of CCNode with its content frame set to the devices screen.

Important Cocos2d Classes

CCLayer – Subclass of CCNode that adds touch and accelerometer input.

CCSpriteBatchNode – Subclass of CCNode that batches drawing sprites for efficiency.

CCSprite – Subclass of CCNode that adds a sprite texture.

Important Cocos2d Classes

CCDirector, CCScene, CCLayer, CCSpriteBatchNode, and CCSprite

CCScheduler & Game Loop

- Schedule a selector to execute.
- Can pass an interval in seconds to execute a selector at a future time. Can also unschedule a selector.

Inside the currently running CCScene:

```
-(id) init {

    self = [super init];
    if (self) {

        [self schedule:@selector(update:)];
    }
    return self;
}

-(void) update:(ccTime)dt {

    // Game Update Loop
}
```

Sprite Sheets and CCSpriteBatchNode

- Retina support is a must for most games.
- Use tools like Texture Packer or Zwoptex to create sprite sheets and add them to CCSpriteBatchNode objects.
  - Reduces draw calls if used properly
  - Reduces RAM memory usage (power of two texture memory size)
  - Reduces loading time

CCSprite and Performance

Using the CCSprite, CCSpriteBatchNode, and CCspriteFrameCache can improve rendering performance by reducing draw calls.

```
// Add sprites to the sprite frame cache
[[CCSpriteFrameCache sharedSpriteFrameCache] addSpriteFramesWithFile:@"mySpriteSheet.plist"];

// Create a sprite batch node
CCSpriteBatchNode* spriteBatchNode = [CCSpriteBatchNode batchNodeWithFile:@"mySpriteSheet.png"
    capacity:30];

// Add the sprite batch node to the node that will draw it.
[self addChild:spriteBatchNode];

// Create a sprite from the sprite sprite cache and add it to the sprite batch node
CCSprite *aSprite = [CCSprite spriteWithSpriteFrameName:@"aSprite.png"];
[spriteBatchNode addChild:aSprite];
```

Sprites, RAM, and Performance

OpenGL ES Pixel format (set in AppDelegate)

- kEAGLColorFormatRGBA8 or kEAGLColorFormatRGB565

Cocos2d texture format for PNG/BMP/TIFF/JPEG/GIF images (can change for each loaded image)

- RGBA8888, RGBA4444, RGB5_A1, RGB565

CCMenu

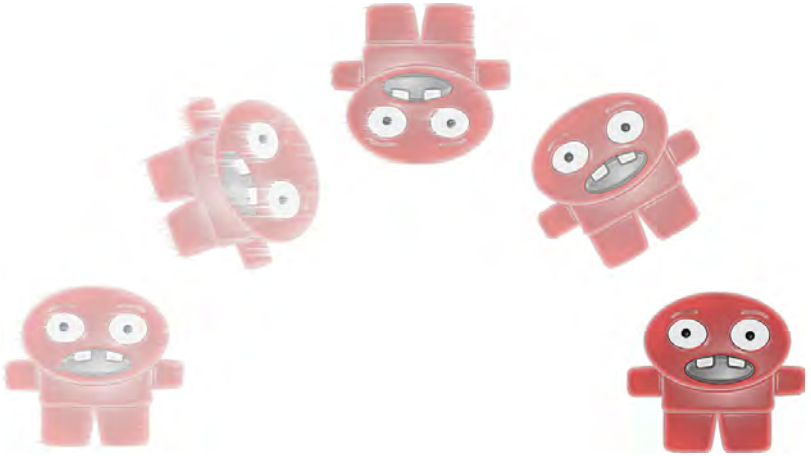
Add CCMenuItem and their subclasses to CCMenu to create a menu.

```
CCMenuItemFont* playButton = [CCMenuItemFont itemWithString:@"Play" block:^(id sender){
    // Do some stuff to start playing...
}];

CCMenu* menu = [CCMenu menuWithItems:playButton, nil];

[self addChild:menu z:0 tag:1];
```

CCAction

Performs actions on sprites such as moving, scaling, skewing, tinting, calling functions, etc.

```
id showMenuAction = [CCMoveTo actionWithDuration:0.5
                                        position:pMenuShowingPosition];

[menuNode runAction:showMenuAction ];
```



Touch Controls

Built-in to CCLayer or register a node to have

- CCTargetedTouchDelegate, or
- CCStandardTouchDelegate

```
@protocol CCTargetedTouchDelegate <NSObject>

/** Return YES to claim the touch.
 @since v0.8
 */
- (BOOL)ccTouchBegan:(UITouch *)touch withEvent:(UIEvent *)event;
@optional
// touch updates:
- (void)ccTouchMoved:(UITouch *)touch withEvent:(UIEvent *)event;
- (void)ccTouchEnded:(UITouch *)touch withEvent:(UIEvent *)event;
- (void)ccTouchCancelled:(UITouch *)touch withEvent:(UIEvent *)event;
@end

@protocol CCStandardTouchDelegate <NSObject>
@optional
- (void)ccTouchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)ccTouchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)ccTouchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)ccTouchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;
@end
```

UIAccelerometer

- CCLayer is a UIAccelerometerDelegate.
- Set self.isAccelerometerEnabled = YES;
- Add the standard UIAccelerometerDelegate methods to the CCLayer instance. Low-pass filter to smooth out values.

```
- (void)accelerometer:(UIAccelerometer*)accelerometer
        didAccelerate:(UIAcceleration*)acceleration
{

#define kFilterFactor 0.55f

    float filterFactor = kFilterFactor;

    accelX = (float)acceleration.x * filterFactor + (1- filterFactor)*previousAX;
    accelY = (float)acceleration.y * filterFactor + (1- filterFactor)*previousAY;
    accelZ = (float)acceleration.z * filterFactor + (1- filterFactor)*previousAZ;

    previousAX = accelX;
    previousAY = accelY;
    previousAZ = accelZ;

}
```

CoreMotion.framework

Alternatively use Apple's CMMotionManager from the CoreMotion.framework.

- Accelerometer
- Gyroscope
- Magnetometer
- Device Motion - processed accelerometer, gyroscope and magnetometer data combined with fusion algorithms and provides the device's attitude, rotation rate, calibrated magnetic fields, the direction of gravity, and the acceleration the user is imparting to the device.

CCParticleSystem & Particle Designer

Create interesting particle effects and add them to your game.

CCTMXTiledMap

Isometric, hexagonal and orthogonal tile maps.

- Create with Tile Map Editor
  - http://www.mapeditor.org/

Override CCNode draw: Method

Override draw method in a CCNode and do custom OpenGL ES drawing

```
-(void) draw {

    // Only draw if this class has the verticies and colors to be drawn setup and ready to be drawn.
    if (shouldDrawDynamicVerts == YES) {

        // Tell OpenGL ES 2.0 to use the shader program assigned in the init of this node.
        [self.shaderProgram use];
        [self.shaderProgram setUniformForModelViewProjectionMatrix];


        // Pass the verticies to draw to OpenGL
        glEnableVertexAttribArray(kCCVertexAttribFlag_Position);
        glVertexAttribPointer(kCCVertexAttrib_Position, 3, GL_FLOAT, GL_FALSE, 0, dynamicVerts);

        // Pass the colors of the vertices to draw to OpenGL
        glEnableVertexAttribArray(kCCVertexAttribFlag_Color);
        glVertexAttribPointer(kCCVertexAttrib_Color, 4, GL_UNSIGNED_BYTE, GL_TRUE, 0, dynamicVertColors);

        // Choose which draw mode to use.
        switch (glDrawMode) {
            case kDrawTriangleStrip:
                glDrawArrays(GL_TRIANGLE_STRIP, 0, dynamicVertCount);
                break;

            case kDrawLines:
                glDrawArrays(GL_LINE_STRIP, 0, dynamicVertCount);
                break;

            case kDrawPoints:
                glDrawArrays(GL_POINTS, 0, dynamicVertCount);
                break;

            case kDrawTriangleFan:
                glDrawArrays(GL_TRIANGLE_FAN, 0, dynamicVertCount);
                break;

            default:
                glDrawArrays(GL_TRIANGLE_STRIP, 0, dynamicVertCount);
                break;
        }
    }
}
```

Override CCNode draw: Method

Study the AppDelegate Code

The application:didFinishLaunchingWithOptions: method does the following from the default Cocos2d 2.0 template:

1. Create a UIWindow.
2. Create an OpenGL view.
3. Setup the GLView pixel format.
4. Setup OpenGL projection.
5. Create a Cocos2d CCDirector (UIViewController) and UINavigationController.
6. Setup default texture cache format.
7. Add the director to the navigation controller.
8. Add the navigation controller to the window.
9. Setup how retina and non-retain images will be loaded.
10. Create a CCScene and tell the director to run it.

Dig under the hood – and learn

It’s open source, don’t be afraid to browse the Cocos2d classes; you can learn a lot by just poking around.

Overview of Chipmunk Physics Engine

What is Chipmunk Physics?

- A open source 2D rigid body physics engine written in C http://chipmunk-physics.net
- **cpSpace** - the physics world simulation container.
- **cpBody** - uses the concept of a body that has mass and a moment of inertia to determine how each body will react/ behave based on forces.
- **cpShape** - uses the concept of shapes that are attached to bodies to manage collision detection and body interaction.
- **cpConstraint** - uses the concept of constraints that are attached to bodies to "constrain" how bodies can move relative to each other.

Overview of Chipmunk Physics Engine

Connecting the physics simulation to your game

- The physics bodies position and rotation drive sprite position and rotation.
- Can use the offset of a circle shape from the body to determine a sprites position to draw.
- Sprites can be scaled between two bodies (suspension spring).
- Be creative. The physics simulation gives you data to use however you please.

Physics Simulation Tuning

- It's a game, so realism may not be desired.
- Getting the simulation just right requires a lot of fine tuning.
- Complicated physics objects with multiple bodies and constraints require accurate initial placement in the physics world.
- Conflicting constraints can cause strange and sporadic instability.
- For most games I have created, between 10 and 20 physics iterations per frame were required to achieve the desired behavior and stability.

Game Architecture & Game Design

Design Tips

- Play your game on a whiteboard before coding.
- Think about the basic flow of the game.
  - What happens in each scene/view?
  - Why is it fun?
  - Is it a “play in the checkout-line game”? Speed of game cycle.
  - What HUD (heads up display) input controls and feedback will be needed?
  - What menus will be needed? Can they be part of the HUD?
  - Which game assets will be required in each scene and which will not?

Game Architecture Implementation

- Apply MVC design patterns like you would do in a CocoaTouch app.
- UML or whiteboard the classes in your game before coding (simple games can be surprisingly complex).
- Build reusable modules that can be registered with a game manager/controller singleton or current running scene.
- Organize game assets on sprite sheets and avoid reloading assets unnecessarily.

Rapid Prototyping

Rapid Idea Testing

- Create a simple prototype of the game with just the core game play and beta test it.
- Refine the core game play.
- Design the complete game, menu system, options, and refactor the prototype code to fit the design.

Hard Coding Vs. Scripting

- Leveraging scripts, configuration files and databases (SQLite) can streamline building a game.
- Excessive configuration can make a code base more difficult to maintain (find a balance).
- Loading and parsing **PLIST files** or tab delimited level scripts, even when encrypted, can be lightning fast on an iPhone.
- Tab delimited **key=value scripts** loaded by a world builder to create levels at runtime. Uses script glue code.

Prototype with Network Asset Sharing

- PHP web service and ASIHTTPRequest.
- Upload/download game script files for rapid prototyping.
- Upload game generated data to server such as vehicle sprite position/rotation data to create computer racers (add binary files to resource bundle when shipping game).
- Swap out image and sound files on the device that are stored in the Documents directory.

Performance Tuning

- Use Instruments time profiler and OpenGL ES profiling to isolate problematic areas of the game.
- Frame Rate
  - 60 fps is ideal, but a consistent frame rate may be more important.
- Large images are expensive to draw.
- Tweak physics engine cycles
  - Numerical integration stability and CPU utilization

Game Center & GREE/OpenFeint

- Add more fun and competition to your game.
- Leaderboards and Achievements are fundamental in most games.
- Social aspects including challenges, real-time and turn based multi-player.

Monetization - Freemium Model

- Try before you buy (80% of Heyalda's revenue)
- IAP - be reasonable with price (total value of game)
- We are selling entertainment.
  - What should 1/5/10/30 minutes of digital media entertainment cost?
- Provide lasting value that does not leave the player regretting the transaction.

Monetization - In Game Advertising

Banner Ads (eCPM, CTR, CPC)

- iAd, AdMob, Millennial Media

Aggregators (to achieve better fill rate)

- Mobclix, AdWhirl, Mojiva, TapJoy

Integrated Advertising (can be more engaging)

- Tap.me, Kiip.me

Direct Relationship Marketing and Sponsorship (big money)

- Work with companies to add their branding into your game.

Getting Digital Media Content

- Hire someone (elance.com, etc.)
- Learn graphic design and audio DSP (books, lynda.com, free online tutorials).
- Buy premade content (pond5.com) but verify license with an attorney.
- Royalty free public domain content (risk of it not actually being public domain and hard to find what you want).

Tools

Graphics Content – Gimp, Photoshop, Illustrator, Toon Boom Animation, Maya, 3dS Max, etc.

Sprite Sheets – Zwoptex, Texture Packer, Automator

Particle Effects – Particle Designer

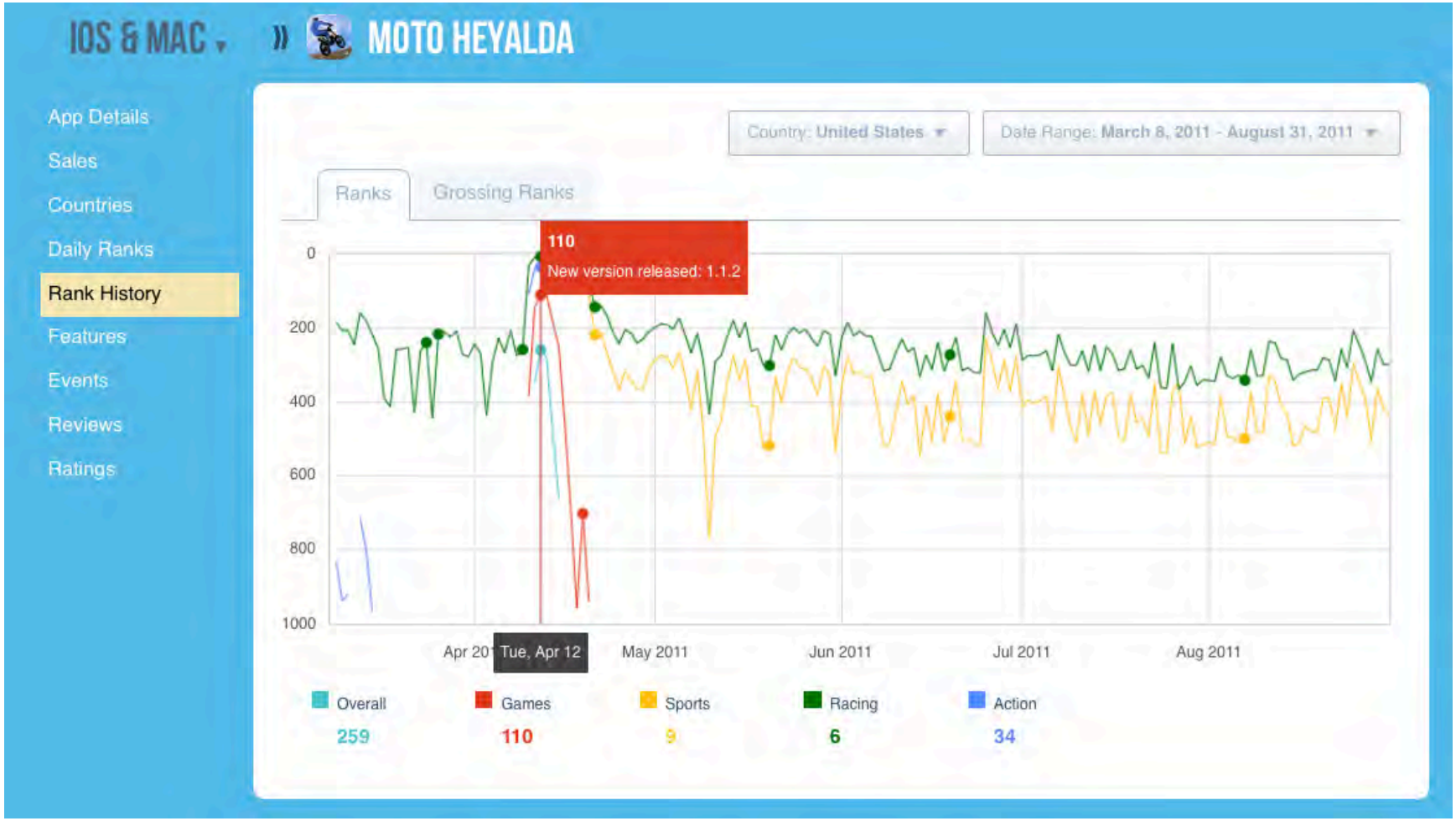
Audio – Cubase, Adobe Audition, QuickTime

Game Prototyping - CocosBuilder

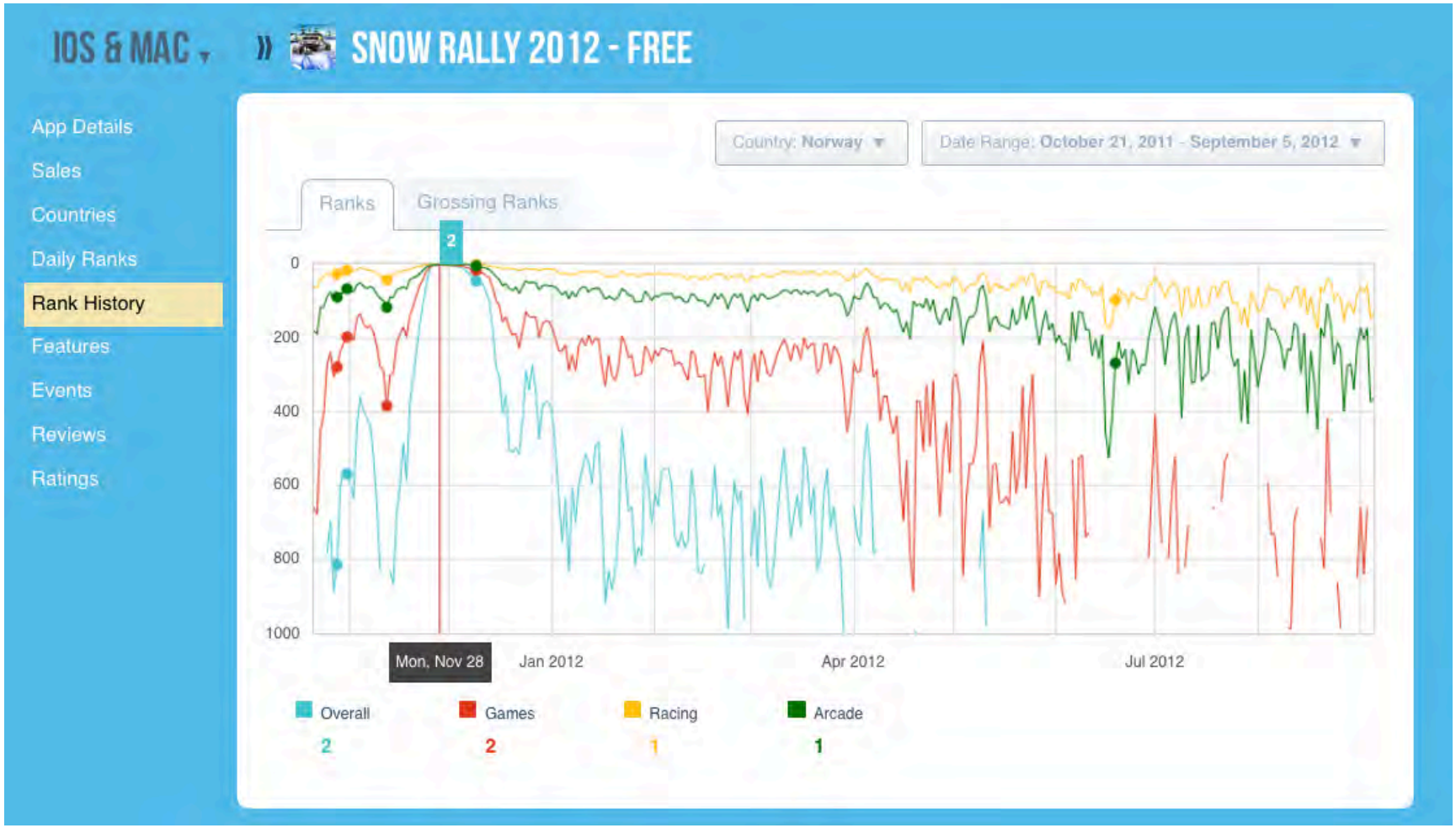
Sales and Ranking Analytics

- Download daily and weekly sales data from Apple using Apple's AutoIngestion Java tool.
  - http://heyalda.com/crunching-apple-app-sales-data-show-me-the-money/
- Signup for a service like App Annie to gain insight into sales trends and app ranking in different markets.

App Annie Analytics
Game Ranking - MotoHeyalda



App Annie Analytics
Game Ranking – Snow Rally 2012



App Annie Analytics
Game Ranking – Enduro Extreme Trials

App Annie Analytics
Game Ranking - MotoXross

App Annie Analytics
Game Ranking- SnowXross 2