Introduction to Cocos2D

Marco Bancale & Sveinn Fannar Kristjánsson

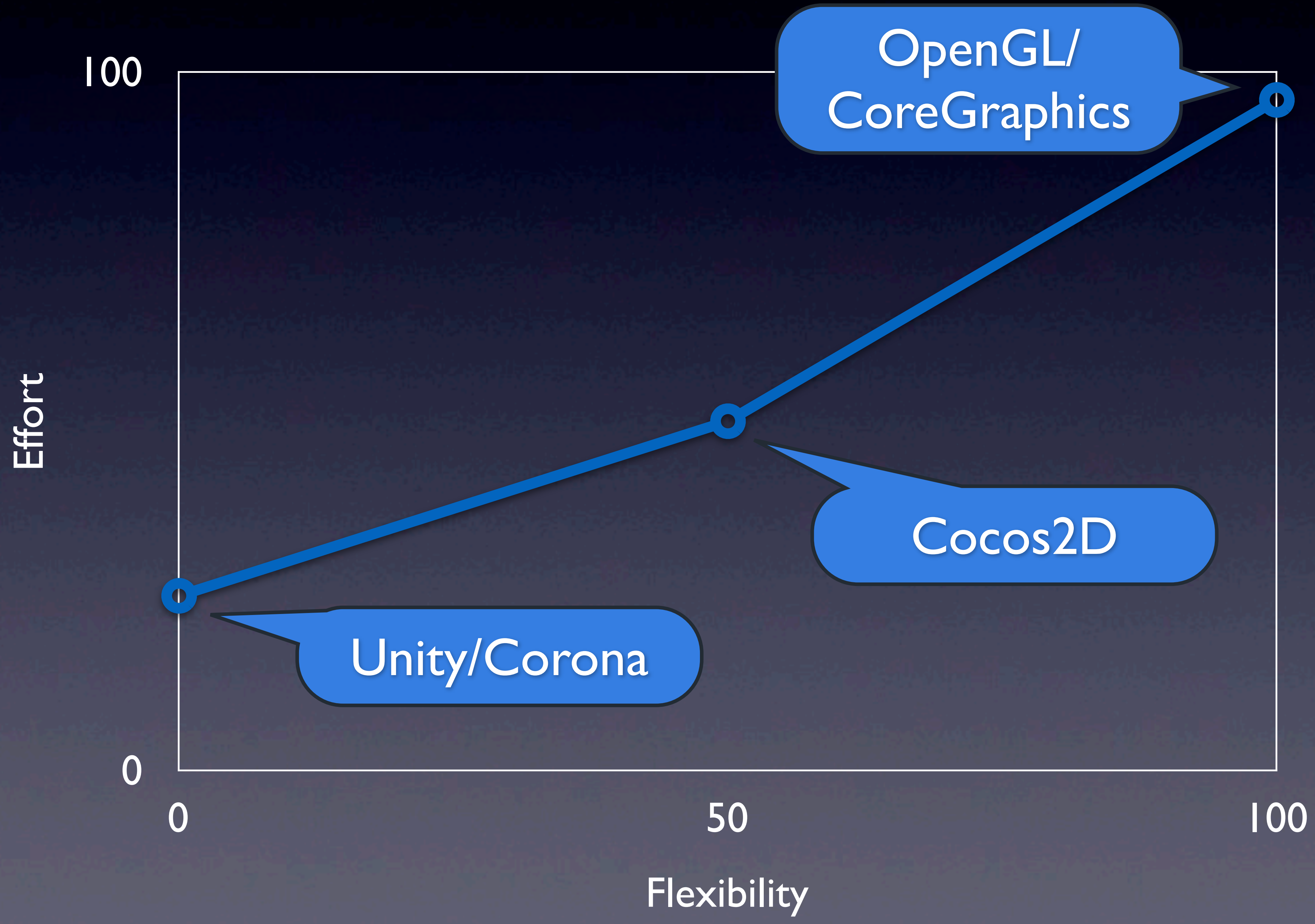
What's Cocos2D?

What's Cocos2D?

- iOS game development framework
- Open source
- Two stable versions under development:
  - 1.0.1 that uses OpenGL ES 1.1
  - 2.1 that uses OpenGL ES 2.0
- ~4.000 games shipped
- Developed by Ricardo Quesada since 2008, acquired by Zinga in 2011
- Big family: Cocos2D-Android, Cocos2D-x, Cocos2D-XNA, Cocos2D-HTML5

We will use Cocos2D 2.1 in this course.

Great Effort-Flexibility Balance



Lots of cool features!

- Easy Handling of Touches and Accelerometer
- Scene Transitions
- Parallax Scrolling
- Sprite Sheets
- Shaders
- Actions: Move, Rotate, Scale, Tint, Fade...
- Tile Map Support
- Full Screen Effects: Ripple, Wave, Lens...
- Integrated Box2D and Chipmunk
- Text Rendering
- Particle Systems
- CocosDenshion
- Cross Device and Cross Resolution
- Basic UI for Menus

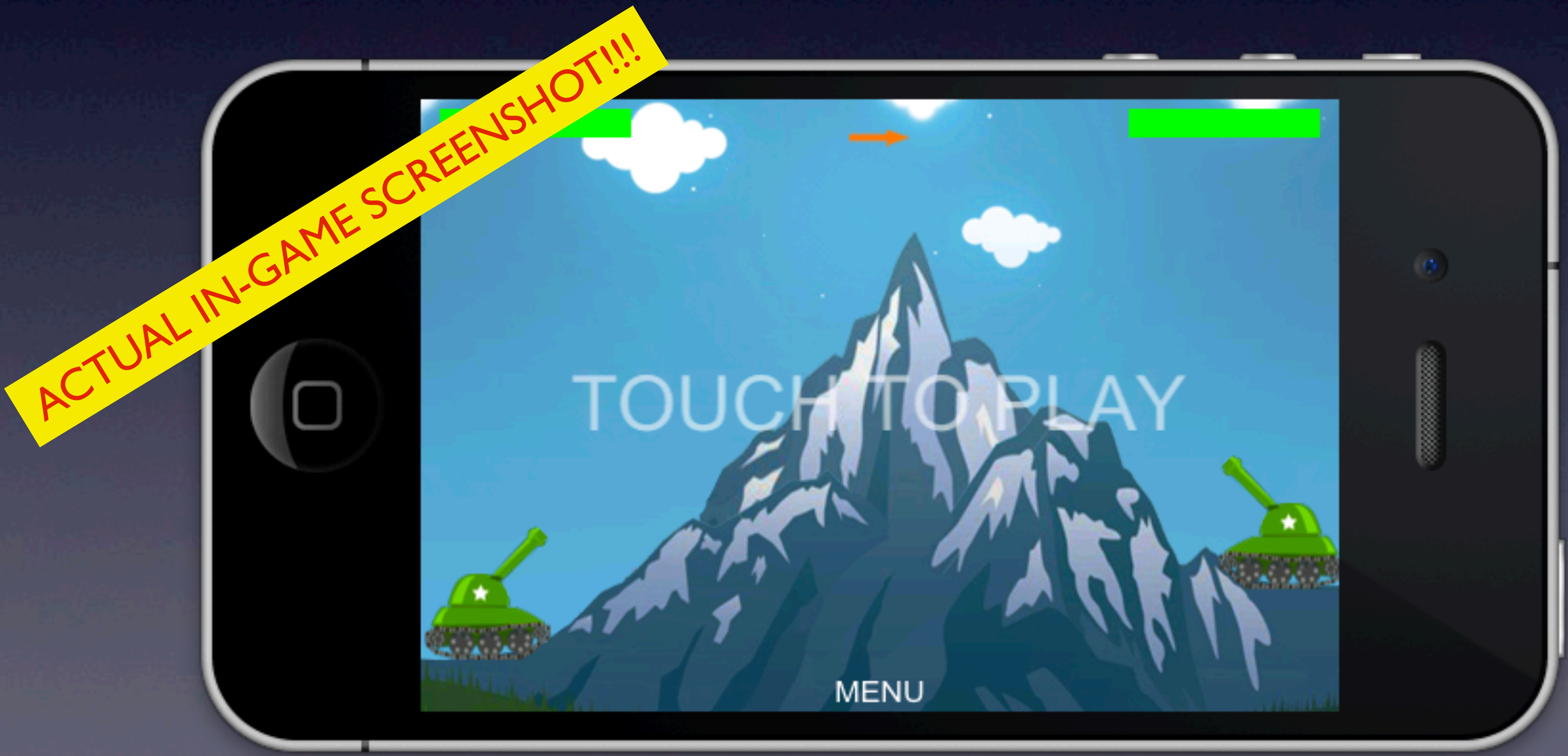
Tools supporting Cocos2D

Glyph Designer	To create bitmap fonts from TTF
Particle Designer	To design, test and share particle effects
Texture Packer	To create optimized sprite sheets
Physics Editor	To export physics data from images
Cocos Builder	To design UI

And many others...

We are going to make

Image stolen from the interwebs

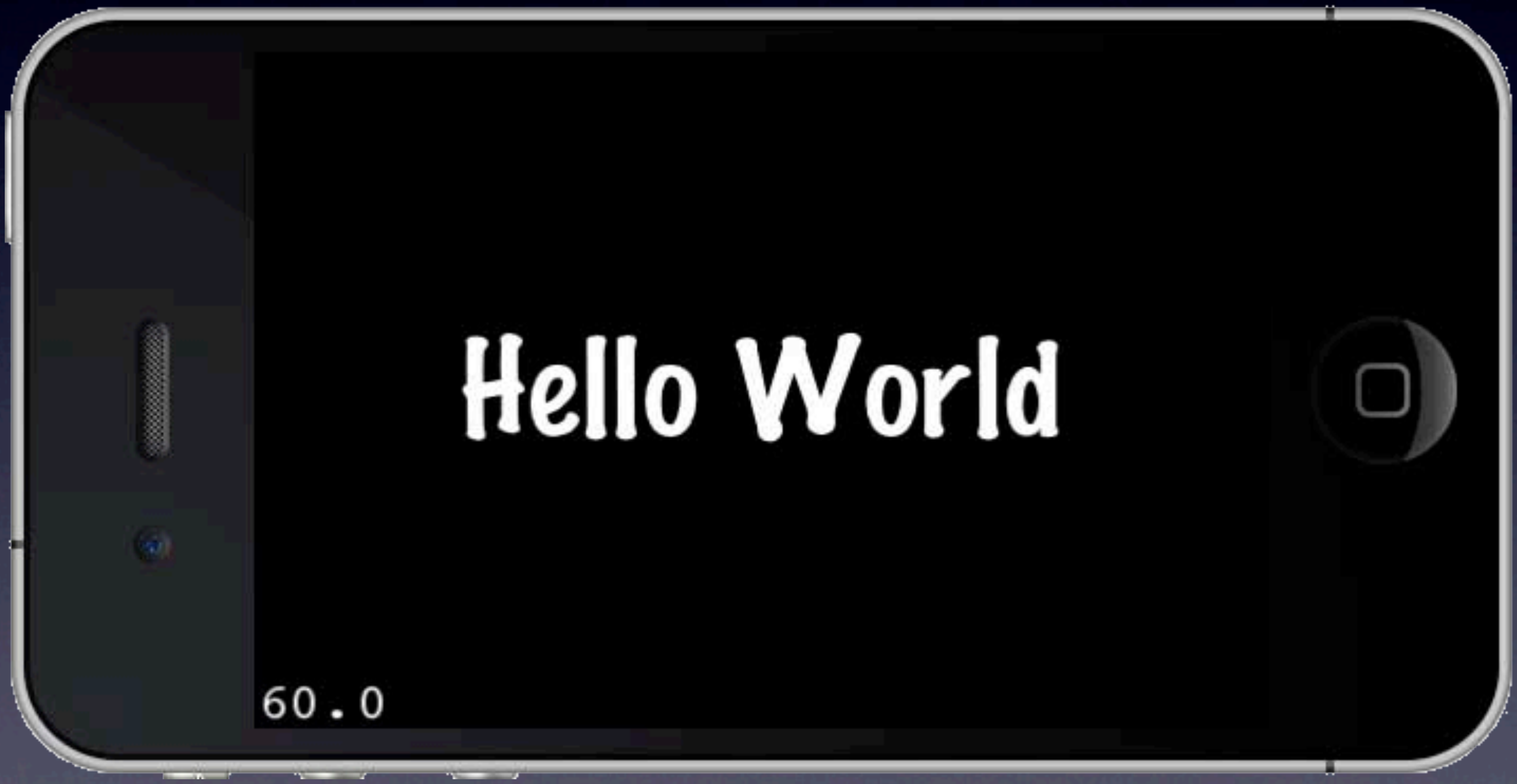


TANKS

Features:

- Turn-based tank game
- Player vs AI
- Physics based (using Chipmunk)
- Touch controls
- HUD
- Particles
- Animations via Actions
- Menus
- Scene transitions

Cocos2D Basics



The Director

CCDirector

- Initializes OpenGL ES
- It’s a singleton
- Handles a stack of Scenes (CCScene)
- Performs push, pop and replace operations on Scenes
- Can pause/resume the game loop

Nodes

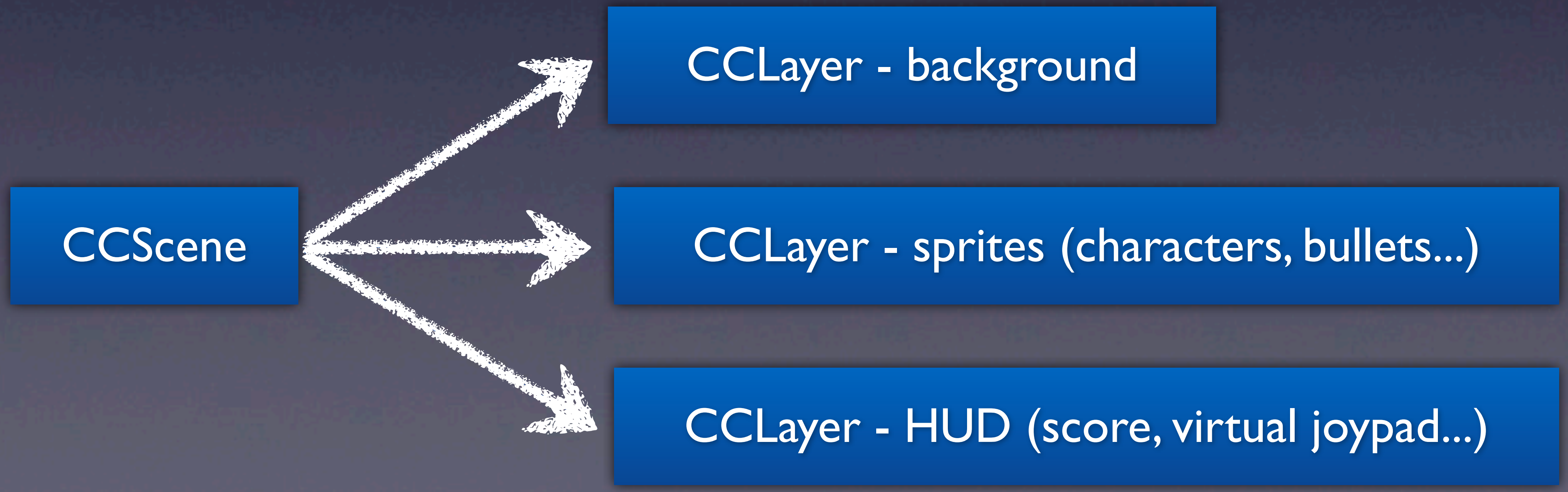
CCNode

- Contains the transformation matrix
    - Position
    - Rotation
    - Scale
- Various properties
    - Visibility
    - Anchor point
    - Content size
- Can contain child nodes
- Can schedule periodic callbacks
- Can execute Actions
- Most Cocos2D classes inherit from CCNode

Scenes and Layers

CCScene, CCLayer

- A scene represents an independent piece of the game (Intro, Main menu, Game screen, Highscore screen, etc...)
- Good place to contain game logic
- Doesn’t have a visual representation (doesn’t draw anything)
- Derives from CCNode (Actions and Transformations can be applied)
- Only one Scene can be executed at a time by the Director
- Usually creates all the Layers and handles resources
- Layers are useful to split code and define the draw order
- Some specialized Layers can draw themselves
- Layers are not enforced by the engine


CCScene
CCLayer - background
CCLayer - sprites (characters, bullets...)
CCLayer - HUD (score, virtual joypad...)

Sprites

CCSprite

- Knows how to draw itself
- Can be created from a file, a spritesheet or a texture
- Derives from CCNode (Actions and Transformations can be applied)
- Special properties
    - Flipping
    - Opacity
    - Color
- Anchor point is at (0.5, 0.5) by default

Coordinates

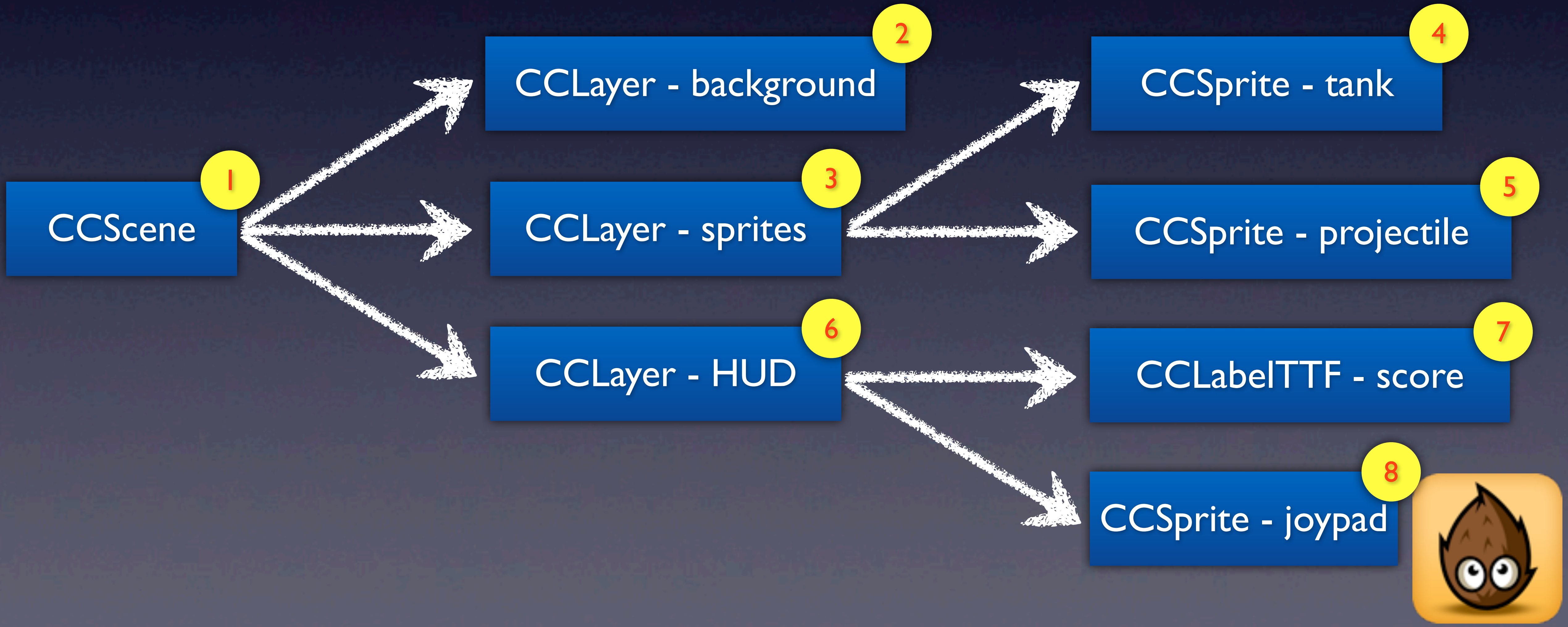
and other stuff you need to know

- Coordinates don't represent pixels, they represent *points*
- Retina and non-retina devices
    - 320x480 (iPhone 3G/3GS)
    - 640x960 (iPhone 4/4S)
    - 640x1136 (iPhone 5)
    - 1024x768 (iPad 1/2/mini)
    - 2480x1536 (iPad 3/4)
- (0, 0) is at the LEFT BOTTOM CORNER; X extends to the right and Y extends upwards
- Anchor points are in the range (0, 1)

Tying it all together

The scene graph

- Cocos2D maintains a scene graph (or scene hierarchy)
- The root of the tree is always a CCScene
- Cocos2D renders the whole scene graph every frame by traversing it with a pre-order algorithm



$$\begin{tikzcd}
 & \text{CCLayer - background (2)} & \text{CCSprite - tank (4)} \\
\text{CCScene (1)} \arrow[ur] \arrow[r] \arrow[dr] & \text{CCLayer - sprites (3)} \arrow[ur] \arrow[r] & \text{CCSprite - projectile (5)} \\
 & \text{CCLayer - HUD (6)} \arrow[r] \arrow[dr] & \text{CCLabelTTF - score (7)} \\
 & & \text{CCSprite - joypad (8)}
\end{tikzcd}$$

Let’s start with

Image stolen from the interwebs

Part 1

GOAL:
The game shows the background and the mountain

Steps needed:

- Set up the project
- Add assets
- Create a GameScene
- Add background and mountain sprites to the scene

Property List files

where to store your game data

- .plist are XML files
- Extensively used on Mac OS X and iOS
- Supported by Foundation and many of its classes
- NSArray, NSDictionary are automatically serialized and deserialized
- Same for NSString, NSNumber, NSValue, NSData and many other classes
- Application bundles

IT'S LIKE MAGIC!

Part 2

GOAL:
The two tanks with turrets are on screen

Steps needed:

- Design and create a Tank class
- Add assets
- Turret is part of the Tank
- Add two tanks to the scene

Actions

Subclasses of CCAction

- Can be performed on any class that inherits from CCNode
- Use the “fire and forget” approach

Basic	Move, Scale, Rotate, Bezier, Hide, Fade, Tint...
Composition	Sequence, Repeat, RepeatForever...
Ease	EaseExponential, EaseSine, EaseBounce...
Effects	Lens, Liquid, Ripple, Shaky...
Special	CallFunc, CallBlock, Follow...

Part 3

GOAL:
Tanks can move their turrets

Steps needed:

- Tank class exposes simple methods to move turret
- Let's have fun with Actions!

Touches

CCTouchAllAtOnceDelegate and CCTouchOneByOneDelegate

- Two ways of handling touches
    - “All at once”
    - “One by one”
- Layers are ready for touch handling
- Custom classes must conform to either CCTouchAllAtOnceDelegate or CCTouchOneByOneDelegate protocols

Part 4

GOAL:
Player can control turrets' movement

Steps needed:

- Design and create a ControlsLayer class
- Handle touches