

CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT

Roman Khandozhenko

CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT

Roman Khandozhenko

Bachelor's thesis

Autumn 2014

Business Information Technology

Oulu University of Applied Sciences

ABSTRACT

Author: Roman Khandozhenko

Title Of Thesis: Development Of Cross-Platform Mobile Game

Supervisor: Matti Viitala

Term and Year of Completion: Autumn 2014

Number of pages: 51

The purpose of this thesis is to develop a cross-platform mobile game and to highlight main aspects of the development process. Game project does not have any commissioner company and no specific requirements. In theoretical part of this thesis main cross-platform development frameworks will be compared and main challenges of mobile development will be defined. In practical part the process of development will be described. The practical outcome of the development will be evaluated, the Glow Hockey 3D game submission to mobile app stores; user's feedback and download rates will be reviewed, and technical aspects of using cross-platform software development tools will be evaluated.

Keywords: cross-platform mobile development, Cocos2d-x, Marmalade SDK

CONTENTS

1 INTRODUCTION.....	6
2 CROSS-PLATFORM DEVELOPMENT.....	8
2.1 Mobile market overview.....	8
2.1 Cross-platform development overview.	14
3 AVAILABLE SOFTWARE SOLUTIONS.....	19
3.1 Xamarin.....	19
3.2 Corona	20
3.3 Cocos2d-x	21
3.4 Unity.....	22
3.5 Marmalade SDK	22
3.6 Chosen software configuration.....	23
4 DEVELOPMENT OVERVIEW.....	25
4.1 Setting up development environment	25
4.2 Creating cross-platform project.....	28
4.3 Handling screen resolutions	31
4.4 Handling formats: sounds, graphics, fonts	34
4.5 Creating game loop.....	35
4.6 Performance optimization	37
4.7 Platform specific APIs.....	40
4.8 Handling deployment targets.....	41
4.9 Deployment	42
4.10 Submitting	45
5 CONCLUSION.....	47
6 DISCUSSIONS.....	48
REFERENCES.....	49

1 INTRODUCTION

Mobile gaming is one of the fastest growing industries in modern IT world. The expansion of smartphones and tablet devices brought portable gaming consoles to vast number of players including those who had no gaming experience on traditional gaming platforms like video game consoles and PCs. To reach this audience game developers should be able to deliver their products to all major mobile platforms. At the moment most significant shares on the market of operation systems are held by Android, iOS, Windows Phone and Blackberry. Development of native applications for each of these platforms will require separate code bases and maintenance of multiple subprojects that is equivalent to lost time and opportunities. The alternative is in the cross-platform development that helps to develop applications once and run them everywhere; in that case game will share common code and resources but will be deployed to every one of the target platforms with minor platform specific changes.

This thesis provides an overview of cross-platform mobile application development process. The theoretical part provides the information about the current situation on the market of mobile operating systems and has the overview of available solutions for mobile applications development, describes the software selection process and main challenges of multi-platform programming. The work is focused on main advantage of cross-platform development: fast delivering of application to many different operating systems, different device form factors with various hardware and software configurations, in practical part the developed application will be delivered to 4 major mobile platforms: Android, iOS, Windows Phone and Blackberry. This thesis explores another benefit of cross-platform development approach - keeping common code base and application look and feel across all platforms and devices which makes maintenance process easier. Depend on availability of software and hardware equipment application will be tested and deployed to all major app stores to gather statistics data of performance on specific devices and platforms such as Google Play, iTunes, Windows Phone store and others.

The case part describes the development process of mobile application. It provides an overview of used software and its requirements, setting up the environment and used libraries, common problems and its solutions that were used during the development.

2 CROSS-PLATFORM DEVELOPMENT

Cross-platform mobile development has become more popular approach to deliver applications to various mobile platforms. The changes on market of mobile operating systems and evolution of hardware components for devices brought many different cross-platform tools (CPTs) for developers.

2.1 Mobile market overview

Share of smartphones and tablets in mobile market keeps increasing for past years starting from release of Apple iPhone on June 29, 2007. Sales of smartphones in 2014 approaches to 1 billion of devices which is more than a half of overall mobile phone sales in 2013 surpassing the sales of feature phones. (Gartner Inc 2014, cited 2.12.2014.) Figure 1 shows the shipments of smartphone devices dynamics for the past 4 years and their share of the mobile phone users market.

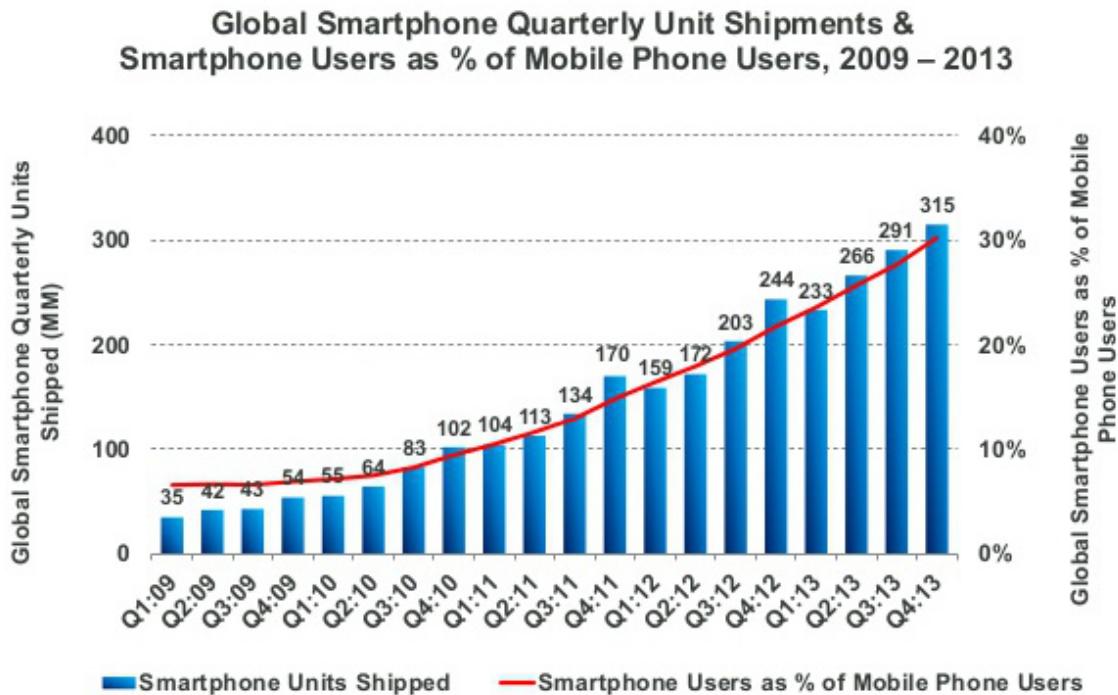


FIGURE 1. Global smartphone quarterly unit shipments (KPCB 2014, cited 3.12.2014).

Such a great number of smartphones and tablets creates a great market for mobile applications and games in particular. The most popular titles that came to mobile stores in early days of Apple's iTunes store and Google's Play Market reached tens of millions of downloads and in some cases like for Rovio's Angry Birds shows 2 billion of downloads and hundreds of millions of active players. (Future Publishing Limited 2014, cited 3.12.2014.) Even without big marketing budgets games on smartphones can reach all the players in the world and get decent number of downloads, many examples of indie games has shown great numbers of downloads like "Flappy Bird" arcade game.

But the smartphones market is very inconsistent and changes over the time: new operation systems come on the stage and some of old ones lose their share over the time. Figure 2 shows the dynamics of mobile operating systems market shares for last 5 years.

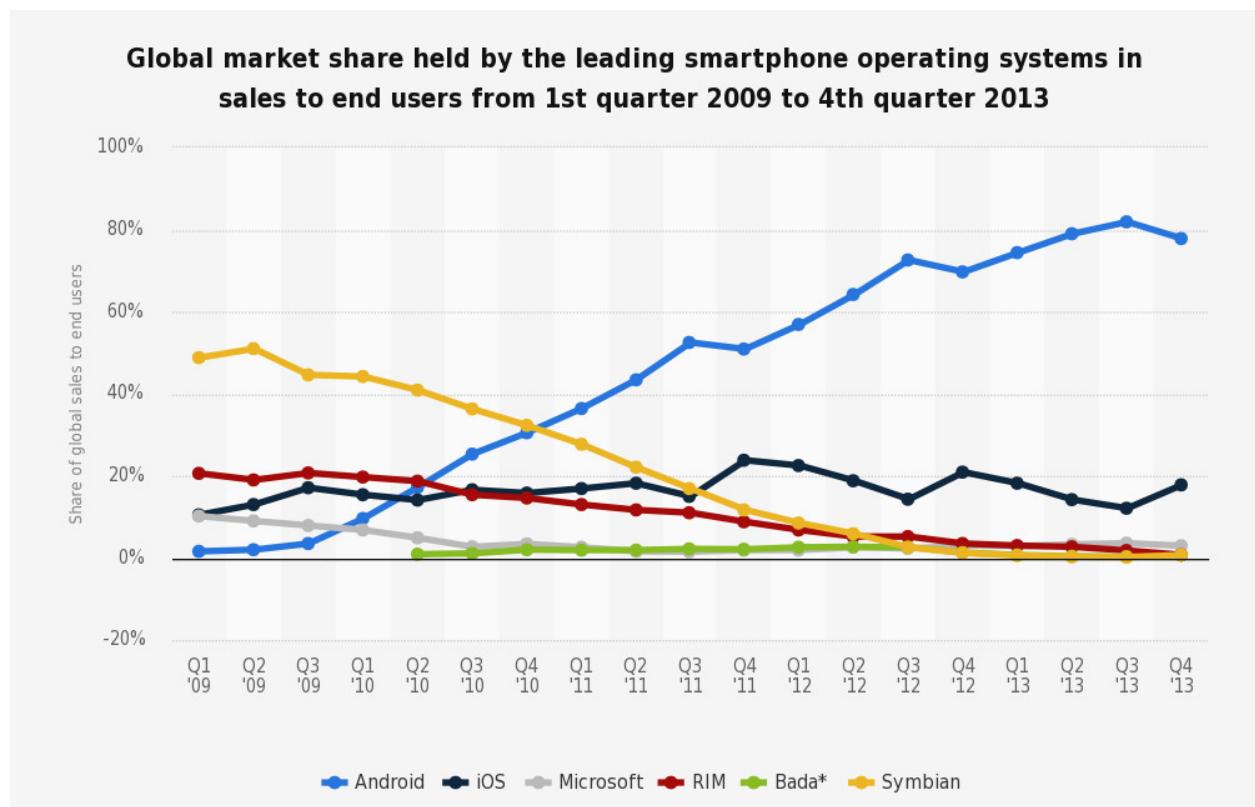


FIGURE 2, *Global market share held by the leading smartphone operating systems in sales to end users (Statista 2014, cited 3.13.2014).*

If 5 years ago game developers were concentrated on Symbian and j2me applications then nowadays these platforms have disappeared from the market. Nokia closed Symbian store in the beginning of 2014 and sold mobile department to Microsoft to start with a new Windows Phone powered devices (Resinger 2014, cited 3.12.2014). With evolution of iOS and Android operating systems and increasing number of their app stores users developers switched to most profitable and fast-growing operation systems of our days.

Target devices for games are different not only by operating system that powers it but also with a different form factors starting from small smartphones with limited computing capabilities to high end tablets with large displays and great performance. The competition of manufacturers over the 5 past years brought new players to dominate over the mobile devices market and has vanished former giants of industry. Figure 3 shows number of shipped devices by manufacturers over the last 5 years. (Cocotias 2013, cited 3.12.2014.)

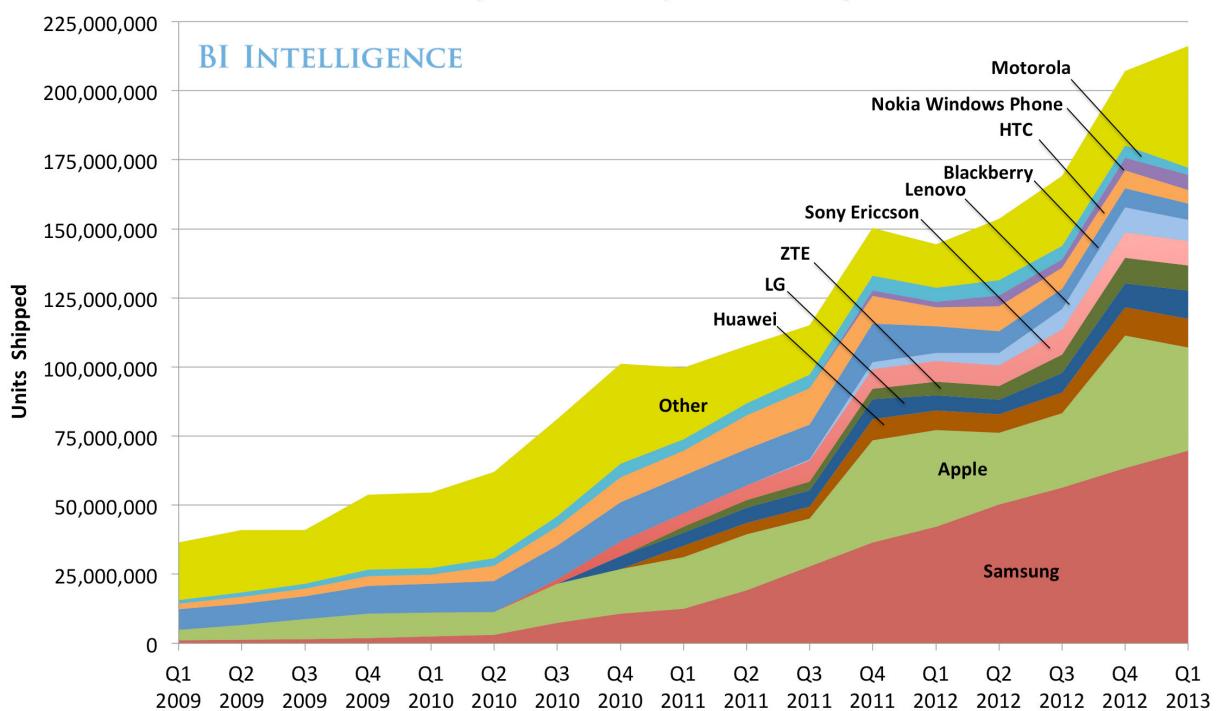


FIGURE 3. Global smartphone shipments by vendor (Cocotias 2013, cited 3.12.2014).

The diversity of operating systems, device form factors, hardware manufacturers and app stores forced game developers to choose the most prioritized targets to deliver games. The most common

approach is to develop game for the most profitable market at first and to port title to other platforms later.

For many years developers built their applications mainly for iOS platform and often ignored other markets or ported titles to Android or Windows Phone with delays. But with growth of Android market share and increase of the revenues from Google Play store by almost 700% from January to December of 2012 situation has changed (H. phonearena.com 2014, cited 3.12.2014). Figure 4 shows the progress of Google Play store compare to iTunes App Store and growth of developers' interest towards the Android market.

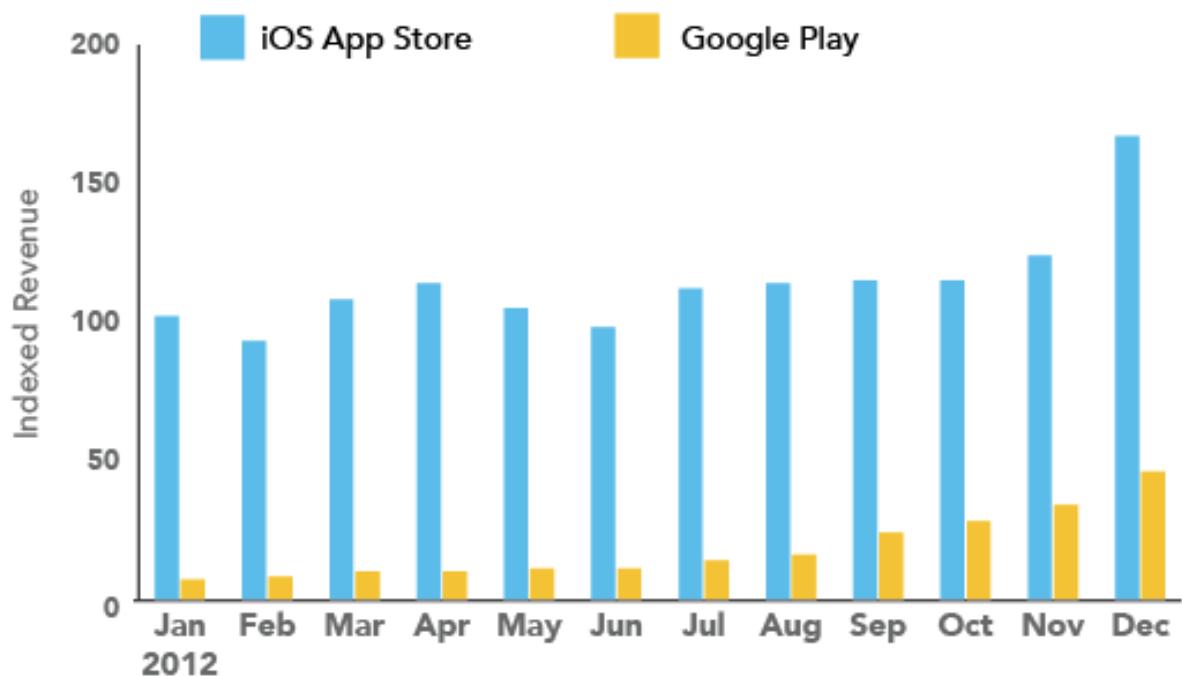


FIGURE 4. Google Play and iTunes app revenue (H. Phonearena.com 2013, cited 3.12.2013).

Apple's iTunes store is still the most profitable place to distribute mobile applications compare to Google Play and stores for Windows Phone and Blackberry; it gives more revenue per user than all other stores together, but the size of Android audience compensates the gap (Asay 2014, cited 4.12.2014). Figure 5 shows the average revenue per application for the main mobile operating systems, developers often prioritize Apple's app store for great number of users who are willing to pay for applications and low level of piracy because of software protection for iOS devices.

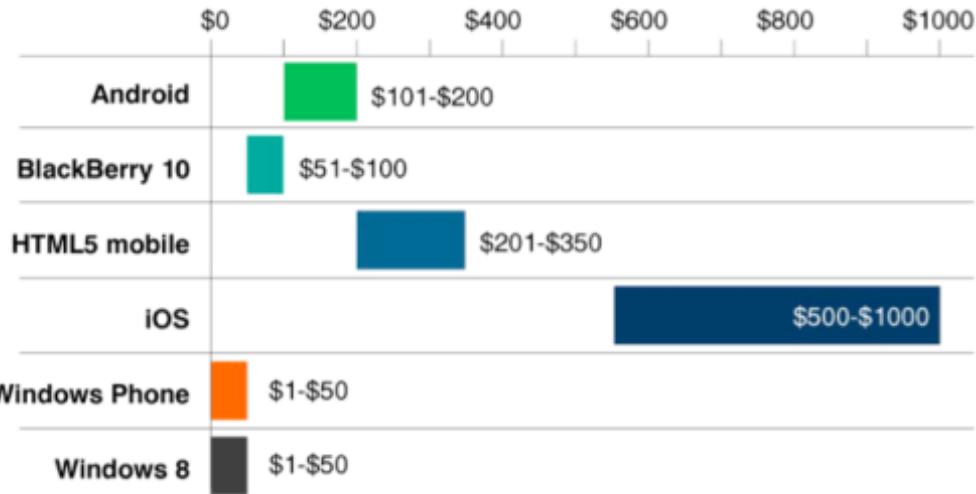


FIGURE 5. Median revenue per app Q1, 2014 (Asay 2014, cited 4.12.2014).

To maximize revenues and reach as many players as possible game developers should target their titles to all major mobile platforms including Windows Phone and Blackberry. Availability of game on several platforms becomes a standard; big titles are coming out to all platforms at once. In some cases developers are asked to keep titles as an exclusives to some platform in return for extra promotion inside the app store. For example Apple's iTunes pushed "Plants vs. Zombies 2" with premium placement in iTunes, Windows Phone store has "windows phone exclusives" section to promote apps available only for Windows Phone operating system. (Dormehl 2014, cited 4.12.2014.) Figure 6 shows that in general developer's loyalty is split between iOS and Android as a first choice platform to deliver application (Voskglou 2014, cited 4.12.2014).

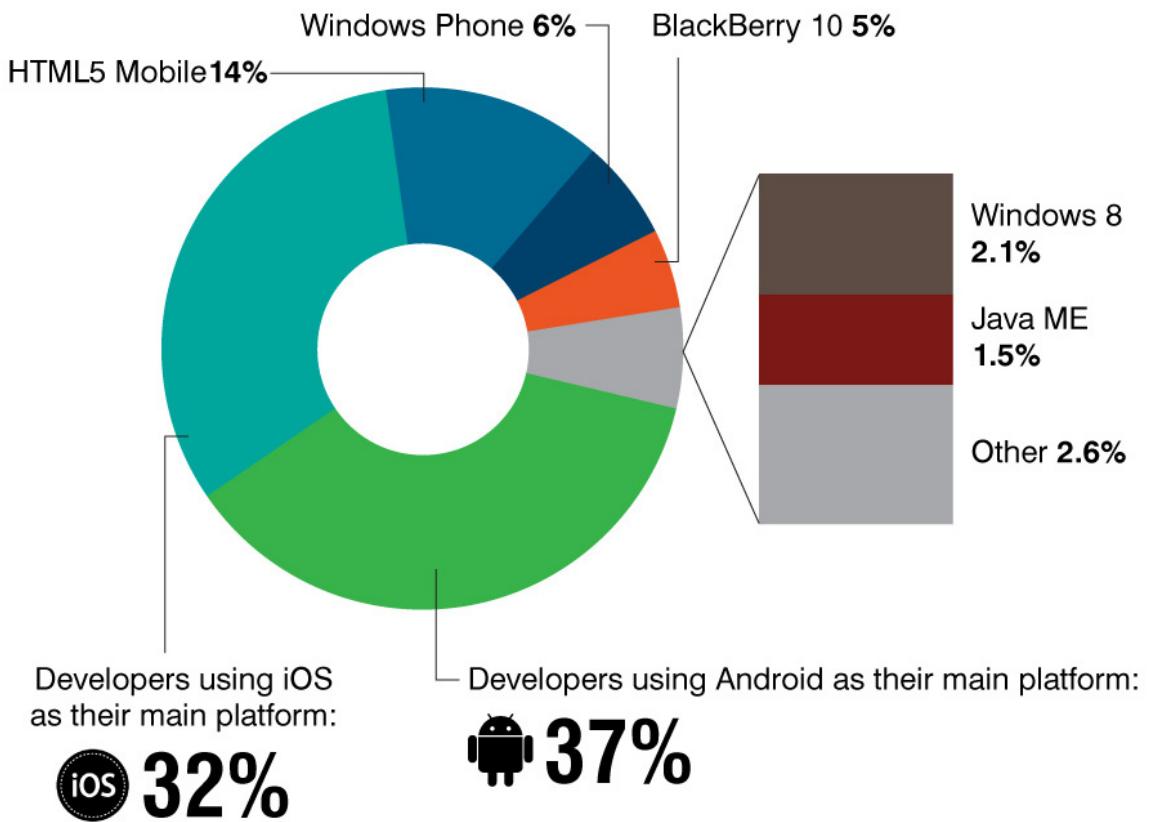


FIGURE 6. First priority platform to deliver application for developers Q1, 2014 (Voskglou 2014, cited 4.12.2014).

Delivering game to multi platforms is no longer a question of profitability but more a technical challenge. Every platform has its own programming language for native development, development tools and environments, limitations for applications running on devices. To build application for iOS developer should use Mac and Xcode IDE, for Windows Phone it should be Windows 8 64bit OS and Windows Phone SDK exclusively. As result game developers often build several standalone projects in parallel that have the same functionality but does not share any common codebase. This approach increases cost of development and maintenance, brings delays in publishing titles for some platforms. Figure 7 shows the common set of available options to download mobile game for any possible operating system or App Store.



FIGURE 7. Download the game from the list of available stores.

2.1 Cross-platform development overview.

The alternative for developing application natively for every platform is in using cross-platform development tools (CPTs) that allow using shared codebase across all target platforms with little modifications. The main advantage of cross-platform development approach is in saving time and resources on development and maintenance phases, using single development environment and programming language that is abstract from the platform native programming languages. Figure 8 shows the share of native developed applications and use of programming languages for different operating systems. (Vision Mobile Limited 2014, cited 4.12.2014.)

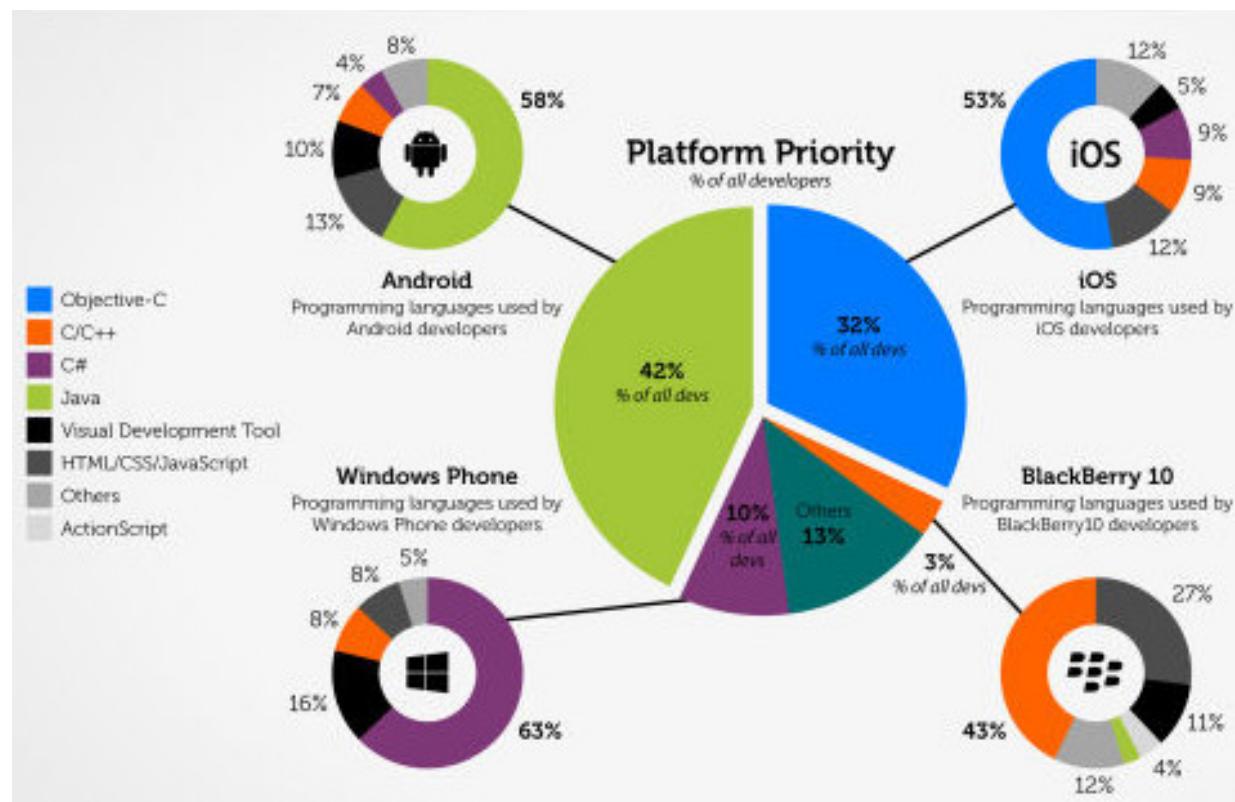


FIGURE 8. Primary language use by platform, Q3, 2014 (Vision Mobile Limited 2014, cited 4.12.2014).

With increasing demand of bringing game to multiple platforms CPTs become more popular choice for developers. But with increasing popularity of CPTs more new tools became available on the market starting from simple drag and drop app builders based on HTML5 to more advanced 3D game development engines. Selection of CPT should be based on application specifications and available resources for development, some tools are open source and free to use and other are proprietary licensed software kits. Also CPTs are based on different technologies and various programming languages, different environments and hardware requirements. At the moment more than 150 tools are available on the market and in some cases it can be difficult to choose the most suitable instrument to invest time and resources in the familiarization with new technology. Figure 9 shows the preferences of developers in using different CPT and the criteria of selection. (Pogorzelska 2014, cited 4.12. 2014.)

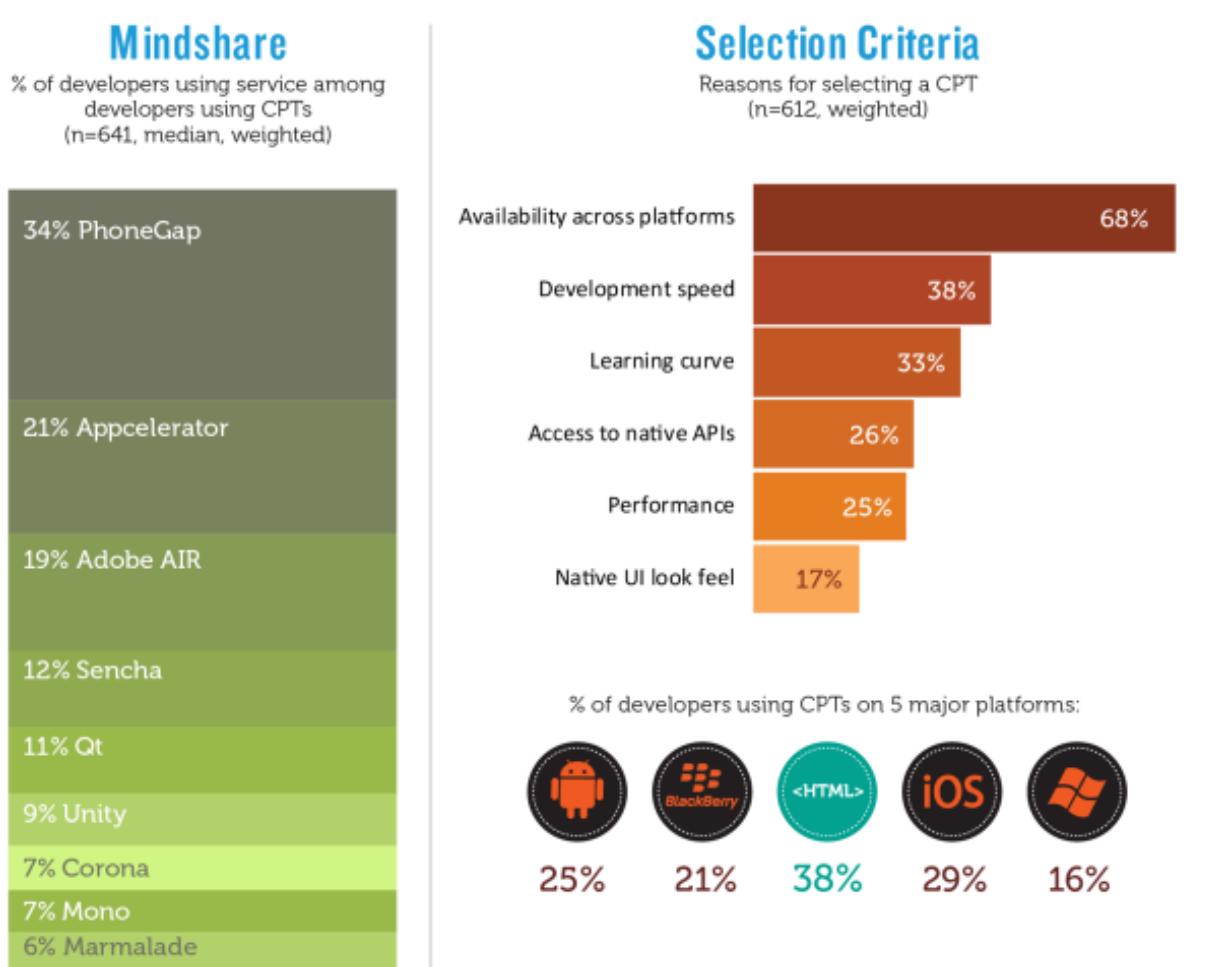


FIGURE 9. Top 9 CPTs, Q1, 2013 (Cowart 2014, cited 6.12.2014).

Fragmentation of the mobile devices becomes one of the biggest challenges for mobile developers. New versions of mobile operating systems with new features from iOS, Google, Microsoft and RIM get releases and updates on monthly basis, but not all devices receives latest software that creates situation when at the same point of time number of versions of same OS are present on the market which brings different level of available APIs and features. Figure 10 shows the fragmentation of Android operating systems with releases of new versions and remaining of devices running on previous Android versions at the same time. (Singh 2012, cited 6.12.2014.) Old versions of Android operating systems are usually running on obsolete devices which means that users are not the target audience of the application because they are most probably not willing to spend enough money and not the target group for built in advertisements.

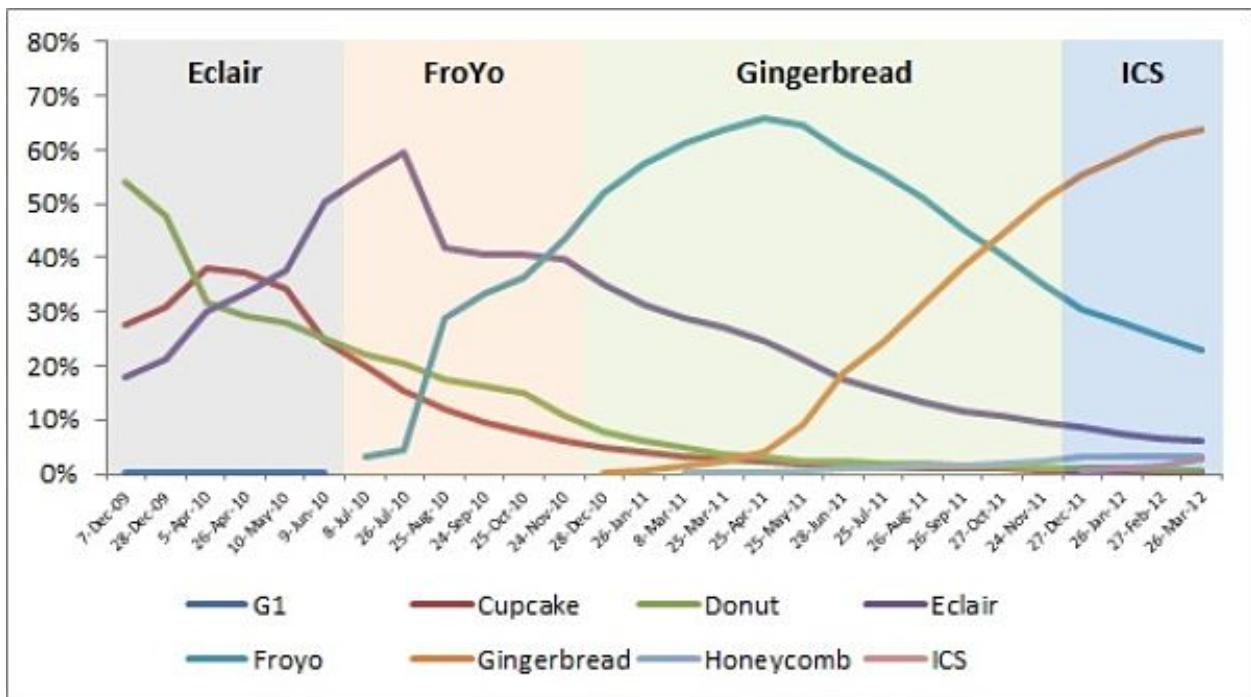


FIGURE 10. Android versions fragmentation (Singh 2012, cited 6.12.2014).

Another fragmentation issue is brought by different hardware capabilities of mobile devices. Starting from early models with very limited performance that are still in use to most advanced hi-end devices which are pushing the limits of computing power and other characteristics. Currently there are more than 18,000 distinct devices running live with unique specifications; compare to 2013 number has grown by more than 60%. Biggest hardware manufacturers are Samsung, Apple, LG, Sony, Motorola, Huawei, Nokia, Lenovo, HTC and Google, but at the same time only 10 of most popular devices represent 21% of the market. (OpenSignal Inc 2014, cited 12.12.2014.)

Hardware fragmentation is brought by endless combinations of installed hardware components and its specifications: CPU, RAM, storage, display, hardware keys, cameras, sensors and other components. Limitations of small storage size is cutting the possible quality of assets used in application, computing capabilities of older devices often makes it impossible to keep appropriate performance rate, presence of multi touch input can be the required functionality for some applications. Differences in screen sizes that are caused by display physical size, number of pixels and aspect ratios can change the look of application, it should fit to any of possible display type starting from square display of Blackberry Q10 to ultra wide screen of iPhone 6. Figure 11 shows the

most common smartphones display sizes that cross-platform application should be able to fit on.
(OpenSignal Inc 2014, cited 12.12.2014.)

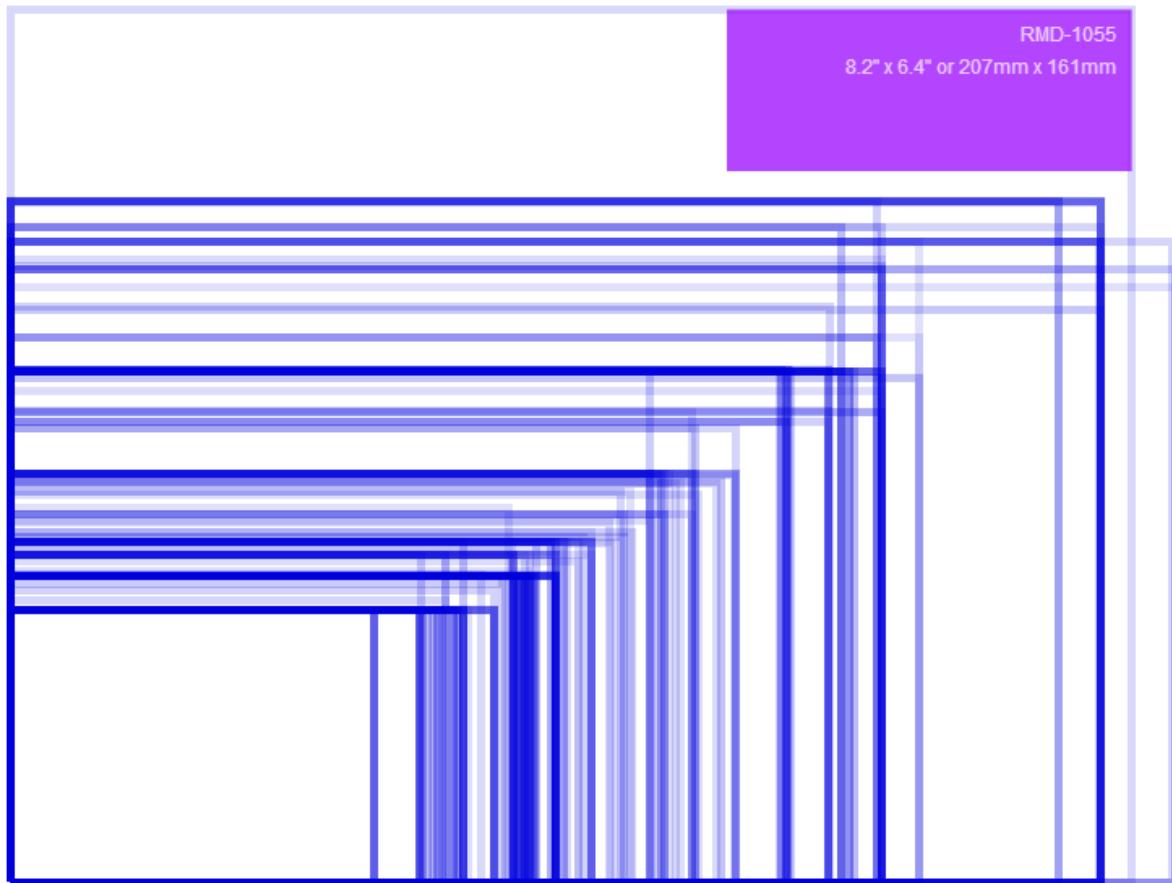


FIGURE 11. Range of screen sizes for modern smartphones (OpenSignal Inc 2014, cited 12.12.2014).

All these challenges of cross-platform development and capabilities of modern CPTs should be taken into account at the stage of planning the project specifications and choosing the CPT.

3 AVAILABLE SOFTWARE SOLUTIONS

This chapter gives brief overview over the 5 CPTs that are most appropriate for multi-platform mobile game development. These tools were selected because of acceptable performance and well recognition in the game development community.

3.1 Xamarin

“Xamarin.Mobile is a library that exposes a single set of APIs for accessing common mobile device functionality across iOS, Android, and Windows platforms. This increases the amount of code developers can share across mobile platforms, making mobile app development easier and faster”. (Xamarin Inc 2014, cited 8.12.2014.) Xamarin applications can share almost 90% of the common codebase across all platforms; the rest is added with the platform specific requirements. The programming language for development is C#, the environment is “Xamarin Studio” on PC or Mac. (Xamarin Inc 2014, cited 8.12.2014.)

Xamarin uses Ahead-of-Time (AOT) compiler to produce ARM binaries binding to use Objective-C libraries on iOS with access to iOS SDK, updating to the latest Apple’s API. On Android Xamarin uses just-in-time compilation with access to existing Java code and frameworks, fully supporting Android APIs. Figure 12 shows the flow of compiling and execution of the application build on iOS and Android powered devices (LeBrun 2014, cited 7.12.2014). With some limitations in functionality Xamarin could be used by individual developers for free and for organizations prices are starting from \$999 per year.



FIGURE12. Xamarin architecture (LeBrun 2014, cited 7.12.2014).

3.2 Corona

"Corona SDK is a mobile development framework for building apps and games for iOS and Android with a single code base. Corona lets developers use integrated Lua, layered on top of C++/OpenGL, to build graphic applications in Corona Editor IDE on MAC or PC. Corona platform is built on industry standards including OpenGL, OpenAL, Box2D, Facebook, SQLite". Corona SDK provides set of plugins to extend default functionality such as: ad providers, analytics, and cross-promotion, game services, monetization, social and other. Figure 13 shows the features and extensions that are available in Corona SDK. Prices for Corona SDK license are starting from \$192 per year for version with limited functionality and supported target platforms. (Corona Labs 2014, cited 6.12.2014.)



FIGURE 13. Corona SDK features (Corona Labs 2014, cited 6.12.2014).

3.3 Cocos2d-x

Cocos2d-x is a cross-platform branch of the Cocos2d 2D games development framework, figure 14 shows the rest of Cocos2d framework components and branches. Cocos2d-x is using C++ as a programming language and distributed under MIT license, the source code of the project can be modified to implement any specific functions and is free of charge to use. Cocos2d-x supports all major mobile platforms: Android, iOS, Windows Phone and Blackberry. Applications can be built on Windows in Visual Studio IDE, Mac Xcode IDE, and Linux or in custom Cocos Code IDE. This framework supports variety of editors and external tools like Cocos Studio, Level Helper, Texture Packer, Tiled editor, Particle Designer, Spine for skeleton animations, Glyph Designer and other 3rd party tools and extensions. (Cocos2d-x.org 2014, cited 11.12.2014)



FIGURE 14. Cocos2d frameworks family (Cocos2d-x.org 2014, cited 11.12.2014).

3.4 Unity

Unity is a cross-platform game creation system developed by Unity Technologies, including a game engine and integrated development environment - Unity Editor. It uses C# as one of the programming language options, the target mobile platforms are iOS, Android, Windows Phone and Blackberry. One of the Unity feature for mobile platforms is platform tweaks: “Keep your code working across all the platforms and many devices. For platform specific tweaks, do runtime platform checks, employ code pre-processors and integrate tightly with platform specific code and plugins.” Unity has developed community and great number of tutorials, courses and items in the “Asset store” there developers can sell 3D models, editor extensions, audio and other types of content. Unity 3D has a free version with limited functionality and Pro version starting from \$1500 per year. (Unity Technologies 2014, cited 6.12.2014.)

3.5 Marmalade SDK

“Single codebase can be compiled and executed on all supported platforms, this is achieved by providing a C/C++ based API which acts as an abstraction layer for the core API of each platform. Apps created with Marmalade are compiled directly for the processor of your target device”. (Marmalade Technologied Ltd 2014, cited 4.12.2014.) Target platforms for mobile builds are iOS, Android, Windows Phone, Blackberry and Tizen, figure 15 shows the deployment targets for Marmalade SDK (Anjum 2014, cited 4.12.2014). The development environments are Visual Studio on PC and Xcode on Mac. Native platform functionality can be accessed via the “Extension Development Kit” (EDK) to use Android or iOS API directly. Marmalade SDK has extra tools to run game in simulator, sign applications and deploy builds to all platforms at once. Marmalade SDK has a free of charge evaluation period with limited functionality and set of available licenses starting from \$499 per year.

Marmalade **SDK**

The fastest way to build cross-platform C++ games



FIGURE 15. Marmalade SDK deployment targets (Anjum 2014, cited 4.12.2014).

3.6 Chosen software configuration

To select the most appropriate candidate from the list of presented CPTs the list of requirements was created. Because of limited resources allocated for development candidate should be free of charge to use, should have evaluation period or license for students. It should have big and active community of developers, and target as many mobile platforms as possible. Also CPT should use the programming language that was covered during OAMK studies.

After the comparison of all parameters Cocos2d-x and Marmalade SDK were chosen to start cross-platform mobile game development. They both have C++ as a main programming language, Marmalade SDK targets every major mobile platform and Cocos2d-x is free and open source with a great number of developers worldwide – more than 400,000 (Cocos2d-x 2014, cited 2.12.2014). Development environment is the same for both – Windows powered PC and “Visual Studio” IDE which is well known to OAMK students.

But the main reason to use these tools together was that game logic written in Cocos2d-x could be wrapped with Marmalade SDK and deployed to target platforms that Cocos2d-x does not support directly. Cocos2d-x as a part of Cocos2d game engine family is very mature for game development and has all necessary functionality for 2D game specifications; it has all standard tools for game developers: sprites, animations, scenes, physics engine, editors and helpers. Marmalade SDK is for lower level programming and operates with graphics directly which is not always convenient to make

game fast. Cocos2d-x is also open source project and every aspect of its behaviour can be reviewed and changed according to developer's needs. Marmalade SDK is a commercial product and all the fixes that can be done to it should be requested from the support.

4 DEVELOPMENT OVERVIEW

This section describes the practical part of thesis – overview of the development process of a small arcade game. Fast gameplay of arcade genre was chosen to evaluate the performance of application on devices with different operating systems and hardware specifications. Limited available set of software and hardware equipment defined development environment and platforms that could receive final builds, not all mobile platforms was tested during this research but game can be deployed there later. The team developed this project contained of game programmer, quality assurance specialist, graphics and sound designers.

4.1 Setting up development environment

The development process was performed on 32-bit Windows 7 powered PC with next set of software components: Visual Studio 2010 or later which can be downloaded free with Microsoft DreamSpark for students of OAMK program; it was used as a main IDE to work with C++ game source code, and also basic environment for Marmalade SDK and Cocos2d-x to launch simulators and compilers. Eclipse IDE, Android SDK and NDK was installed to deploy project for Android, game accesses native Android functionality with a Java Native Interface (JNI). To share the project state with other team members Git revision control system was installed and Bitbucket hosting service was set up to create remote project repository. Set of small programs for game development process was also installed: Texture Packer to compose sets of images into the sprite sheets, GIMP image manipulation program, Audacity sound editing tool, Tiled map editor, and others.

Latest version of Cocos2d-x can be downloaded from the official web page of Cocos2d-x project or can be checked out from the git repository as shown on figure 16. For this project latest available version 2.2.5 of Cocos2d-x was chosen.

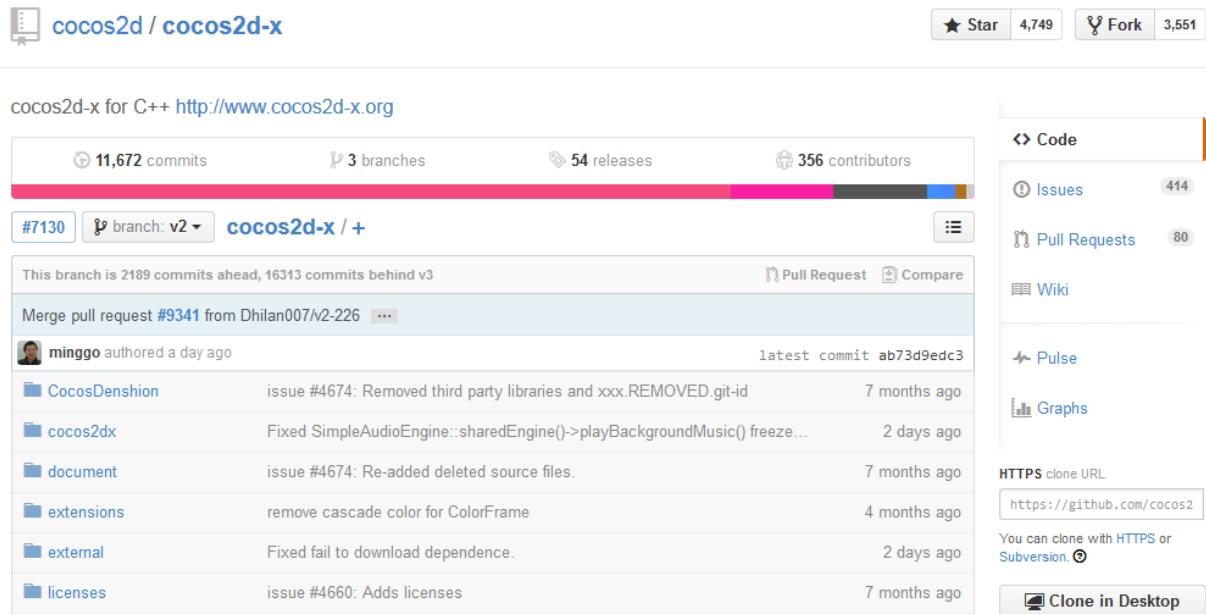


FIGURE 16. Cocos2d-x Git repository

Installing “install-templates-msvc.bat” creates a Cocos2d-x template for Visual Studio IDE that can be seen on figure 17.

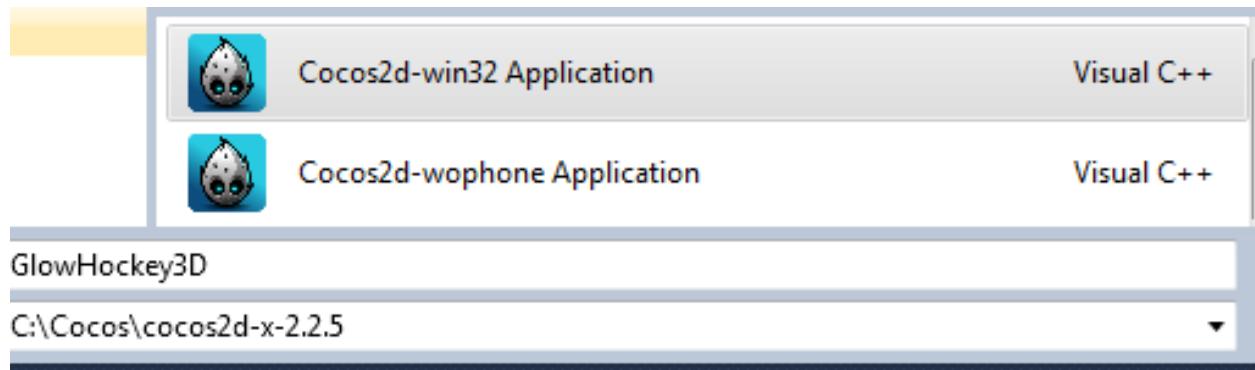


FIGURE 17. Cocos2d-x Visual Studio template

Marmalade SDK can be downloaded from <https://www.madewithmarmalade.com/download>, it should be activated for the trial period. Figure 18 shows the available options of Marmalade SDK configurations and system requirements for development environment.

Home » Download

Get started with Marmalade... for free!

System requirements

OS

- Microsoft Windows 7 or newer
- Apple Mac OS X (10.8 or higher)

Development Environments

- Microsoft Visual Studio 2008/2010/2012
- Xcode 5.0 or higher

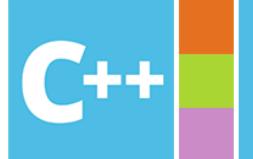
[Find out more](#)

Includes all four flavours of Marmalade:

- **C++** Maximum flexibility and performance
- **Juice** Simplify porting of iOS apps and games to Android
- **Quick** Rapid application development using Lua
- **Web** Combine the power of C++ with HTML5

And out of the box publishing to:

- iOS
- Android
- Windows Phone
- Windows Store
- BlackBerry 10
- Tizen



[Download Marmalade for Windows](#)

By downloading the SDK you accept our [privacy policy](#), [EULA](#) and [third party license terms](#).

[Download Marmalade for Mac OS X](#)
[Already have Marmalade?](#)

FIGURE 18. Marmalade SDK requirements and specifications

After installation process that is present on figure 19 is complete, all Marmalade's ".mkb" project files will be associated with Visual Studio IDE.

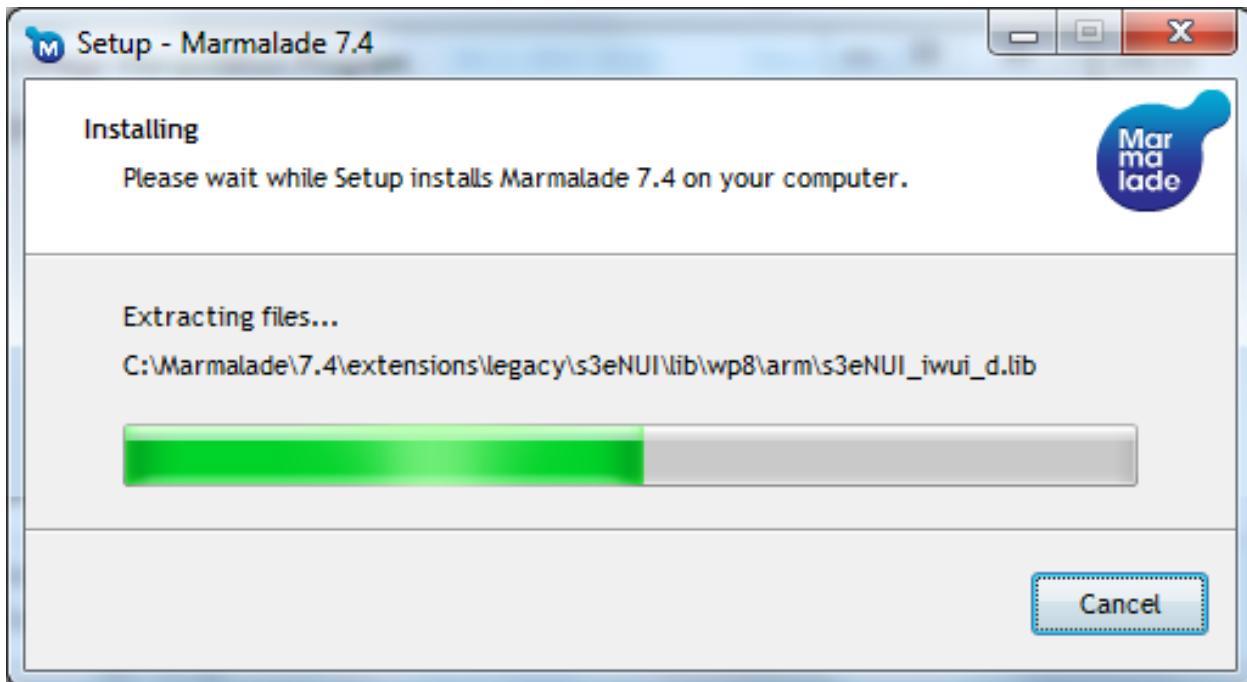
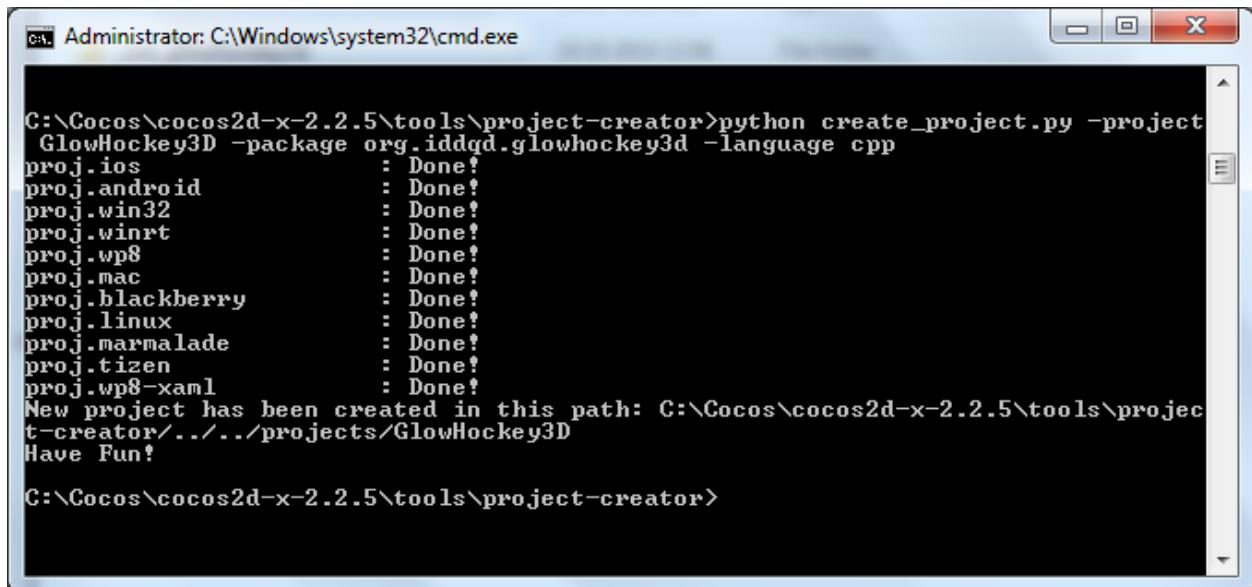


FIGURE 19. Marmalade SDK installation

4.2 Creating cross-platform project

To create a multi-platform project with Cocos2d-x “create_project.py” should be executed with set of parameters: project location, project name, package name and development programming language. The setting chosen for game are shown on figure 20, game title set to be as “Glow Hockey 3D” and programming language as C++.



The screenshot shows a Windows Command Prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The command entered is:

```
C:\Cocos\cocos2d-x-2.2.5\tools\project-creator>python create_project.py -project GlowHockey3D -package org.iddqd.glowhockey3d -language cpp
```

The output shows the creation of various platform-specific projects:

```
proj.ios          : Done!
proj.android     : Done!
proj.win32        : Done!
proj.winrt        : Done!
proj.wp8          : Done!
proj.mac          : Done!
proj.blackberry   : Done!
proj.linux         : Done!
proj.marmalade    : Done!
proj.tizen         : Done!
proj.wp8-xaml     : Done!
```

A message indicates the new project has been created:

```
New project has been created in this path: C:\Cocos\cocos2d-x-2.2.5\tools\project-creator/../../projects/GlowHockey3D
Have Fun!
```

The final prompt is:

```
C:\Cocos\cocos2d-x-2.2.5\tools\project-creator>
```

FIGURE 20. Creating cross-platform Cocos2d-x project

Script creates project folder under “projects” directory in Cocos2d-x root location. It contains subfolders “Classes” and “Resources” what will be shared and used across all platforms. Also specific to platforms sub-projects: “proj.win32”, “proj.android”, “proj.ios”, “proj.marmalade” are created as shown on figure 21.

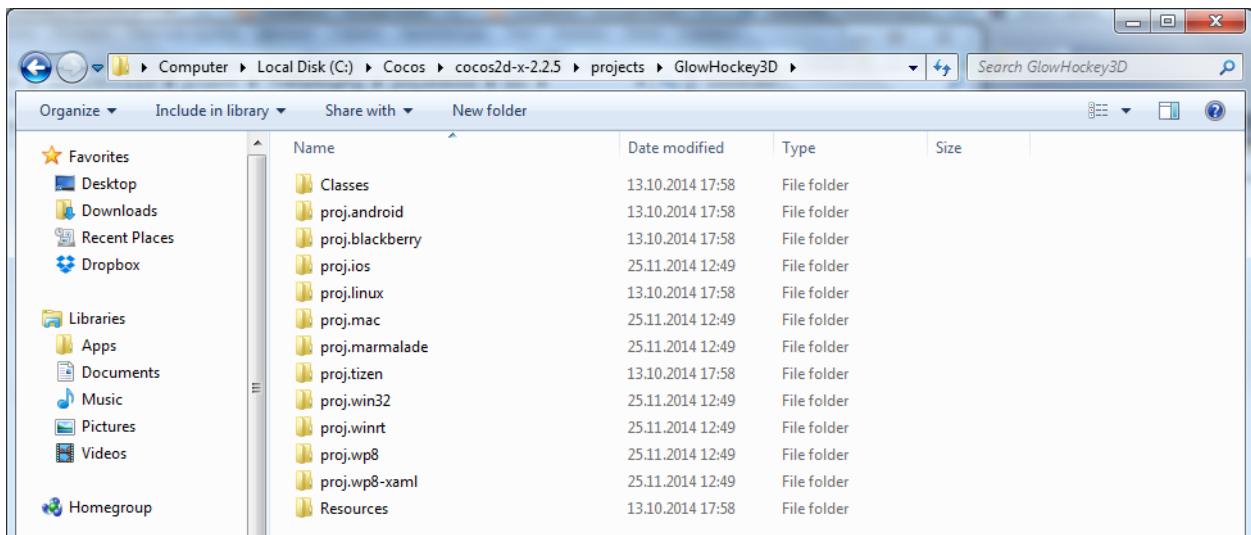


FIGURE 21. Cross-platform Cocos2d-x project structure

Cocos2d-x will be used for most of game development process and for deploying to Android platform, delivering the game to all other platforms will be done with help of “proj.marmalade” and Marmalade SDK deployment tool. The most of development time will be spent for creating the game logic and visuals that is abstract from any platform specific features and requirements. To open Cocos2d-x project in Windows Visual Studio IDE should be launched and “proj.win32” should be imported in current solution, building and running this project will open Cocos2d-x in Visual Studio IDE that can compile “proj.win32” and run built in emulator. The size of emulator surface can be changed also debug statistics data can be shown - this will be enough for development purposes. Figure 22 shows Cocos2d-x Visual Studio emulator launched with resolution of 480x320 pixels and displayed debug output in the left bottom corner.

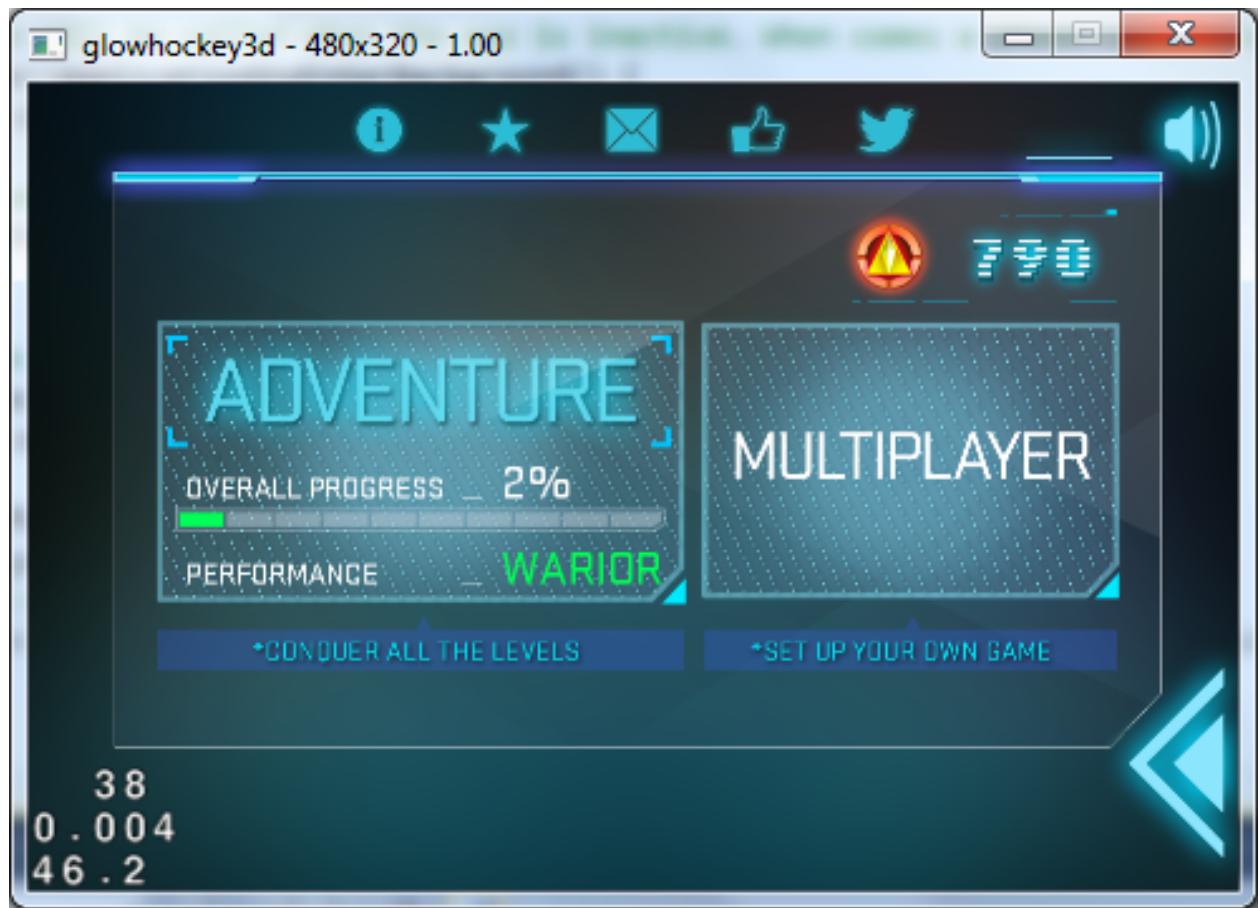


FIGURE 22. Cocos2d-x Visual Studio emulator

More advanced simulator from Marmalade SDK can be launched on Windows PC. Figure 23 shows additional options that can be applied to running Marmalade SDK simulator.

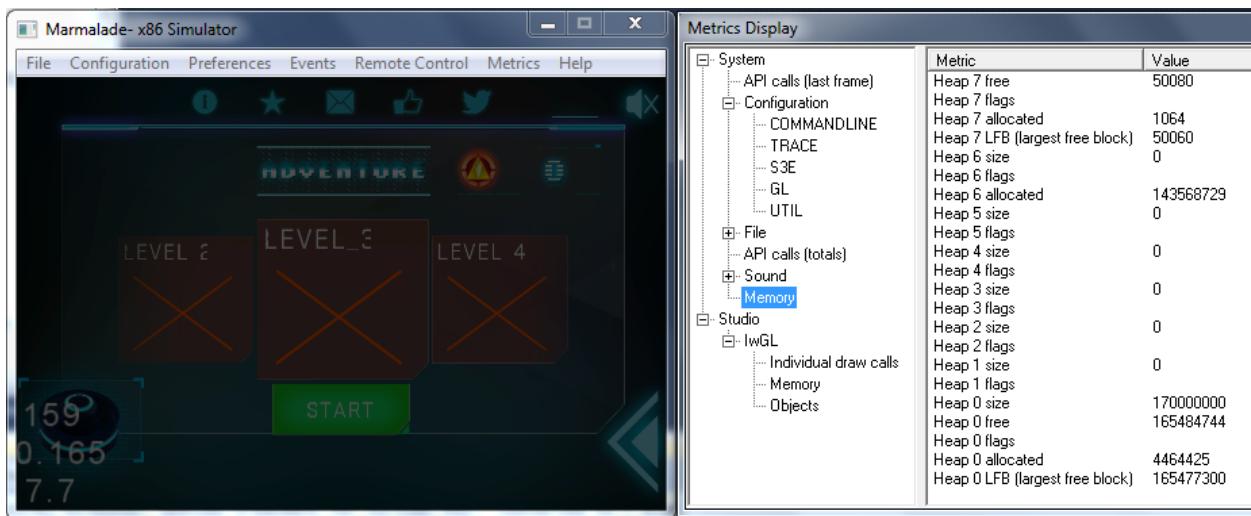


FIGURE 23. Marmalade SDK Visual Studio simulator

4.3 Handling screen resolutions

Project should be deployed to many platforms and many different devices with the wide range of screen sizes starting from small low-end smartphones to extra-large tablets, with different pixel density and different screen size ratios. Game should have proper look on all target devices, no element can be cut or displayed corrupt.

To handle different resolutions game scales up or down graphical assets to match the current resolution in a best way. The set of art was prepared for retina quality screen of iPhone 4S with screen resolution of 960x640 pixels, devices with lower resolutions will get scaled down images and devices with higher resolution will get images scaled up. Multiple sets of assets can also be used to have images in several packages to use the one that fits better for current device and avoid scaling images and getting best looking graphics on every device, but this task was not prioritized because actual results of chosen resolution manager approach satisfied quality assurance of the project.

Default resolution managing system that was available with Cocos2d-x framework could not display all the elements of the gameplay on all types of screens and have to cut some visible parts of canvas or stretch and shrink the objects. This behaviour remains in Cocos2d-x because originally Cocos2d was designed for iPhone and iPad devices that have very limited set of possible resolutions. To scale

images and locate them properly on the screen, resolutions manager class was developed during this project. It can detect current device screen properties and decide how to scale available assets and change their coordinates on the screen. The main requirement for this logic was in keeping the graphics aspect ratios same as original to have objects proportions unmodified and to locate elements inside the visible area of the canvas to avoid moving elements outside of visible borders or cut them.

The approach that was used is in creating of the virtual game screen that matches initial graphics dimensions of 960x640 pixels and locate all game elements on it. The goal of resolutions manager is to find such scale values and screen positions that virtual game screen canvas that all elements will be always visible and fit to the screen of any device. On devices with screen ratios like iPad 1024x768 game will have extra visible parts on top and the bottom of the screen like show on figure 24. And on devices with wide screens like iPhone 5 1136x640 on figure 25 game will have extra visible areas by sides, but all game elements will always fit on the screen.

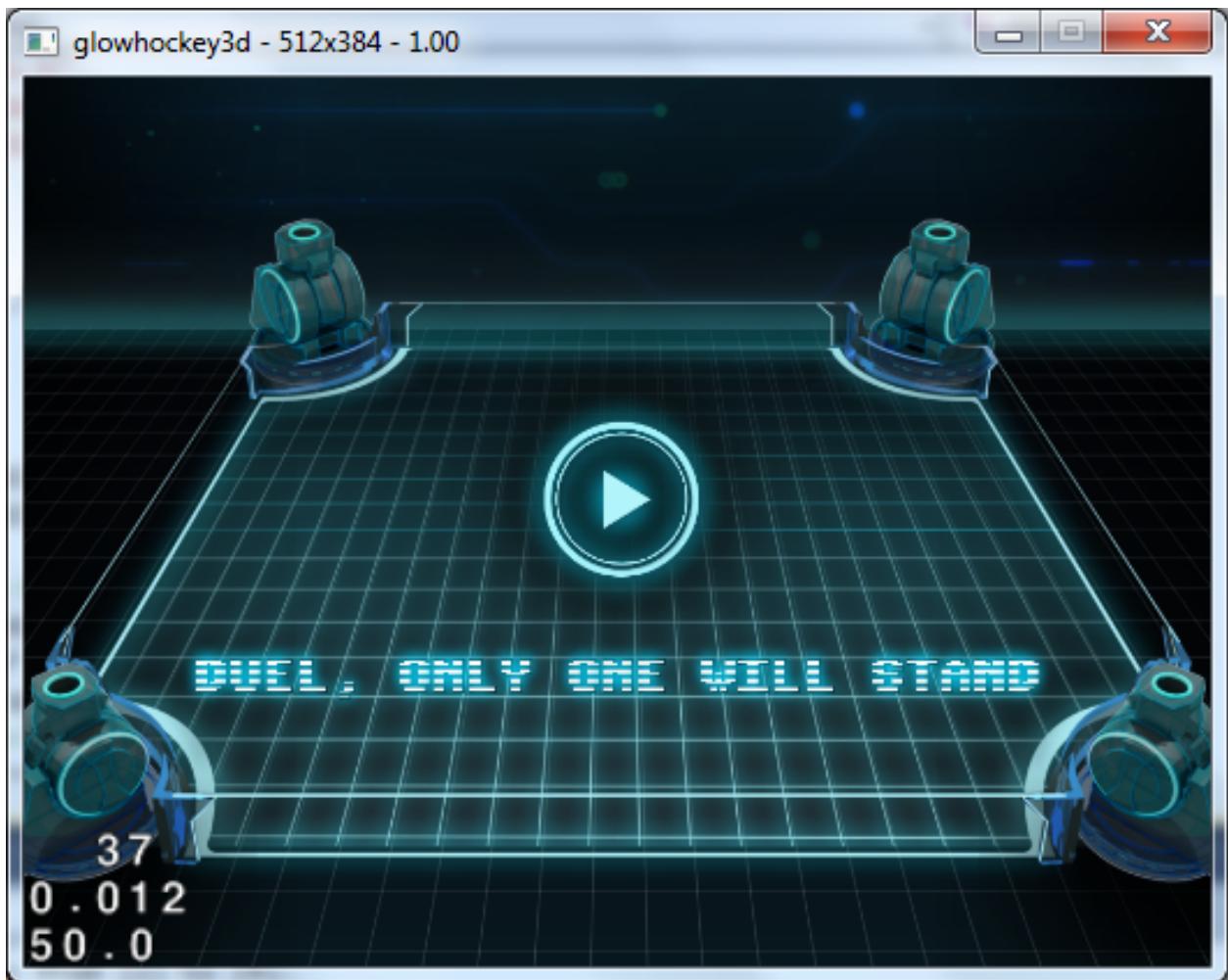


FIGURE 24. Glow Hockey 3D running in iPad screen size resolution mode

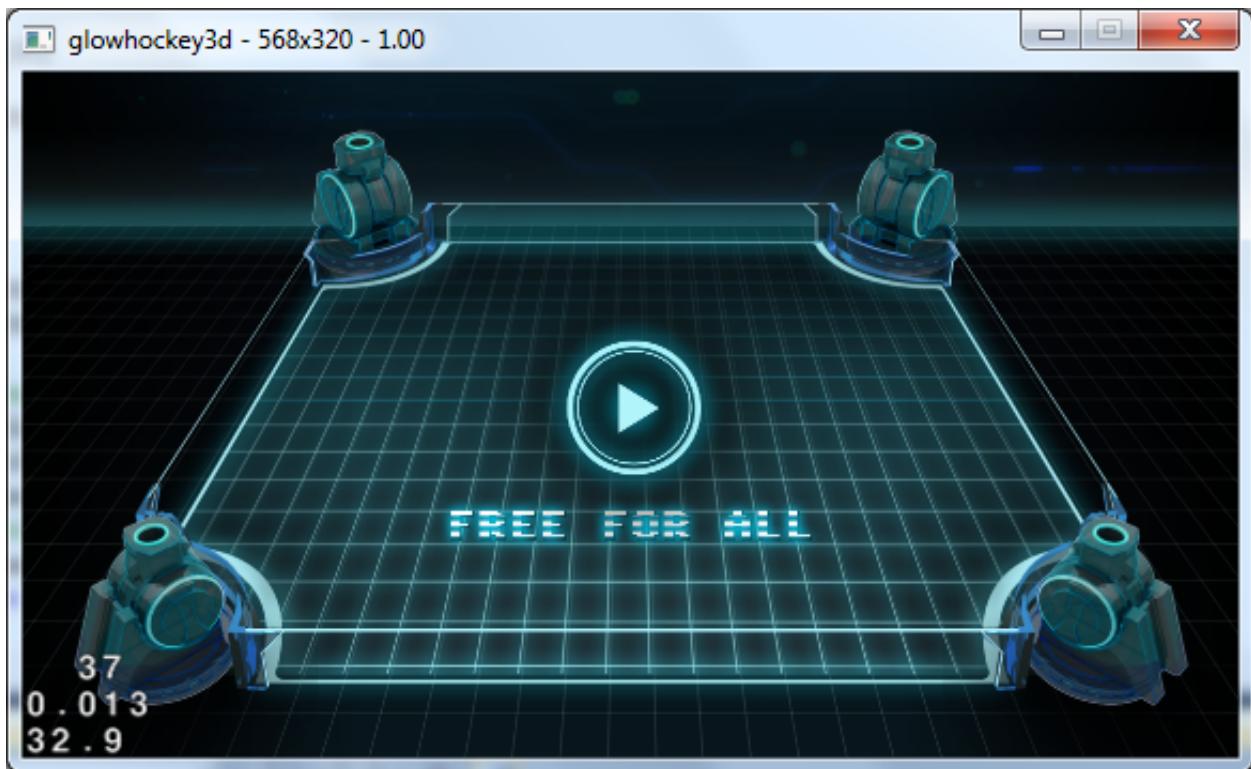


FIGURE 25. Glow Hockey 3D running in iPhone 5S screen size resolution mode

Setting up this configuration once provides for the game developer opportunity to concentrate on the game logic programming and animations on abstract game canvas and not carrying about the device specific screen size and aspect ratios. Using the emulator with default 960x640 pixels resolutions would be enough for the time of game development process, leaving the actual graphics positioning on target devices to resolution manager.

4.4 Handling formats: sounds, graphics, fonts

Different platforms supports sound effects and background music in different formats, according to Cocos2d-x sounds API guide.

Sets of audio assets should be prepared for every target platform: for Android sound effect file is stored in .ogg format, for Windows in .wav, for iOS in .caf and for Marmalade in raw PCM format. To handle audio playback of different sound formats project implements Sound manager class that loads proper files according to current running platform and adjusts sound properties like shown on figure 26.

The screenshot shows the 'SoundManager.h' file in the 'SoundManager' folder of a Cocos2d-x project. The code defines platform-specific sound file paths using preprocessor directives:

```

// android sound effects support ogg format
#ifndef CC_TARGET_PLATFORM
#define CC_TARGET_PLATFORM CC_PLATFORM_ANDROID
#endif

#define SFX BALL BALL "sfx/sfx_ball_ball.ogg"

// marmalade sound effect support PCM format
#ifndef CC_TARGET_PLATFORM
#define CC_TARGET_PLATFORM CC_PLATFORM_MARMALADE
#endif

#define SFX BALL BALL "sfx_raw/sfx_ball_ball.raw"

// win32 sound effects support wav format
#ifndef CC_TARGET_PLATFORM
#define CC_TARGET_PLATFORM CC_PLATFORM_WIN32
#endif

#define SFX BALL BALL "sfx_wav/sfx_ball_ball.wav"

#endif // SoundManager_h

```

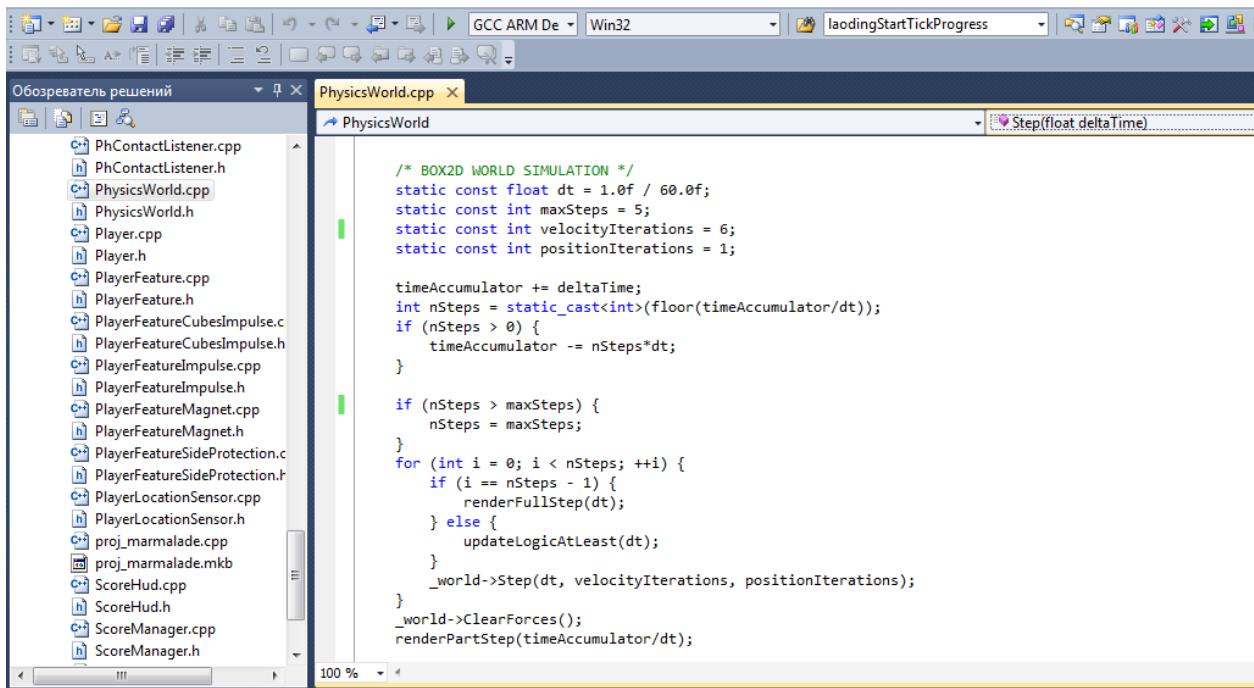
FIGURE 26. Sound formats specific to different platforms definitions

4.5 Creating game loop

The gameplay of developed game is a variation of Pong or Air Hockey games genre - high-speed arcade action with many moving elements and collisions. The best way to implement game mechanics was to delegate physics simulation of moving bodies and players to one of the supported by Cocos2d-x physics simulation engine. The most common engine for 2D game development is Box2D - it is free and integrated in Cocos2d-x as a library, well documented and with many examples of use. Box2D creates a physics layer of the game; it contains level boundaries, players on arena edges, balls running on arena and will simulate their interactions based on the objects properties. Input from players or AI changes the position of paddles on arena, all the rest is handled by Box2D simulation: the positions of elements, the collisions and bouncing, crossing the goal line, collision with bonuses or obstacles, boundaries of the level, speeds and other properties.

To simulate the physics layer game should create the Box2d world, add all required objects with custom properties for behaviour and update the simulation physics step constantly in a game loop.

Game loop function in Cocos2d-x updates with given time interval, usually it is 60 times per second, but because of different performance on end devices, different events on background and lag delays update() function is called after different amount of time every time, the float variable `delayTime` passed to update() function indicates number of seconds passed from previous call, using this value game can adjust number of physics steps performed by one game loop to keep physics simulations and movements on the screen constant and independent from the device performance, this technique is called “fixed time step” and shown on figure 27.



```

/* BOX2D WORLD SIMULATION */
static const float dt = 1.0f / 60.0f;
static const int maxSteps = 5;
static const int velocityIterations = 6;
static const int positionIterations = 1;

timeAccumulator += deltaTime;
int nSteps = static_cast<int>(floor(timeAccumulator/dt));
if (nSteps > 0) {
    timeAccumulator -= nSteps*dt;
}

if (nSteps > maxSteps) {
    nSteps = maxSteps;
}
for (int i = 0; i < nSteps; ++i) {
    if (i == nSteps - 1) {
        renderFullStep(dt);
    } else {
        updateLogicAtLeast(dt);
    }
    _world->Step(dt, velocityIterations, positionIterations);
}
_world->ClearForces();
renderPartStep(timeAccumulator/dt);

```

FIGURE 27. Fixed time step in Box2D

When physics layer is simulated and all objects act according to the rules that game set to them and reacts on player's input, it is time to create visual representation of processes that take place in Box2D world. Game uses art that creates an illusion of 3D game, this is also known as 2.5D perspective or pseudo-3D, this technique was widely used to add depth in simple 2D games but with increasing performance of modern smartphone devices it has become much easier to create game in real 3D environment. All images and sprites in game were modelled in 3D editor and rendered into 2D sprite sheets that represents square game field arena from first player view perspective. To achieve the effect of 3D depth shown on figure 28, game introduces the mechanism for repositioning

elements and changing depth property based on 2D coordinates from physics layer, objects that are closer to player's edge will appear bigger and in front of objects that are further from player's point of view. This coordinates transformation algorithm updates position of all game elements at every game loop step depends on player's perspective and chosen edge position.

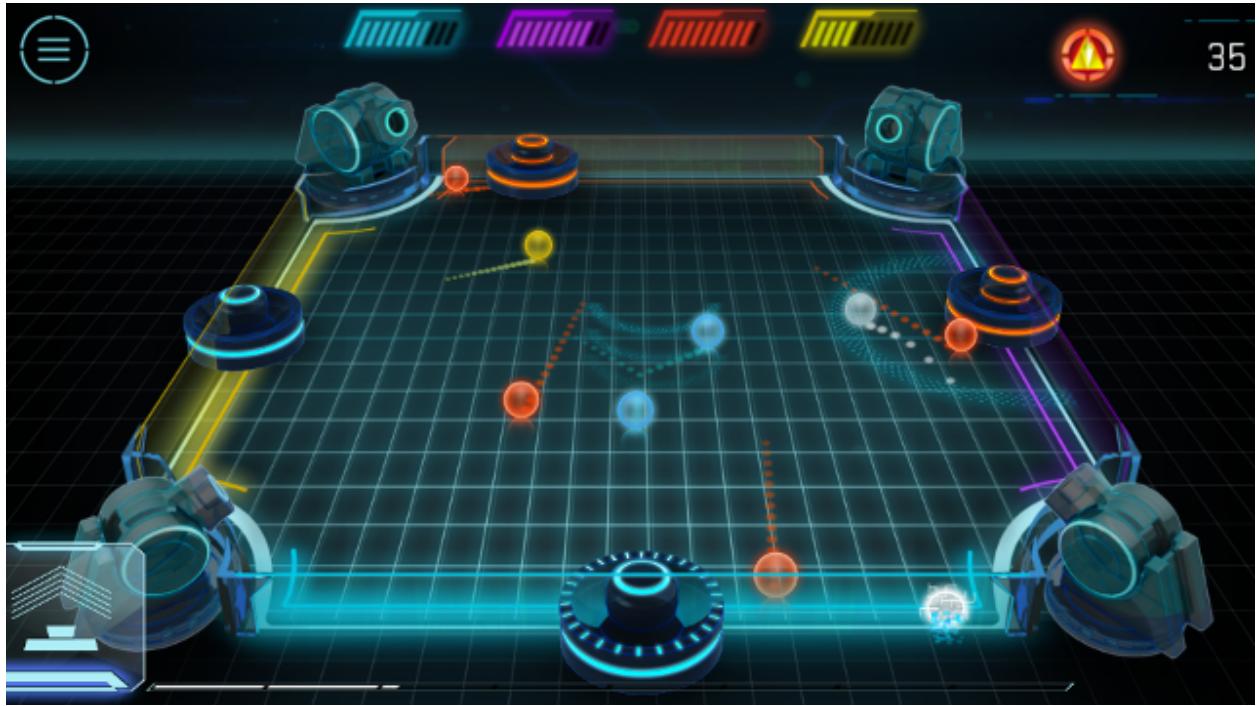


FIGURE 28. Pseudo 3D look in Glow Hockey 3D

4.6 Performance optimization

Generally game performance depends on target device specifications, but also performance level depends on game code and resources optimization. To achieve stable 60 FPS on most of devices for all gameplay situations game should use limited CPU and GPU resources wisely. Most significant drawbacks in performance for 2D games are brought by number of draw calls, physics simulation and memory allocations. We will take a short overview on improvements that Glow Hockey 3D game introduced to increase performance and to achieve desired FPS rate.

In Cocos2d-x terminology draw call refers to a number that indicates how many objects are drawn on screen canvas per one game tick, higher number means more time spent on drawing the whole game state to the device screen. Current number of draw calls can be checked in debug output of running Cocos2d-x application next to current FPS value. Glow Hockey 3D has many complex elements on the screen that might consist of great number of sub-images, for example ball object has 21 sub-sprites to indicate its body, shadow, reflection, tail and spawning animation. Simultaneously on arena can be 4 players, dozens of balls, bonuses and obstacles, not limited number of collision visual effects, elements of arena itself and user interface at one point of time. Not optimized version of Glow Hockey 3D had more than thousand of draw calls depend on number of balls on the arena and number of players, because in multiplayer mode number of draw calls get doubled as shown on figure 29 of split screen mode look.

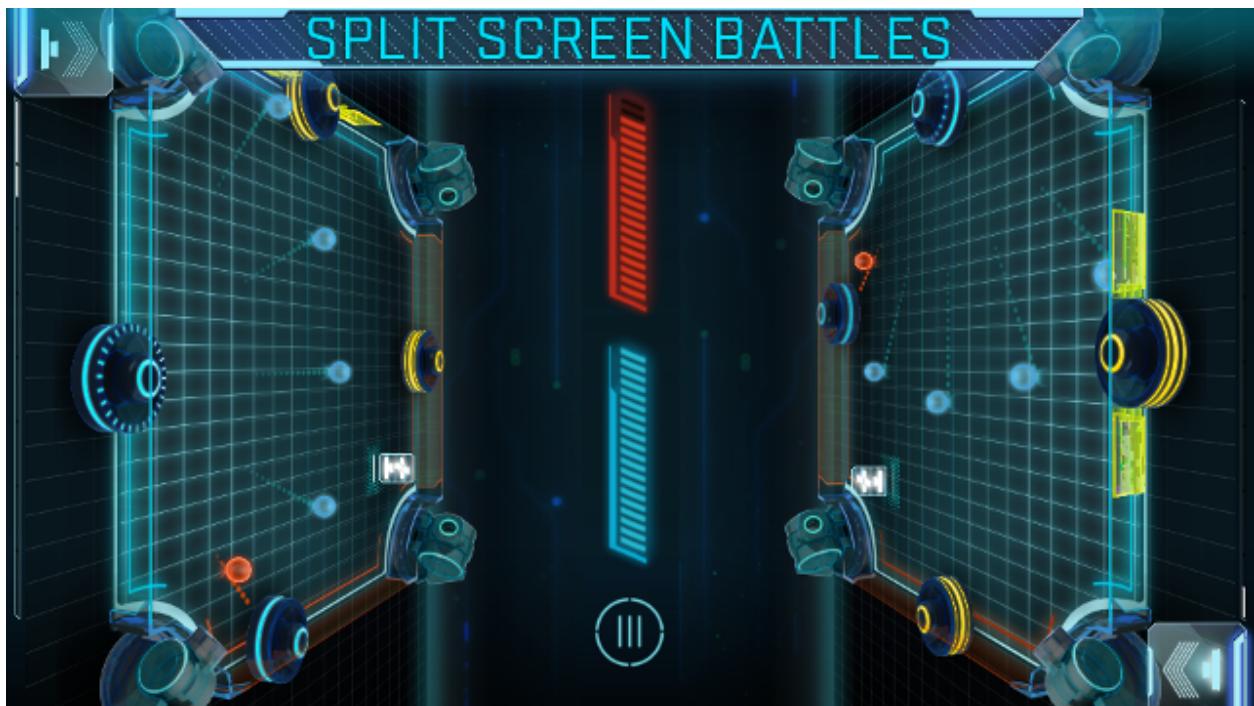


FIGURE 29. Glow Hockey 3D in split screen mode

This can be easily optimized by using Batch Nodes, this technique consists of grouping sprites into sprite sheets and attaching grouped elements to one node. As the result all elements attached to same node will be drawn at one draw call, no matter how many elements are present. Though sprite sheets have some limitations depend on target devices and in general are bounded by 2048x2048

pixels of PNG image as shown on figure 30. As the result of grouping sprites to the Batch helped to reduce number of draw calls to one dozen and improved game performance noticeable.



FIGURE 30. Glow Hockey 3D images grouped into sprite sheet

Another significant performance drop that cause lags in the gameplay is a memory allocation or deallocation procedures during active gameplay. All expensive memory allocation operations such as game objects creation and deletion should be performed during level loading time. In the early versions of Glow Hockey 3D every spawning ball or obstacle was created and destroyed at gameplay

time, every new ball spawn operation created the new instance of ball object that includes creation of Box2D body for physics layer world, initialization of sprites and animations for visual effects. That flow brought time lags depend on device computing power. Fortunately this can be easily avoided by creating the set of reusable objects at level loading time and bring them into gameplay of necessity putting them aside after they were used. Adding reusable objects pools helped to improve performance to keep FPS level stable during the gameplay.

4.7 Platform specific APIs

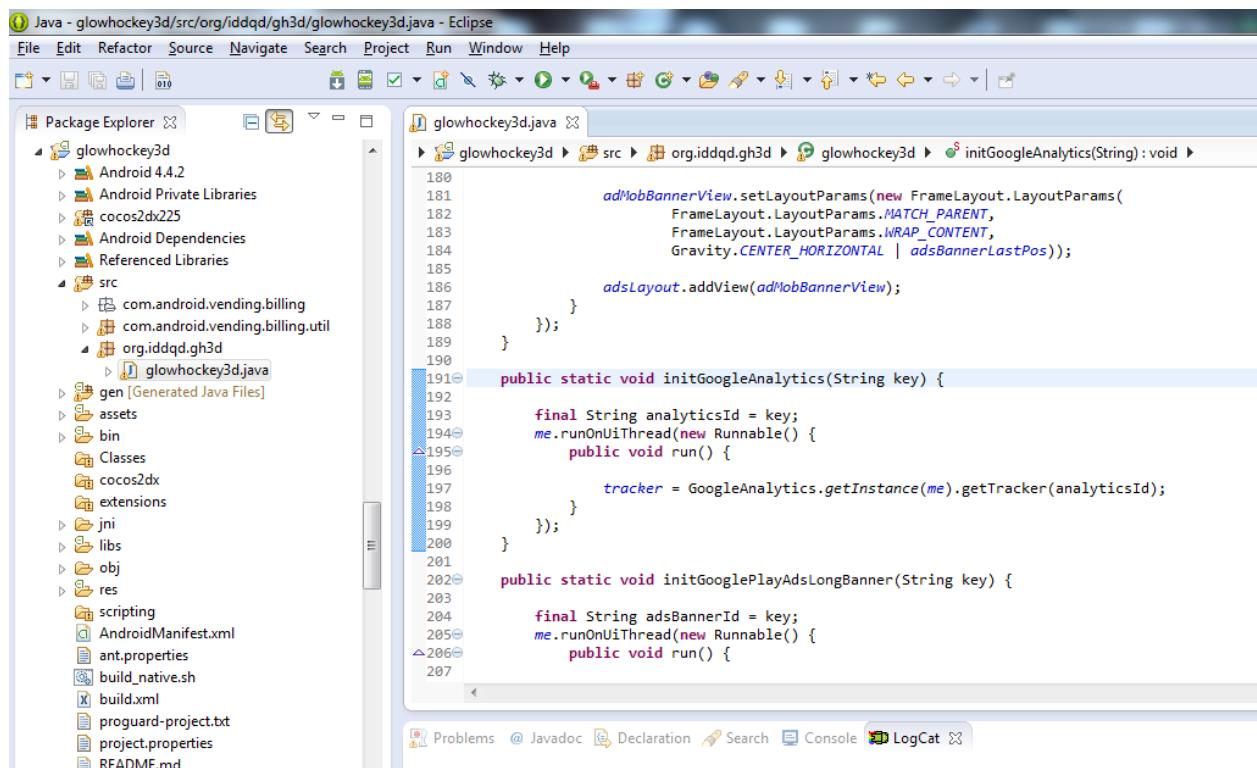
The developed gameplay is common for all platforms, but many features are platform specific and should be implemented separately for every platform or even specific to the store of app distribution. For example in-app purchases mechanism, advertising, analytics systems, opening external links and rating the game in app stores.

Google Play and Android was chosen as first target platform to deploy Glow Hockey 3D game and adding platform specific services. Cocos2d-x provides JNI bridge to call Java methods from main thread of C++ application, this allows application to invoke methods written in Java to perform native Android functions like opening URL, launching Google Play application and everything defined in Java part of the project as shown on figure 31. Glow Hockey 3D project contains number of custom managers that handles platform specific functionality to invoke relevant methods depend on current build configuration.

```
void initGoogleAnalytics(const char* packageName, const char* key)
{
    JniMethodInfo t;
    if (JniHelper::getStaticMethodInfo(t, packageName
        , "initGoogleAnalytics"
        , "(Ljava/lang/String;)V"))
    {
        jstring StringArg0 = t.env->NewStringUTF(key);
        t.env->CallStaticVoidMethod(t.classID, t.methodID, StringArg0);
    }
}
```

FIGURE 31. Cocos2d-x calling Android Java methods with JNI

In Android Java project part game defines all corresponding methods and functionality. The required functionality that should be implemented on Android side of the project are: Google Analytics instance to track player's activity and performance, Google AdMob advertisement service to show and hide banners during different states of the gameplay, Google Play in-app purchases to sell extra content, opening the external links and navigate user to ranking page, social media accounts and contact options. Figure 32 shows the implementation of JNI function on Android part of the project.



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "glowhockey3d". It includes the main source file "glowhockey3d.java" and various build and resource directories like "gen", "assets", "bin", "jni", "libs", "obj", "res", "scripting", "AndroidManifest.xml", "ant.properties", "build_native.sh", "build.xml", "proguard-project.txt", "project.properties", and "README.md".
- Code Editor:** Displays the Java code for "glowhockey3d.java". The code implements methods for initializing Google Analytics and Google Play Ads. The method signatures are:


```

public static void initGoogleAnalytics(String key) {
    final String analyticsId = key;
    me.runOnUiThread(new Runnable() {
        public void run() {
            tracker = GoogleAnalytics.getInstance(me).getTracker(analyticsId);
        }
    });
}

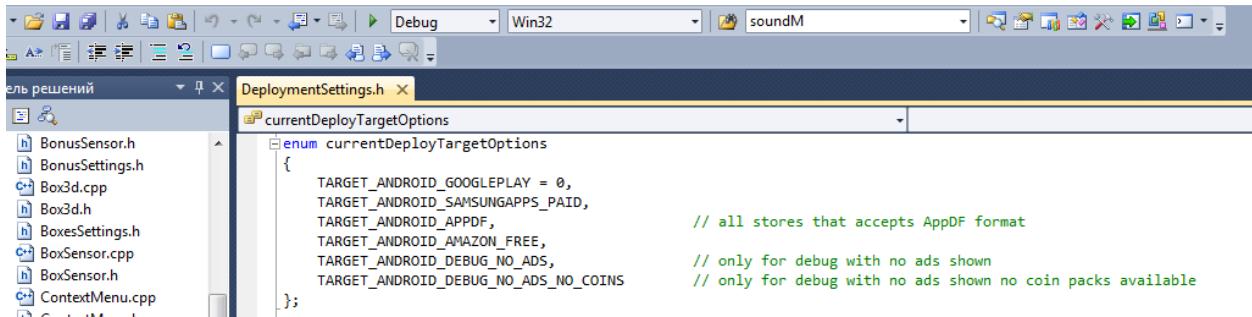
public static void initGooglePlayAdsLongBanner(String key) {
    final String adsBannerId = key;
    me.runOnUiThread(new Runnable() {
        public void run() {
    
```
- Toolbars and Status Bar:** Standard Eclipse toolbars and status bar at the bottom.

FIGURE 32. Implementation of called from C++ with JNI Android Java methods

4.8 Handling deployment targets

To keep project clean and to avoid creating platform specific branches Glow Hockey 3D introduced deployment target class that includes specific configurations depend on build target. Every defined build target has its own profile that contains set of specific build parameters; it might include platform specific options or particular store settings. The main properties that deployment profile should define for current build are: profile name, package identifier, type of analytics system, type of advertising

system, type of in-app purchase system, starting balance, availability of contact options. Using this build profile system allows adding extra functionality for specific platforms individually, updating the core game functionality at once for all platforms and building game to all target platforms faster just changing the deploy profile key as shown on figure 33.



```

DeploymentSettings.h
enum currentDeployTargetOptions
{
    TARGET_ANDROID_GOOGLEPLAY = 0,
    TARGET_ANDROID_SAMSUNGAPPS_PAID,
    TARGET_ANDROID_APPDF,
    TARGET_ANDROID_AMAZON_FREE,
    TARGET_ANDROID_DEBUG_NO_ADS,
    TARGET_ANDROID_DEBUG_NO_ADS_NO_COINS
};
```

The code shows an enum named `currentDeployTargetOptions` with several members. The first member, `TARGET_ANDROID_GOOGLEPLAY`, is assigned the value `0`. The other members are listed without explicit values. To the right of each member, there are comments: `// all stores that accepts AppDF format` for `TARGET_ANDROID_APPDF`, `// only for debug with no ads shown` for `TARGET_ANDROID_DEBUG_NO_ADS`, and `// only for debug with no ads shown no coin packs available` for `TARGET_ANDROID_DEBUG_NO_ADS_NO_COINS`.

FIGURE 33. Deployment target management

4.9 Deployment

Glow Hockey 3D project was deployed to Android, Blackberry 10 and Windows PC devices. The absence of iOS and Windows Phone platforms in this list is caused by software and hardware absence during the project development: to deploy application for iOS developer should purchase Apple license and to build game for Windows Phone platforms hardware requires 64bit Windows 8 OS.

Deploying game for Android OS requires installed Android SDK and NDK and the Eclipse IDE, “proj.android” should be imported in Eclipse as “Existing Android Code into Workspace” as shown on figure 34.

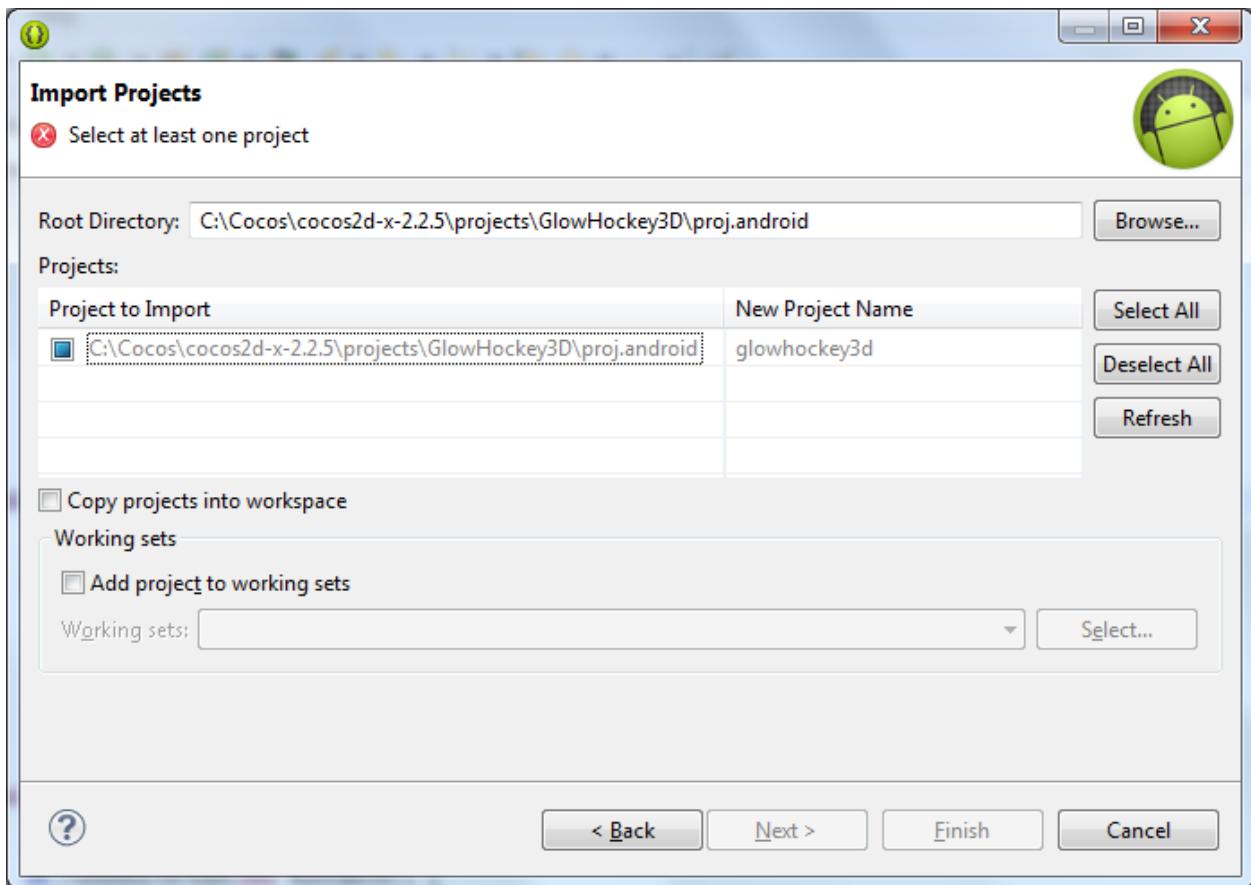
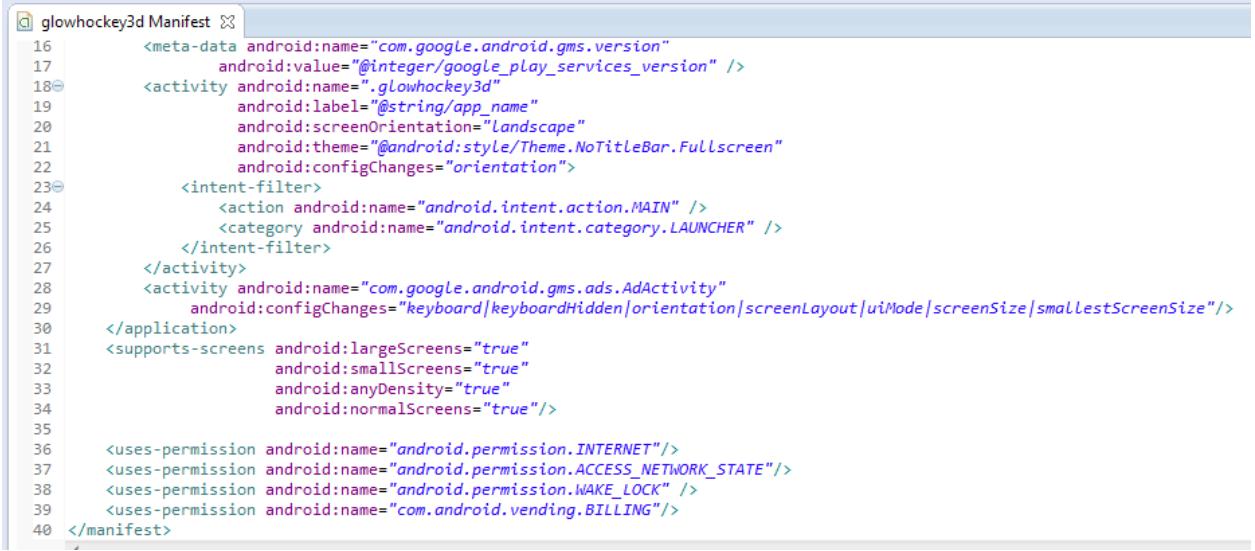


FIGURE 34. Import of Android Cocos2d-x project into Eclipse workspace

Android project will require special functionality that will be provided by 3rd party libraries: Google Play and Billing services, Google Analytics, Social Services and Leader boards. To compile and run project adding of special permissions to Android manifest file should be done as shown on figure 35.



```
glowhockey3d Manifest
16     <meta-data android:name="com.google.android.gms.version"
17             android:value="@integer/google_play_services_version" />
18
19     <activity android:name=".glowhockey3d"
20             android:label="@string/app_name"
21             android:screenOrientation="landscape"
22             android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
23             android:configChanges="orientation">
24         <intent-filter>
25             <action android:name="android.intent.action.MAIN" />
26             <category android:name="android.intent.category.LAUNCHER" />
27         </intent-filter>
28     </activity>
29     <activity android:name="com.google.android.gms.ads.AdActivity"
30             android:configChanges="keyboard/keyboardHidden/orientation/screenLayout/uiMode/screenSize/smallestScreenSize"/>
31 </application>
32 <supports-screens android:largeScreens="true"
33                     android:smallScreens="true"
34                     android:anyDensity="true"
35                     android:normalScreens="true"/>
36
37 <uses-permission android:name="android.permission.INTERNET"/>
38 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
39 <uses-permission android:name="android.permission.WAKE_LOCK" />
40 <uses-permission android:name="com.android.vending.BILLING"/>
41
42 </manifest>
```

FIGURE 35. Glow Hockey 3D Android manifest file in the Eclipse IDE

After all native Android functionality is added and project contains no error it can be compiled and run on Android device or exported as a signed Android application protected with a keystore and a private key.

Deploying game for Blackberry 10 OS requires Marmalade SDK deployment tool and installed Blackberry Web Works SDK that provides set of binary tools for application signing process. Compiling game code with ARM release profile and selecting a target platform from the list of available options will create an executable Blackberry 10 .bar file. Figure 36 shows available options of deployment to Marmalade SDK builds.

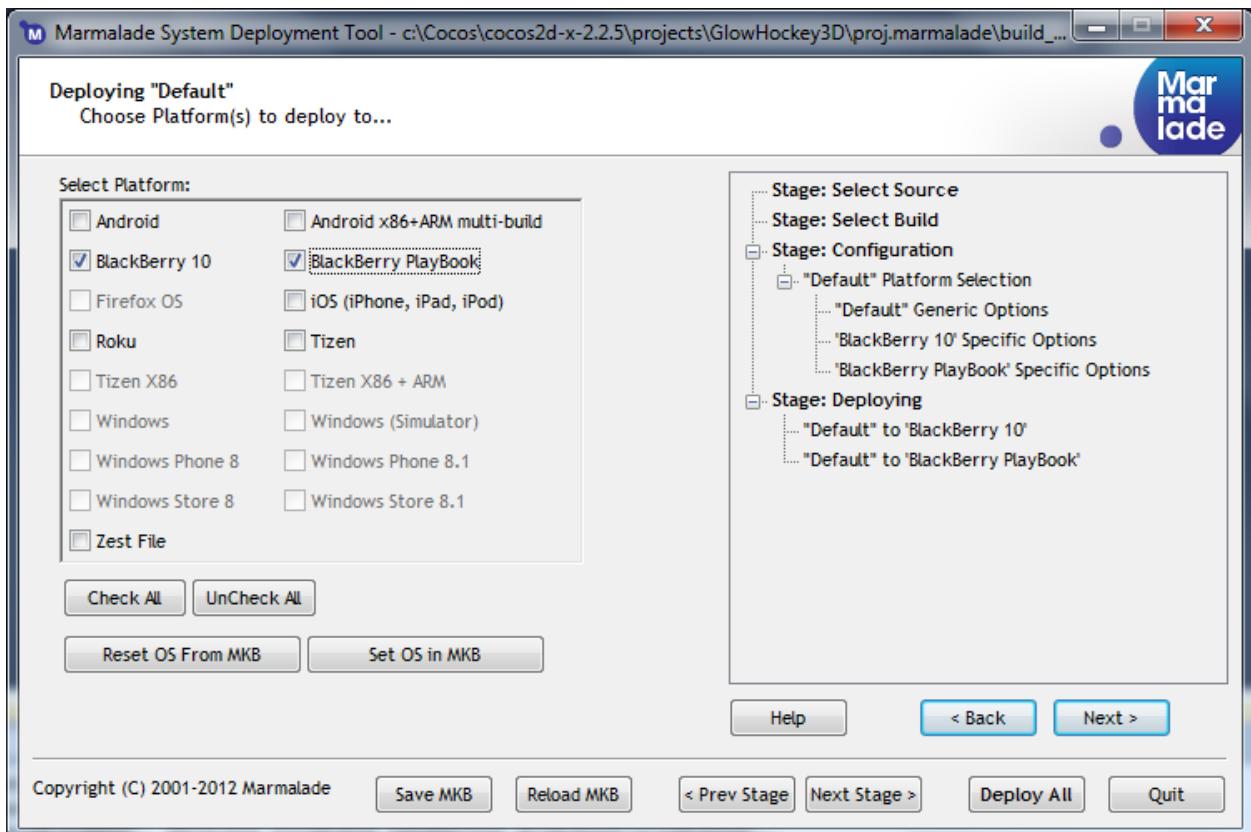


FIGURE 36. Marmalade SDK deployment tool

4.10 Submitting

Submitting game to Android stores like Google Play and Amazon Appstore requires signed version of Android executable .apk file, set of graphical assets for specific store rules and application description sections. Figure 37 shows Glow Hockey 3D game on the Google Play store.



FIGURE 37. Glow Hockey 3D Google Play Android application in developers console

Glow Hockey 3D was submitted to number of Android markets to evaluate the quality of game based on player's reviews and statistics that is gathered by stores functionality such as crash reports and based on analytics data available in real time like shown on figure 38.

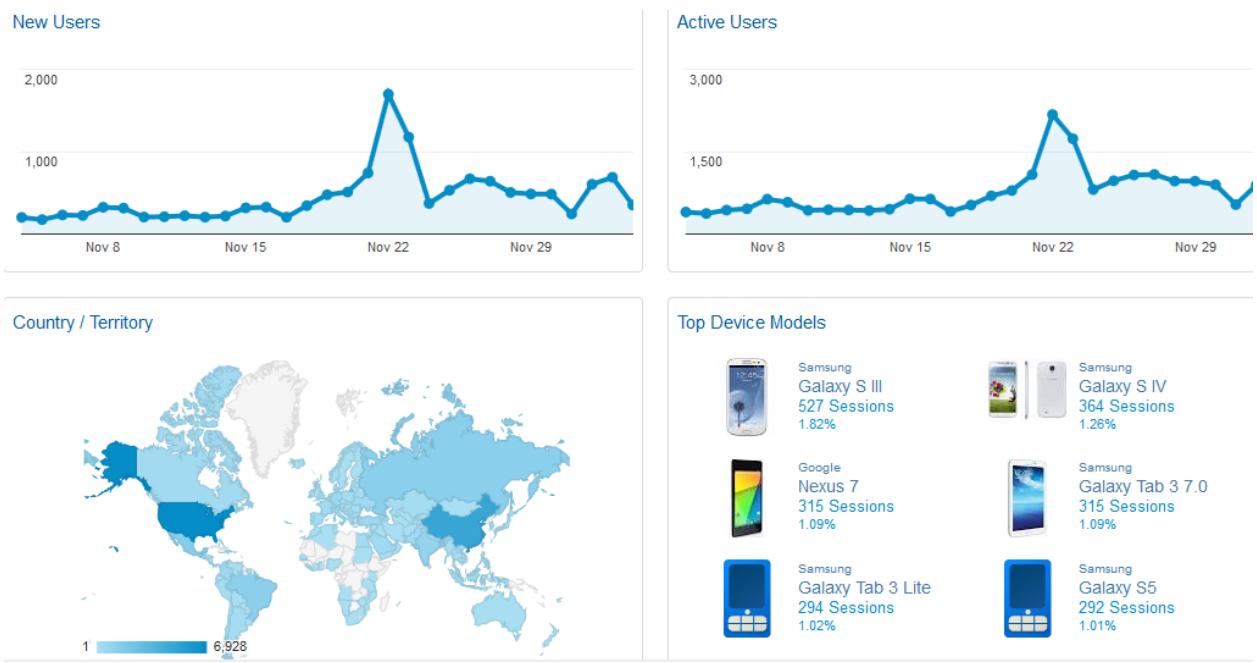


FIGURE 38. Google Analytics for Glow Hockey 3D application new and active users for 30 days period

5 CONCLUSION

Cross-platform mobile game development has many advantages over the native development process. Use of the multi-platform development frameworks helps to bring application to many target platforms at once, make the process of maintenance simpler but also allows using any platform specific functionality if needed.

Developed application has shown great performance on many different platforms and different devices that covers significant share of modern mobile devices market. Proper choice of CPT from the list of available options what was based on theoretical studies helped to save resources and to choose most suitable game engine for selected game genre.

6 DISCUSSIONS

During the theoretical part of research I have learned about many topics related to mobile software development, current state of the market of the mobile operating systems and trends for mobile market in general. The overview of available software solutions for cross-platform mobile development helped me to choose set of software and hardware components for fast and reliable mobile game development. The game developed during this project shows positive progress in number of downloads and user ratings on all mobile platforms that received final builds.

The knowledge that I gained developing Glow Hockey 3D game helped me to find a position of mobile game developer and continue cross-platform game development with Cocos2d-x and Marmalade SDK.

REFERENCES

- Anjum, M. 2014. Marmalade SDK 7.3 for Windows platform released, new features and is now free. Cited 6.12.2014. <http://www.winbeta.org/news/marmalade-sdk-73-windows-platform-released-new-features-and-now-free>
- Asay, M. 2014. It Really Pays To Be An iOS Developer, But There's Still Hope For Android. Business Insider Inc. Cited 4.12.2014. <http://www.businessinsider.com/android-vs-ios-developers-2014-8>
- Cocos2d-x.org 2014. About Cocos2d-x. Cited 2.12.2014. <http://www.cocos2d-x.org/about>
- Cocos2d-x.org 2014. The first joint release of the Cocos2d family. Cited 11.12.2014. <http://www.cocos2d-x.org/news/81>
- Cocotias, A. 2013. Samsung and Apple Still Dominate Smartphone Shipments. Cited 3.12.2014 <http://www.businessinsider.com.au/samsung-and-apple-split-phone-shipments-2013-5>
- Corona Labs 2014. Corona SDK. Cited 6.12.2014. <http://coronalabs.com/products/corona-sdk/>
- Cowart, J. 2014. Pros and Cons of the Top 5 Cross-Platform Tools. Cited 6.12.2014. <http://www.developereconomics.com/pros-cons-top-5-cross-platform-tools/>
- Dormehl, L. 2014. Apple Is Beating Google When It Comes To iOS Game Exclusives. Cited 4.12.2014) <http://www.cultofmac.com/275289/apple-beating-google-comes-ios-game-exclusives/>
- Future Publishing Limited 2014. Two billion downloads? We're just getting started, says Angry Birds creator Rovio. Cited 3.12.2014 <http://www.edge-online.com/features/two-billion-downloads-were-just-getting-started-says-angry-birds-creator-rovio/>

Gartner inc 2014. Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. Cited 2.12.2014 <http://www.gartner.com/newsroom/id/2665715>

H., M. Phonearena.com 2013. Google Play app revenue could surpass iTunes App Store by the end of 2013, will developers follow? Cited 3.12.2014 http://www.phonearena.com/news/Google-Play-app-revenue-could-surpass-iTunes-App-Store-by-the-end-of-2013-will-developers-follow_id39292

KPCB 2014. Internet trends 2014 – code conference. Cited 3.12.2014. <http://www.kpcb.com/internet-trends>

LeBrun, T. 2014. Build MVVM Apps with Xamarin and MvvmCross. Cited 7.12.2014.
<http://msdn.microsoft.com/en-us/magazine/dn759442.aspx>

Marmalade Technologies Ltd 2014. Technical details. Cited 4.12.2014.
<https://www.madewithmarmalade.com/products/technical-details/cpp/cross-platform-apis>

OpenSignal Inc 2014. Android Fragmentation Visualized. Cited 12.12.2014.
<http://opensignal.com/reports/2014/android-fragmentation/>

Pogorzelska, Z. 2014. Cross-Platform Tools Benchmarking 2014 results: Cross-Platform Tools are making a big progress in the app developer community. Cited 4.12.2014.
<http://research2guidance.com/cpt-benchmarking-2014-results-cross-platform-tools-are-making-a-big-progress-in-the-app-developer-community/>

Reisinger, D. 2014 Nokia officially walks away from Symbian, MeeGo. Cited 3.12.2014
<http://www.cnet.com/news/nokia-officially-walks-away-from-symbian-meego/>

Statista 2014. Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 4th quarter 2013. Cited 3.13.2014.
<http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>

- Singh, S. 2012. The fallacy of Android fragmentation – a statistical analysis. Cited 6.12.2014.
<http://www.androidauthority.com/the-fallacy-of-android-fragmentation-a-statistical-analysis-73646/>
- Unity Technologies 2014. EFFORTLESSLY UNLEASH YOUR GAME ON THE WORLD'S HOTTEST PLATFORMS. Cited 6.12.2014. <http://unity3d.com/unity/multiplatform>
- VisionMobile Limited 2014. Developer Economics Q3 2014: State of the Developer Nation. Cited 4.12.2014. <http://www.developereconomics.com/reports/developer-economics-q3-2014/>
- Voskoglou, C. 2014. Getting your priorities straight. Cited 4.12.2014.
<http://www.developereconomics.com/report/q1-2014-getting-your-priorities-straight/>
- Xamarin Inc 2014. Platform. Cited 8.12.2014. <http://xamarin.com/platform>