Android* platform: What game engines, libraries, and APIs should I choose?

There are a lot of games available for the Android* platform. Independent developers can find it difficult to choose which tool, library, or API to use when developing a game. This article describes the best tools and engines to use for game development.

**Google Play* Games Services**

Google Play Games Services provides the Android SDK, which is equipped with all the tools and software to help developers to produce a fairly solid game. The Android SDK is intended for developers to package APIs that allow implementation with Google+ services. Because it’s a cloud-connected SDK and platform, developers can store data about players, game progress, achievements, and more in the cloud.

With Google+, developers get Google’s handy tools to make their games more social. To use game services, developers can set up the Google Play services SDK and study the game services samples to learn how to use the major components of the SDK. The SDK contains detailed documentation for Google Play game services. For quick access while developing apps, the API reference is available.

One more interesting feature is the ability to sync game data between the Web and Android games. In this way, the same game can be played on multiple platforms and data can be stored in the cloud. All Android devices from Android 2.2 that have the Google Play Store are all equipped with the Play Games capabilities.

**Unity* game engine**

The differences between platforms often mean having to use different programming languages and separate APIs, and dealing with different behaviors. Multiplatform game engines have become the go-to tool. One such game engine that is most popular among Android developers is Unity.

Unity can be used to create a game that can be used on computers, smart phones, the iPhone*, PlayStation*3, and even Xbox*. Unity provides an entire ecosystem for game development. This game development tool consists of a powerful rendering engine, fully integrated with a complete set of intuitive tools and fast workflow guides to create interactive 3D content, easy publishing on multiple platforms, and thousands of high-quality ready resources in the Asset Store.
The assets supporting the object image provided is quite diverse, ranging from the simplest 2D asset to a complex 3D one. Moreover, Unity is able to take pictures of software assets such as Autodesk 3ds Max*, Autodesk Maya*, Softimage*, Blender*, MODO*, ZBrush*, Cinema 4D*, Cheetah 3D*, Adobe Photoshop*, Adobe Fireworks*, and Allegorithmic Substance*.

The game engine also supports application development languages such as C#, UnityScript* (in the form of JavaScript*), and can be integrated with the Boo Script Python language. Games developed in Unity will be able to support multiple platforms including iOS*, Android, Windows* 8, Windows Phone* 8, BlackBerry* 10, Mac*, Windows, Linux*, Web Player*, PlayStation 3, Xbox 360, and Wii* U.


**App Game Kit**
The App Game Kit is a cross-platform game development language and library. The tools provided allow AGK Basic apps to be wirelessly broadcast to devices for testing. The App Game Kit community is very helpful, and the developers frequently publish tutorials in addition to regular documentation. Games can be developed through the AGK IDE in AGKBasic, or the libraries can be used with either C++ or Pascal. The software produced with the App Game Kit is written in a language called AGK Script. This language has powerful commands including commands for 2D graphics, physics, and networking. The commands make use of the platforms' native functions to improve performance. They are also designed to enhance code readability. The AGK Script commands have extensive online documentation. It contains many commands for OpenGL* 3D graphics and Shader deployment. One of the problems of using this tool is that it has a lot of bugs. A quick look at the release notes for each new version will quickly show that more time seems to be spent fixing issues with the existing command set than introducing new and improved features. For example, the latest version has a serious bug whereby Android apps that are placed in the background result in the app showing just a black screen. Another problem is that documentation is minimal.

**Cocos2D**
Cocos2d-x is an open source cross-platform game framework written in C++/JavaScript/Lua. It can be used to build games, apps, and other interactive programs. Cocos2d-x allows developers to exploit their existing C++, Lua, and JavaScript knowledge for cross-platform deployment into iOS, Android, Windows Phone, Mac OS X*, Windows desktop, and Linux, saving time, effort, and cost. Cocos2d-x is fast, easy to use, and loaded with powerful features.
Cocos2D-x is not only open source but also supported by Chukong Technologies of China and the United States. The framework is regularly updated, and support is regularly added for the latest technologies. 2014 has already seen the release of version 3, a new Cocos Studio development toolkit (optional), and support for new technologies like skeleton animation systems Spine* and Adobe DragonBone*. This tool supports Lua and JavaScript with full-feature support. Especially with Cocos2d-JS developers can develop games—cross-web and native—and the native solutions have great performance with JS Bindings, much better than with using a hybrid solution. Unfortunately this tool is not popular among Android developers, so users won’t be able to find a lot of games in Google Play that use this engine.

**Monkey* X Pro**
The Monkey engine is a next-generation games programming language. Developers can create apps on multiple platforms with great ease. The engine works by translating Monkey code to one of a different number of languages at compile time, including C++, C#, Java*, JavaScript, and ActionScript*. It is possible to write code once for multiple platforms, including iOS, Android, Windows Phone, HTML5, Flash*, Windows, OS X*, Linux, and more. Develop using Windows, OS X, or Linux.

Monkey X has a selection of great built-in modules: graphics, audio, input, data and file systems, networking, math, text and strings, collections, and online services.

Developers are not restricted to only the modules they get from the official release. They can even build an "app" module. It feels limitless. In comparison to other cross-platform solutions, with Monkey X developers actually get the translated source code, which they can play with.

Monkey is an easy-to-learn language that's object-oriented, modular, statically typed, and garbage collected. Language features include classes, inheritance, generics, interfaces, reflection, exceptions, pre-processor directives, and native code support.

As an App Game Kit, this tool has poor documentation. The documentation contains a reasonably detailed language overview and a somewhat generated list of the included modules, classes, and methods. Module descriptions are rather lax, but usually present. Method descriptions tend to be short, and a majority of them contain no usage snippets; most parameters have minimal descriptions. And besides GitHub, there are no community collaboration features to help improve it.

**Godot***
Godot is a fully featured, open source, MIT-licensed game engine. It focuses on having great tools and a visually oriented workflow that can export to PC, mobile, and Web platforms with no hassle. The editor, language, and APIs are feature rich, yet simple to learn, allowing developers to become productive in a matter of hours.
Godot has its own scripting language called GDscript. The scripting language is easy to learn with a Python-like format, but it is not Python. Rather, it is a mix of JavaScript, PHP, and C++. It's very powerful, and it's free of unnecessary things because it's designed for one purpose.

It can be used to add custom behaviors to any object by extending it with scripting, using the built-in editor with syntax highlighting and code completion.

A built-in debugger with breakpoints and stepping can be used and graphs for possible bottlenecks can be checked.

**Conclusion**

In this article we described several engines and tools for game development. All of the tools are powerful. For fast development of mobile games, developers should choose the tool that is easiest to use. Developers also need to determine the tool that can best meet the needs of their tasks.

**Related Articles and Resources**

- Official site and documentation of Cocos2d-x
- Development and Optimization for NDK-based Android* Game Application on Platforms based on Intel® Architecture
- Game Engines for Android*

**About the Author**

Vitaliy Kalinin works in the Software and Services Group at Intel Corporation. He is a PhD student at Lobachevsky State University in Nizhny Novgorod, Russia. He has a bachelor’s degree in economics and mathematics and a master’s degree in applied economics and informatics. His main interest is mobile technologies and game development.

**Notices**

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.
The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

© 2015 Intel Corporation.