An evaluation of the framework Libgdx when developing a game prototype for Android devices

Date	2011-06-14	Examiner	Mads Dam
Location	Stockholm, Sweden	Supervisor	Henrik Eriksson
University	KTH - Royal Institute of Technology		
School	CSC - School of Computer Science and Communication	Author	Rakiv Ahmed rakiv@kth.se Beckombergavägen 13 168 54 Bromma 073 - 996 21 67
Course	DD143X Degree Project in Computer Science, First Level (15.0 credits)	Author	Jonas Aule aule@kth.se Blendavägen 31 187 77 Täby 076 - 160 32 26

Abstract
This paper is about evaluation of the framework *Libgdx,* regarding 2D game development for the Android platform. Other frameworks are also investigated briefly. The game under development is an original game called *Magnificent Marbles* and is multi-modal as other games for smartphones. The Background section talks about the Android platform and its related tools along with game examples, also frameworks are introduced here. The main focus of this paper will be on the Implementation section and the actual usage of *Libgdx*. Methods and results for different development versions of the game will be displayed.

The Discussion section concludes that *Libgdx* is a good framework due to its desktop support. *Libgdx* may not suitable for smaller game projects or when wanting to get started quickly without too much knowledge in other areas, such as OpenGL. Another issue is that one should not solely rely on *Libgdx* desktop support, but knowledge in other areas is required as well to utilize Libgdx to the fullest. The Appendix section includes different documents for more information about *Magnificent Marbles.*

Sammanfattning
Denna rapport handlar om utvärderingen av ramverket *Libgdx* och spelutveckling för Androidplattformen. Även andra ramverk undersöks också kortfattat. Spelet under utveckling är ett nytt spel som heter *Magnificent Marbles* och är multimodalt liksom andra spel för *smartphones*. Background-avsnittet berättar om Androidplattformen och dess relaterade verktyg tillsammans med spelexempel. Olika ramverk kommer att presenteras. Tyngdpunkten kommer att ligga i Implementation-avsittet som handlar om den faktiska användningen av *Libgdx*. Metoder och resultat för olika versioner av spelet visas upp.

Discussion-avsnittet drar slutsatsen att Libgdx är ett användbart verktyg, mycket tack vare dess *desktop support*. *Libgdx* kanske inte lämpar sig för mindre spelprojekt eller när man snabbt vill komma igång utan förkunskaper i andra ämnen, såsom OpenGL. Dessutom ska man inte enbart förlita sig på Libgdxs *desktop support,* utan kunskaper i andra områden krävs fortfarande för att utnyttja Libgdx till fullo. Appendix-avsnittet innehåller dokument med ytterligare information relaterad till *Magnificent Marbles.*

Table of Contents

List of Figures

List of Tables

Definitions

Term	Abbreviation	Description
Android Development Tools	ADT	A plug-in for Eclipse (see Eclipse) to develop Android applications.
Android		Operating system for smartphones.
Android Market		The Android distribution service of mobile applications.
Android Lifecycle		A model Android uses to handle the lifecycle of an activity in applications. There are some required methods, such as onCreate, onResume, onPause, that needs to be implemented for every Android activity following the lifecycle model.
Box2D		A physics engine library that are used in 2D games and have implementations in different programming languages and frameworks.
Eclipse		An Integrated Development Environment (see IDE) that supports Android Development via ADT.
Frames Per Second	FPS	A way to measure how fast a device and/or application is at displaying and updating the graphics on the screen. How many frames it draws each second.
Framework		A library containing commonly used functionality to use when wanting to ease the development of a specific type of application. The difference between a library and a framework is the so called “inversion of control”, the framework runs the main program loop letting the program using the framework override certain default framework behavior by providing methods that the framework calls at certain times.
Multi-modal		Using other ways of interacting with a computer device other than the regular mouse and keyboard combination. In our case the use of touch and accelerometers on a smartphone.
Open Source		A software project where all the source code is freely available to developers, to adjust and use. May come with different licenses.
Open Graphics Library Embedded Systems	OpenGL ES	An open source graphics library in native code (C) for creating graphics. ES is adapted for mobile devices.
Prototype		The development up to an early alpha version of a game or application. Used for testing purposes and to extend or learn from for future versions.
Software Development Kit	SDK	A package of functionality to makes it easy to build software for a specific platform, such as Android.
Swedish Game Awards	SGA	A yearly game development competition for students.

Term	Abbreviation	Description
Smartphone		The latest generation of mobile phones that can run advanced software and have a multi-modal interface, such as HTC Desire or iPhone. For example have a touch interface that enables users to control the mobile by touching the screen.
Tiled Map Editor		Tiled Map Editor is a third party tool to enable easy map (level) creating for games, using XML. Has built-in support in Libgdx.
Themable Widget Library	TWL	TWL is a graphical user interface library for Java built on top of OpenGL. It provides a rich set of standard widgets such as labels, buttons, fields etc.

1. Introduction

The market for mobile applications, and especially games, has exploded since the introduction of Apple's iPhone[1] and the later entrance of Android-based smartphones. Earlier mobile games were simple and a pastime activity. However now, advanced and high quality games exist, and there are more of them as games for handheld consoles are being further developed.

The business and technology involved with smartphones have also evolved, such as the introduction of accurate touch interfaces and accelerometers. The smartphone application industry is relatively new and an unexplored market with these new features and opportunities. It is very attractive for a software engineer who wants to work with something new, exciting and lucrative, in the views of the authors.

This report will be about game development for the Android platform. The game, called *Magnificent Marbles,* is an original and takes advantage of the multi-modal features of a smartphone. The development will occur with the help of a framework, called Libgdx.

1.1. Intended readership

This report aims at supporting other computer science and software engineering students that have a couple of years of experience with programming (object oriented, such as Java) and different fundamental theories. An understanding of basic terminology for software development is required. Specific terminology and techniques for mobile, Android, Libgdx and game development will be provided. The reader is interested in mobile or game development, preferably both. Particular interest lies in the use of a framework to simplify and speed up development and especially looking for an evaluation of the Libgdx framework.

[1] Smartphones Provide Extra Mana for Mobile Games Industry […]
<http://www.comscore.com/Press_Events/Press_Releases/2009/1/Mobile_Gaming_Grows>
[Retrieved 2011-04-08] [Published 2009-01-30]

1.2. Purpose

1.2.1. Problem statement

The main problem statement is the following:
Which framework should be used if aiming to develop a mobile game with multi-modal interaction for the Android platform?
With the follow-up questions of:

- Which available frameworks exist that are complete, simple and free to use for developing 2D games?
- How well-suited is the framework for our project? What are the advantages and drawbacks for each and one of them when comparing?

The framework will be used to develop an original game called *Magnificent Marbles*. The evaluation will be based on how suitable the framework is for the development of a prototype of *Magnificent Marbles*. Criteria for framework selection and evaluation is declared in section 2.2.4 Frameworks.

1.2.2. Scope

The scope of the project is mainly our resources and the game specifications. The project is time-limited to a 16 week long period or 320 work hours for two persons. This includes everything, such as learning Android development, game development and writing the report. Other technical and planning issues are factored in as well.

The scope of this evaluation is limited to concern Libgdx and not the game development of *Magnificent Marbles*. The game specifications depend on our experience with both game and Android development, thus the specifications are made to suit us and our time frame. There is not enough time to have a fully developed game that is user friendly, graphically appealing and stable. The development of a game prototype is the base for the evaluation of Libgdx.

Some sections only concern the game itself, regarding design and technical aspects. These are not involved in the evaluation and included in this report to give more information about the game.

1.3. Statement of Collaboration

Both authors view this project as a very interesting and rewarding one concerning future education and professional aspects. Both are interested in game development and also mobile development for smartphones. We both understand that we need to co-operate to achieve the best results but have decided to have different responsibilities, especially in the beginning and during different start-up phases, to be able to draw advantages of our strengths.

All decisions regarding the game design, development and the report structure are taken together, even if each person makes the first draft in his respective area of responsibility. The goals for the game and report have been defined and agreed upon in advance, before each draft, thus, there are guidelines for each draft. There are room for changes that happen dynamically and continuously throughout the project with constant feedback and discussion between the authors.

Rakiv Ahmed
Rakiv is responsible for shaping up the concept document for the game design. The different preliminary drafts for this report are also written by Rakiv.

Jonas Aule
One of Jonas' areas of responsibility is to do the preliminary and also the more in-depth technical research regarding different frameworks. Most of the actual game development will start with Jonas' initiative.

2. Background

2.1. Games

2.1.1. Magnificent Marbles

The game in question that is going to be developed is an original game by the authors called "Magnificent Marbles". The game has been designed with our scope in mind but also to be able to compete in the Swedish Game Awards, thus be rather innovative, interesting and fun but limited with regard of game time. The focus is on the first 5 minutes of the game by developing tutorial levels. Please refer to *Appendix* 6.1 *Magnificent Marbles – Concept Document* for more information.

iPhone dominates the game market for mobile phones and has very advanced and high quality games that use multi-modal control[2]. The Android Market is behind regarding larger game productions but since we are only two persons developing *Magnificent Marbles*, it should suit us better.

Magnificent Marbles intends to bring back the nostalgic marble playing experience you had when you were a child, with a magnificent twist! This twist consists of using the modern functionality of a smartphone to touch and emulate some hands-on play you had as a kid.

The main concept behind *Magnificent Marbles* is to have different marbles and move them with the flick of your fingers, just as in the playground of young times. The view will be a top-down bird's view with 2D graphics. The magnificent part is that the marbles have different properties and are used for special purposes. For example fire marbles that would be able to burn through wood or water marbles that would extinguish fires.

There will be different modes of interaction for the player to choose from and use depending on the situation. A marble can interact with other marbles in different ways, such as simply pushing them away and absorbing, merging or multiplying with them to create other marbles (Marble Mixing).

2.1.2. Multi-modal games on smartphones

This section is mainly for those that are unfamiliar to the new types of mobile games. It presents some examples of multi-modal games that already exist for the Android platform and describes how they utilize the interface functions of a smartphone. Multi-modal games are games that use other interactive means than a mouse, keyboard or game pad as in regular PC or console games. For smartphone games, it is mainly by touch and/or accelerometers. For other platforms there is also interaction with the eyes, voice or force feedback devices.

[2] iPhone "most successful" mobile games OS.
<http://www.gamesindustry.biz/articles/iphone-most-successful-mobile-games-os>
[Retrieved 2011-04-08] [Published 2009-03-12]

The game Teeter[3], included in HTC devices, such as HTC Desire, is a game controlled by the accelerometer in the phone. Guide a ball through different levels and labyrinths to reach the green goal hole to advance to the next level. The game is very simple in its production level but is very well suited for introducing a new user to multi-modal control with the accelerometer. Teeter also has basic haptic feedback in form of vibrations.

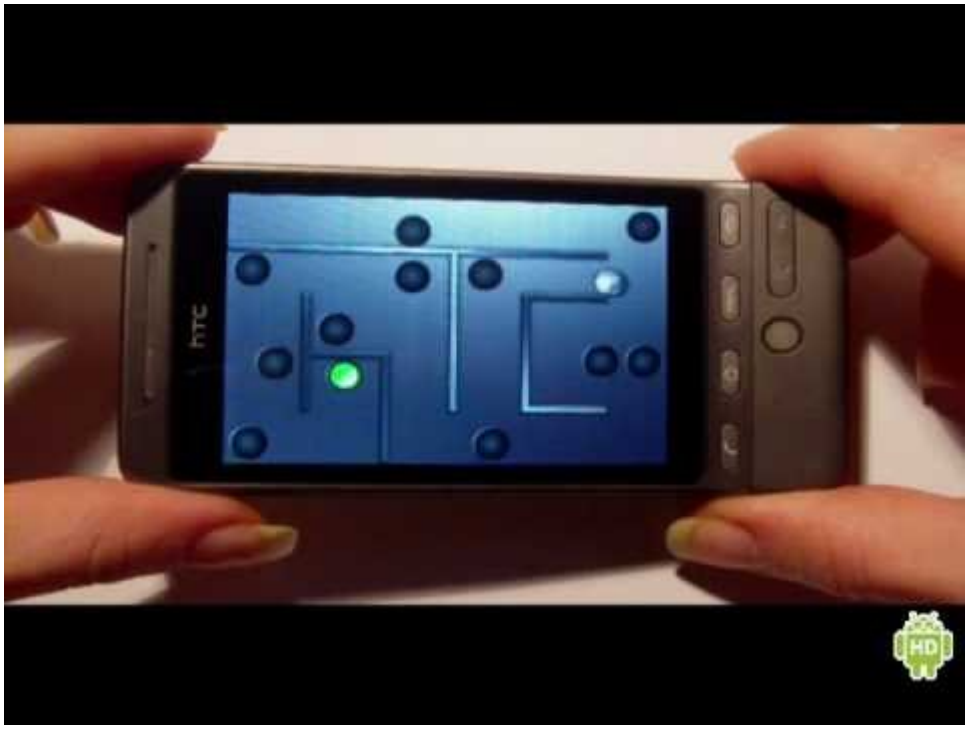


Figure 1. A picture of the Android mobile HTC Hero with the game Teeter. The mobile is being titled and the teeter ball is controlled by the tilt.[4]

[3] HTC's Teeter shows off how physical feedback can immerse you in a game. <http://www.zdnet.com/blog/mobile-gadgeteer/htcs-teeter-shows-off-how-physical-feedback-can-immerse-you-in-a-game/1168> [Retrieved 2011-02-20]

[4] Youtube video image. <http://img.youtube.com/vi/l2o_YKlS6Ns/0.jpg> [Retrieved 2011-02-20]

Air Control[5] is an Android game that utilizes the touch interface in an innovative and intuitive way. Act as a flight controller and steer different airplanes into the different landing strips depending on the plane type. Air Control is not the first game based around this concept, the older and popular iPhone game Flight Control[6] is similar and has also been ported to other platforms. Other comparable games for Android exist as well. Air Control has a unique playing mode called "Cargo Mode" which makes it different from Flight Control. In "Cargo Mode" the goal is to direct planes into specific landing strips determined by the type and color of the incoming plane. The destination landing strip changes over time.

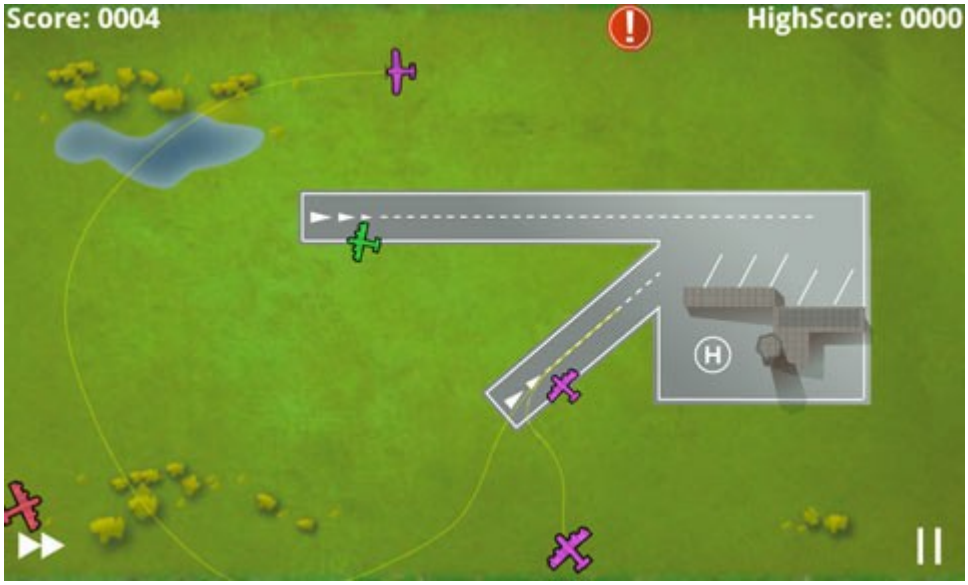


Figure 2. An in-game screenshot from the game Air Control. Cargo mode is played, the smaller air plane icons indicating which type of planes or colors that are able to land.[7]

[5] Air Control Game Review by AndroidTapp.com. <http://www.androidtapp.com/air-control/> [Retrieved 2011-02-20]
[6] Flight Control. <http://firemint.com/flight-control-choose-your-landing-platform/> [Retrieved 2011-02-20]
[7] Air-Control-in-game_Play-6.jpg <http://www.androidtapp.com/wp-content/uploads/2010/04/Air-Control-in-Game-Play-6.jpg> [Retrieved 2011-02-19]

2.2. Software development

Software development tools used when evaluating Libgdx:

- Libgdx; started with the nightly build from 2011-02-18 and then upgraded to 0.9.
- Eclipse 3.5 (Galileo) and 3.6 (Helios).
- ADT version 10.
- Android SDK r10 for Windows.
- Android 2.1 platform (Éclair) and 2.2 (Froyo).

2.2.1. Android platform

"Android is a software stack for mobile devices that includes an operating system, middleware and key applications", quoted from the official Android development site[8]. The Android platform is for the most part open source and based on the Linux kernel. The programming language for applications is Java with some addition of XML for GUI layout. There are different versions of the platform and different mobile manufacturers support them in their own way. Nowadays, most smartphones use platform version 2.1 and newer; up to 2.3[9]. The different platforms also correspond to different API levels, 2.1 being API level 7. For most part the version are backwards compatible, except for some rare security issues.

2.2.2. Eclipse and ADT

To start developing with Android, the first step is to download the Android SDK and follow the instructions on their page[10]. This includes installing the ADT plug-in for the Eclipse IDE as well. The SDK has the source code necessary for building the application and the ADT has the tools needed to compile the code and run it on the emulator. There are different downloads for different SDKs depending on the API level. It is possible to develop Android applications without Eclipse and use another IDE, but not without the same SDK, which is mandatory. The documentation for Android development together with Eclipse is better compared to other IDEs, for example, there are two easy start-up tutorials named "Hello, World"[11] and "Notepad Tutorial"[12].

[8] What is Android? - Android Developers. <http://developer.android.com/guide/basics/what-is-android.html> [Retrieved 2011-01-29]

[9] Android Developers, 2011, Platform Versions. <http://developer.android.com/resources/dashboard/platform-versions.html> [Retrieved 2011-02-13]

[10] Installing the SDK - Android Developers. <http://developer.android.com/sdk/installing.html> [Retrieved 2011-02-13].

[11] Hello, World - Android Developers. <http://developer.android.com/resources/tutorials/hello-world.html> [Retrieved 2011-02-13].

[12] Notepad Tutorial - Android Developers. <http://developer.android.com/resources/tutorials/notepad/index.html> [Retrieved 2011-02-13].

2.2.3. Development without frameworks

There are previous works regarding development on Android, which are documented in the books *Pro Android Games*[13] and *Hello Android*[14]. They describe in detail how to develop games and applications using only Android programming without frameworks. There is also an article called *Developing Mobile Applications on Android*[15] , but in smaller scope of course.

The books describe in great detail on how you can develop a game with Android programming. *Hello Android* only has a simple game example while *Pro Android Games* focuses on game entirely and has advanced 3D examples. *Pro Android Games* also writes about how to use OpenGL and C/C++ API.

2.2.4. Frameworks

There is a sea of frameworks that are more or less adapted for game development. To find the ones that suited us, we had to set up our criteria, such as:

- **a.** The framework should be an open source project.
  - **i.** Free of charge to use for us.
  - **ii.** Without licensing restrictions for the purpose of developing *Magnificent Marbles*.
- **b.** The framework needs to be actively maintained:
  - **i.** It should support the Android 2.1 platform and newer.
  - **ii.** Updated regularly, monthly approximately.
  - **iii.** An active community and other developers using the framework.
  - **iv.** Have an interactive development team that communicate with and respond to the community.
- **c.** The framework must have documentation or an active support platform:
  - **i.** An updated Javadoc should be easily available and accurate.
  - **ii.** Tutorials, test classes and demos should be available.
  - **iii.** Forums or other interactive means of communicating with and getting support from other developers and the framework development team.
- **d.** Performance and functionality:
  - **i.** We should not need to worry too much about performance issues, since we will have other issues to handle because our inexperience with Android development. Our game should run smoothly on mid-end Android phones.
  - **ii.** The framework should have support for basic game functionalities such as menus, GUI components and more.
  - **iii.** For our specific game the framework also need to have a built in 2D physics engine.

These criteria are also what the evaluation of the framework will be based on. More details are given in Section 3 Implementation, with different focus for different development versions of the game, regarding the chosen framework.

[13] Vladimir Silva (2009). *Pro Android Games.* Apress.
[14] Edward Burnette. (2010). *Hello Android*. Pragmatic Bookshelf.
[15] Guiran Chang, Chunguang Tan, Guanhua Li, and Chuan Zhu. (2007)
Developing Mobile Applications on Android. Springerlink. LNCS 5960, pp. 264–286.

The frameworks we found during our one week search period were following:

Name	Website
1. AndEngine	http://www.andengine.org/
2. E3roid	http://www.e3roid.com/
3. Libgdx	http://code.google.com/p/Libgdx/
4. Ardor3D	http://www.ardor3d.com/
5. Rokon	http://code.google.com/p/rokon/
6. Android GL Engine(angle)	http://code.google.com/p/angle/
7. jMonkeyEngine	http://code.google.com/p/jmonkeyengine/
8. Slick2D Android	http://slick.cokeandcode.com/index.php?entry=entry101126-123313
9. JGame	http://www.13thmonkey.org/~boris/jgame/
10. AGE	http://code.google.com/p/age/
11. Juicygames	http://code.google.com/p/juicygames/
12. Cloak	http://code.google.com/p/cloak/
13. Android-2d-engine	http://code.google.com/p/android-2d-engine/

Table 1. List of 13 frameworks for game developong on Android

Our search method consisted of using the search engine Google and searches for different terms and visiting the relevant results. The following terms were searched for:
"developing games on Android", "Android game development tools", "Android game framework", "Android 2D game development", "Android 2D game development framework". We found a top list with many of the frameworks listed, although the list was outdated, it contained good information.[16]

The top 3 (AndEngine, E3roid and Libgdx) became our main contenders that we choose to investigate further. The other 10 frameworks did not fulfill enough of our criteria. The frameworks numbered from 10-13 were immediately disregarded since they were very outdated.

2.2.4.1. AndEngine

AndEngine was the first framework we found and also started to investigate at once, since at first glance it was very promising. It had an updated homepage and an active forum. There were a lot of examples games and demos, so many developers had used this framework. The downside was that there was no official Javadoc online, the source code had to be downloaded and Javadoc be generated locally. Also the Javadoc was not complete.

Since the framework seemed simple enough, looking at example code might have sufficed. Another concern that came up a bit into the investigation was that the performance might not be the best for a more complex game or that we ourselves had to optimize much of our code or even the framework's source code (refer to 2.2.4.3 Libgdx).

2.2.4.2. E3roid

E3roid was another framework that seemed easy enough to immediately start investigating further. In contrast to AndEngine, E3roid is well documented but lacks the example arsenal of AndEngine. The E3roid user base was more limited. As with AndEngine, we were not sure on how the performance would be. AndEngine would be better choice due to the active user base AndEngine has.

[16] Top Ten Open Source Android (2D or 3D) Game Engine […]
<http://www.cuteandroid.com/ten-open-source-android-2d-or-3d-game-engine-for-android-developers>
[Retrieved 2011-02-12] [Published 2010-10-05]

2.2.4.3. Libgdx

We found the Libgdx framework after AndEngine but chose not to investigate it immediately since it seemed more complex. Libgdx is a more comprehensive framework and gives more access to OpenGL ES[17]. The Box2D[18] component of Libgdx is even used by AndEngine and E3roid for the physics engine. Since AndEngine and E3roid were previously judged as good, we decided to investigate their source of inspiration so to speak, even though Libgdx was a bit overwhelming at first.

Libgdx met more than enough of our criteria, the downside would be its complexity. It was even something the main developer of Libgdx mentioned when he compared Libgdx to AndEngine. In an online mail conversation[19] , the main developers behind each framework gave their views of themselves and in comparison of each other. Mario Zechner being the founder of Libgdx and Nicolas Gramlich behind AndEngine.

The main attraction for Libgdx is that it has desktop support that enables to test the game directly on a computer, without having an Android device available or using slow emulators. That feature would make coding and debugging the game simpler. Libgdx had more features than we needed, such as 3D support so it seemed worthwhile for the future to spend the extra time needed for learning the framework compared to AndEngine and E3roid. Performance also seemed promising, especially when compared to AndEngine[20]. There are not many comparisons with different frameworks and this one might not be the most objective either.

There are also plenty of games which have been developed with Libgdx. Here is a short list consisting of games in varying statuses, found in Libgdx forum in the showcase section[21]:

Name	Android Market Website
1. **Wood Olympics**	https://market.android.com/details?id=org.tt
2. **Knockheads**	https://market.android.com/details?id=com.treyygames.knockheadsBeta
3. **Face Hunt**	https://market.android.com/details?id=com.gemserk.games.facehunt
4. **Oceans Unleaded**	https://market.android.com/details?id=com.darkrockstudios.games.oceansunleaded

Table 2. A list of 4 games that used Libgdx framework

[17] OpenGL ES. <http://www.khronos.org/opengles/> [Retrieved 2011-02-23]
[18] Box2D – Home. <http://www.box2d.org> [Retrieved 2011-03-01]
[19] [android-developers] Libgdx vs. andengine.
<http://www.mail-archive.com/android-developers@googlegroups.com/msg133167.html>
[Retrieved 2011-02-19]
[20] Libgdx, Andengine & Rokon Micro-Benchmarks.
<http://www.badlogicgames.com/wordpress/?p=803> [Retrieved 2010-02-15]
[21] Badlogic Game – View forum - Showcase <http://www.badlogicgames.com/forum/viewforum.php?f=16>
[Retrieved 2010-04-08]

2.2.4.4. Other disregarded or outdated

The following frameworks were quickly disregarded without further investigation due not fulfilling our criteria to a satisfactory level and other various reasons.

(The bullet numbers refers to Table 1 mentioned previously in page 18)

4. **Ardor3D**

Unfortunately we discovered Ardor3D after we decided to develop with Libgdx, so we did not have time to investigate this framework. It were similar to AndEngine and E3roid at first glance, although a more professional and for 3D games. It did not seem to have the unique features of Libgdx so it did not seem worthwhile to investigate further.

5. **Rokon**

Rokon seemed to have a good history but turned inactive to the point that the owner wrote ”A final goodbye” on its website.

6. **Android GL Engine**

This framework was active until late autumn 2010.

7. **jMonkeyEngine**

Made for regular Java and adapted to Android later. Mainly for 3D games.

8. **Slick2D Android**

A framework made for regular Java with experimental Android support.

9. **JGame**

A framework made for regular Java with experimental Android support.

The following did not get a preliminary look since they were very outdated.

10. **AGE**
11. **juicygames**
12. **cloak**
13. **android-2d-engine**

3. Implementation

This section is about our development with Libgdx in chronological order. A list of the development tools is provided in the Background section (see 2.2 Software Development). We have defined different versions of the game to easier plan the different stages of the development and focus on the critical features early on.

The method for each version concerning the usage of Libgdx will be described and the results for each version will be shown. Game features developed for each version will be listed along with the focus of the Libgdx evaluation for that version. One could view the different version sections as a method and a result part together, as it is more commonly known. At first there was an *Experimental phase* and then the actual game development started with *Version* 0.0 – Pre-Game. The evaluation ends with *Version 0.1 – Game objects* design.

Computer hardware for development varied but it did not impact the evaluation. The Android phone used for testing was a HTC Desire[22] with Android 2.2. HTC Desire has a screen resolution of 480x800, which is what the screenshots are based on.

3.1. Experimental phase

An experimental phase with the purpose of experimenting with Libgdx and getting started with Android development.

The main focus of the framework evaluation for this version was:

- Setting up and configuring the Libgdx and Android development environment.
- Drawing basics shapes with the provided methods via sprites, textures and meshes.
- Implementing input handlers that are able to process touch input.

Libgdx was fairly easy to install but it was confusing with different versions and unsynchronized tutorials. Libgdx has stable builds and nightly builds. Nightly builds gets updated daily, or nightly. For these builds, maybe not everything is 100% stable but it has all the latest features.

The most recent stable build was released in November 2010 and seemed natural to start with as new Libgdx developers. The available startup tutorials indicated that either the nightly build or the stable one would suffice. Although later it turned out that the tutorial needed some files from the nightly build.

There is a large amount of example code to look at but the problem is that many of the examples are not updated for the latest Libgdx build and therefore does not work without some modification. The same applies to some of the tutorials; they reference classes that have been renamed, moved or changed. It is not always easy to know how different classes should be used together either.

After a first small hiccup of finding and installing the right build, we started with the actual development. Following the HelloWorld[23] and MyFirstTriangle[24] tutorials were easy. Although it became a bit more difficult when we had to look up what to do next for some more advanced features or a more comprehensive application.

[22] HTC – Products – HTC Desire – Overview. <http://www.htc.com/www/product/desire/overview.html> [Retrieved 2011-01-28]
[23] HelloWorld <http://code.google.com/p/libgdx/wiki/HelloWorld> [Retrieved 2011-02-14]
[24] MyFirstTriangle <http://code.google.com/p/libgdx/wiki/MyFirstTriangle> [Retrieved 2011-02-14]

Libgdx provides many test classes and some demos but they lack documentation so they are not so easy to understand. After a few hours we were able to draw some shapes and text on our screens. Since we were unfamiliar with OpenGL it took us some time to be able to draw simple objects using the Libgdx classes such as Pixmap, Texture and TextureRegion. Figure 3 displays our first screen developed with Libgdx.

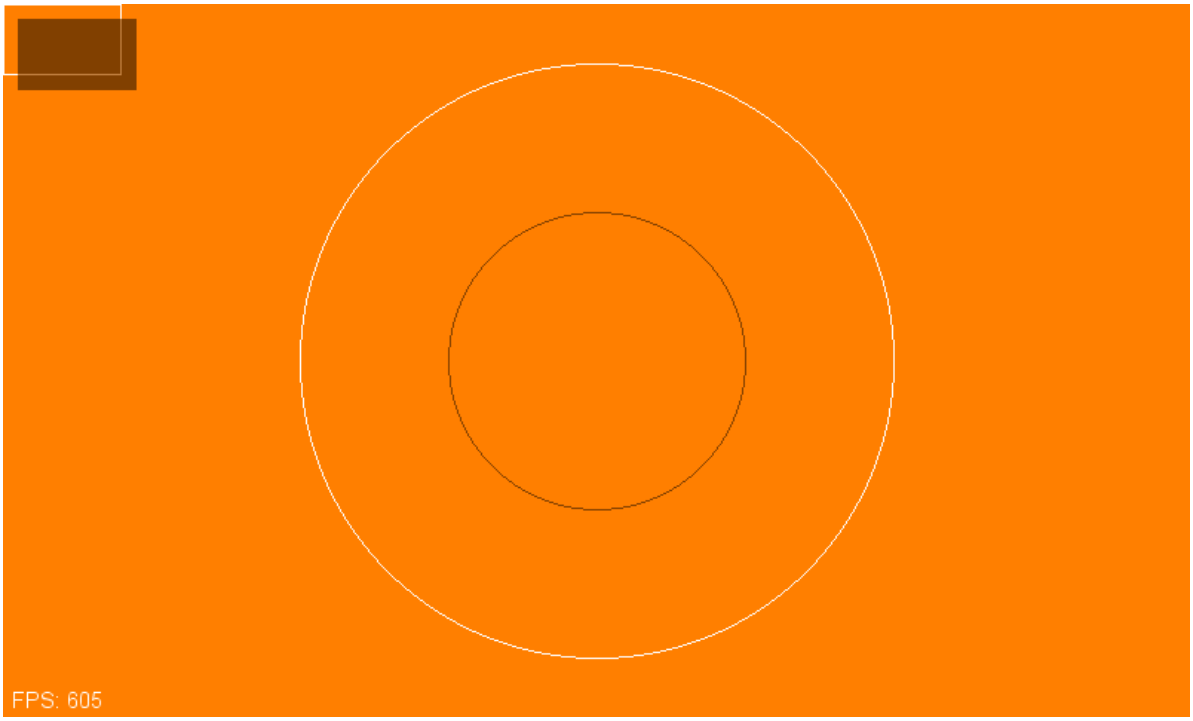


Figure 3. Image of our first Libgdx application

We noticed a bug and reported it. It was in the Pixmap class, which did not differentiate between the drawCircle and fillCircle methods. We had to dive into the source code and find out some interesting details about how Pixmap worked. Since Libgdx is an open source project and relies on users to improve it, we of course decided to report this bug[25] and let the developers be aware of it. The Libgdx developers responded quickly within 30 minutes, after fixing the bug in the latest nightly build. After downloading the latest version of Libgdx we could confirm that the problem was indeed fixed.

We wanted to be able to handle input as soon as possible, since it is a very important part of a game. Libgdx supports two different ways to handle input, event-based and polling-based. With polling-based input each step in the game update loop the game calls a method in the Input class such as isKeyPressed to check if a certain key is pressed right now or isTouched to check if the screen is touched right now.

For event-based input either implements the InputProcessor interface and provide implementations for the various methods such as keyDown and touchDragged or if not wanting to handle all types of events, extend the InputAdapter class which provides empty handlers for the things not implemented. The framework then calls these methods when an event of that type occurs.

[25] Issue 151 - libgdx - Pixmap.FillCircle dont work <http://code.google.com/p/libgdx/issues/detail?id=151&can=1&q=drawcircle> [Retrieved 2011-02-21]

When we added a moving circle the game slowed down dramatically. The reason for the slowdown was the naive code we were using which was drawing the whole screen from scratch each frame. Using sprites, not rendering everything from scratch each frame as well as using the SpriteBatch and SpriteCache classes for graphics that changes every frame and more static graphics respectively speeded up the game again.

Performance optimization is as already known rarely straightforward. Trial and error along with benchmarking (FPS) is used to see what works and what does not. The performance characteristics vary between desktop Java, the Android emulator and an Android device.

Another issue was that different coordinate systems were used in different parts of Libgdx. Sometimes the origin of the axes was in the bottom left corner with the y-axis pointing upwards and sometimes in the top left corner of the sprite pointing downwards. When drawing Meshes the origin was even in the center of the screen. This caused quite a bit of confusion and extra work to get everything in the correct place on the screen.

3.2. Version 0.0 – Pre-Game

This version is for implementing fundamental features without any actual gameplay experience. The purpose is to try out Android and the Libgdx framework. Fundamental concepts of the game will be introduced too. A simple view of one fixed activity is enough for testing purposes. The following list describes which features are included:

- Have a round circle.
- Be able to touch and move the circle around.
- Be able to push the circle with flicks (touch of a finger).
- Have some moving physics.
- Introduce elemental materia.
- Let the circles absorb the materia and evolve into marbles.
- Have static marble images (about 8-16).
- Be able to add new marbles (select, random or neutral) (debug feature).
- Let the marbles push each other.
- Have neutral materia so the marbles can multiply.
- Let the marbles merge with each other.

The main focus of the framework evaluation for this version was:

- Getting started with Libgdx and actual game development.
- Testing the desktop and Android support link.
- Utilizing OpenGL and creating more advanced graphics via sprites, textures and meshes

A new stable Libgdx version was released during our previous phase and we decided it was time to upgrade to Libgdx version 0.9. Most Libgdx examples seemed to work after the upgrade.

Box2D support was added to the game, as show in Figure 4. The marbles can now be pushed around by touching/clicking somewhere other than the marbles. The game will create a square box in that location that represents a finger. Then the square gets dragged around and pushes the marbles with it. The actual box will be invisible for the player in the real game. In Figure 4 , there are no actual marble game objects, merely a circle representation in Box2D. The circles shown are debug graphics used to align the real game object graphic.

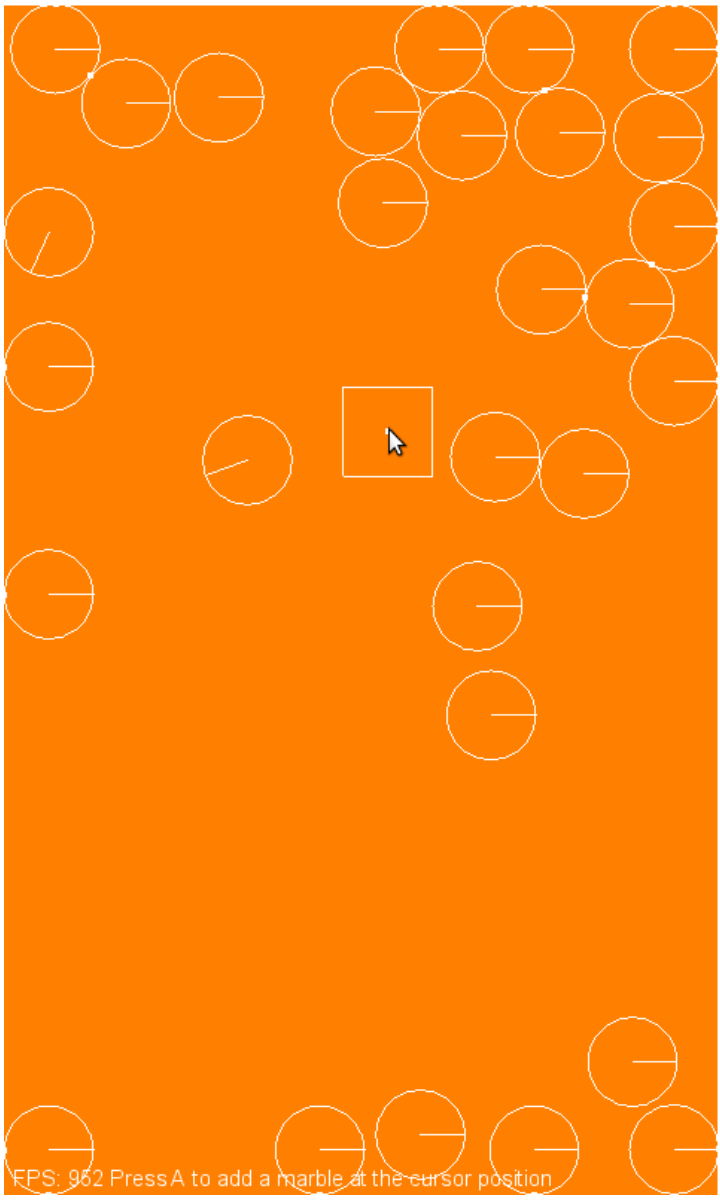


Figure 4. First version with Box2D support

We refactored the input handling code to an InputHandler class; responsible for reading input such as touch (or mouse, depending on platform) and keyboard. InputHandler also takes appropriate actions depending on what input was received. An example of such action is pushing physics bodies around or dragging them around when clicked on.

Major refactoring in preparation for future features, split big classes into smaller new classes. Added some physics debug graphics. Libgdx provides with a class for easily drawing debug graphics which can more precisely display where the different Box2D objects are located. This is useful to have when adjusting game graphics and logic. The debug graphics can be seen in Figure 4.

After the upgrade to Libgdx 0.9, where the drawCircle and fillCircle bug mentioned earlier had been fixed, we were able to fill our marbles, represented by circles, with colors now. Figure 5 shows simple marble graphics with a solid black filling.

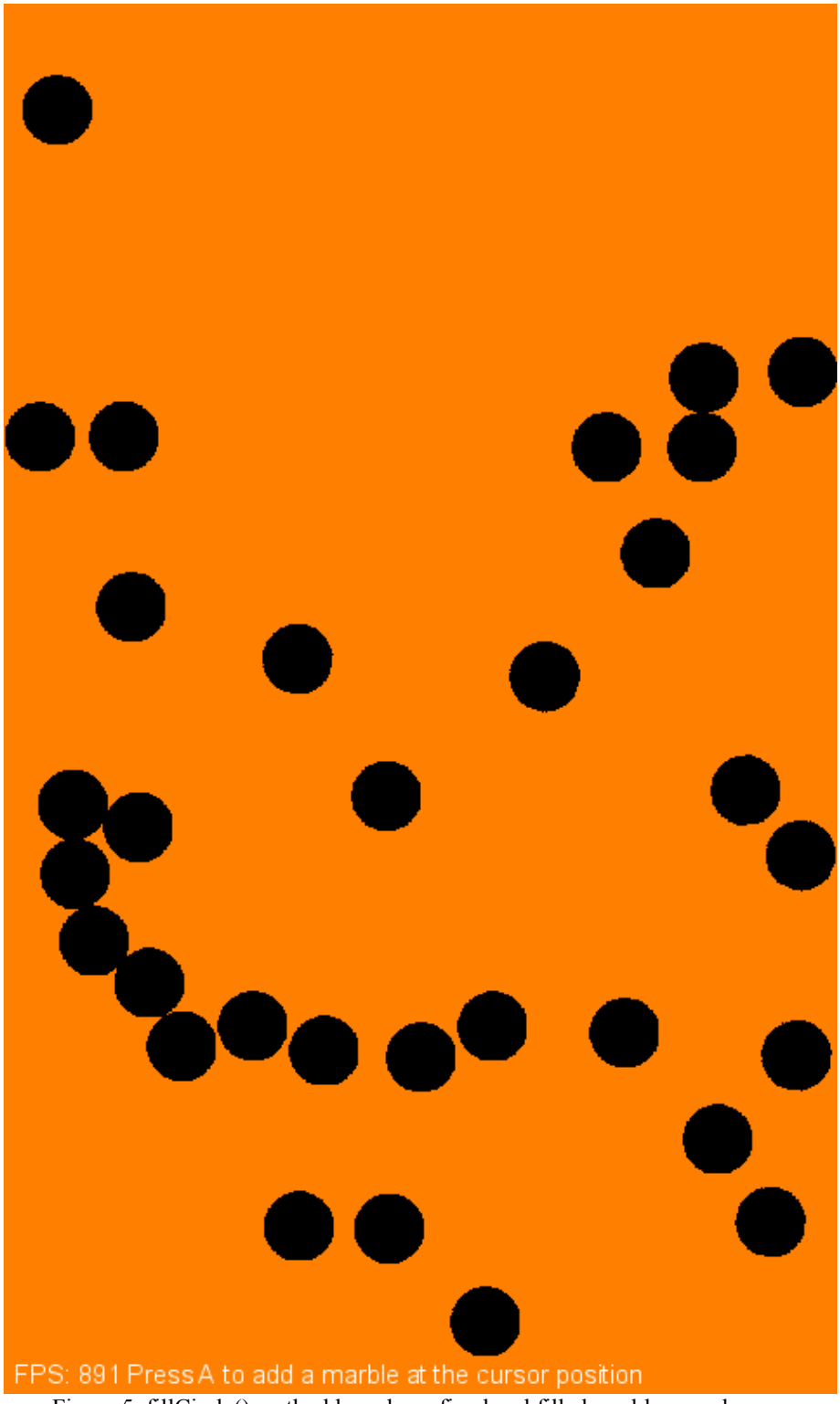


Figure 5. fillCircle() method have been fixed and filled marbles are shown.

We worked on getting the graphics to draw in the correct places. Different coordinate systems are used in the Box2D world and the actual game screen. We needed to recreate unmanaged graphical resources, such as textures and pixmaps, yet another complication or overhead to keep track off for the programmer.

Figure 6 shows that the marbles are colored but do not accurately show that the creations of the sprites are managed incorrectly. Sprites got created on each frame update, several times per second, and it also created a disco effect where the marbles changed color all the time. The figure depicts this by showing that some of the marbles have two colors, a screenshot taken at the moment of sprite and color change.

Figure 7 shows the disco effect more clearly, since it is taken with a camera not dependent on the FPS or simulation. Different elements are supposed to have their own color, not being mixed.

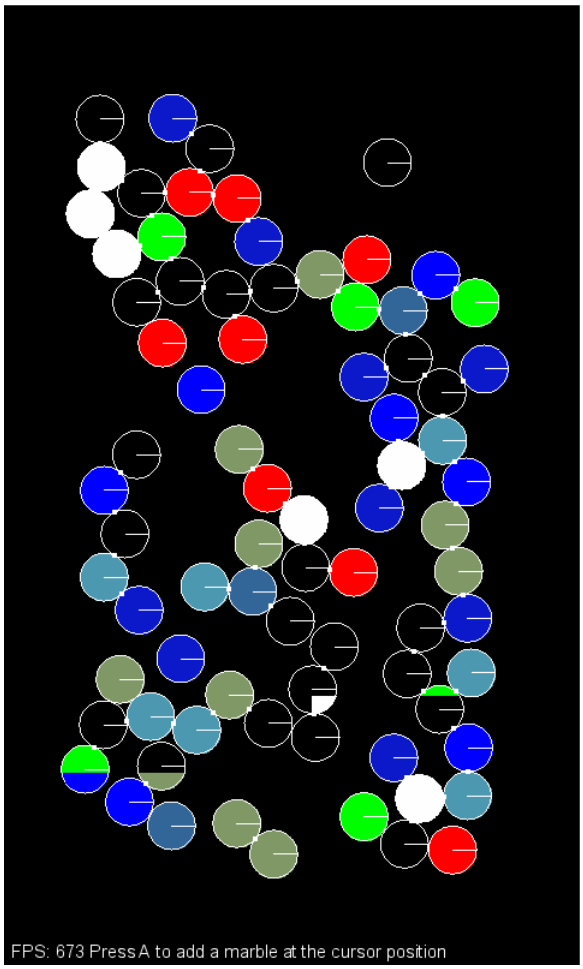


Figure 6. Marble showing their colors but sprites not are created in the correct way – Disco Marbles. (left figure)

Figure 7. Disco Marbles visualized more clearly with a camera shot, independent from the FPS. (right figure)

We added elemental materia, which can be picked up by rolling marbles over it. In Figure 8 the material is shown as small circles. Element colors are the same for marbles and materia.
Figure 9 shows all the marble types and their materia. These figures have no impact on the evaluation and are displayed only for visual reasons.

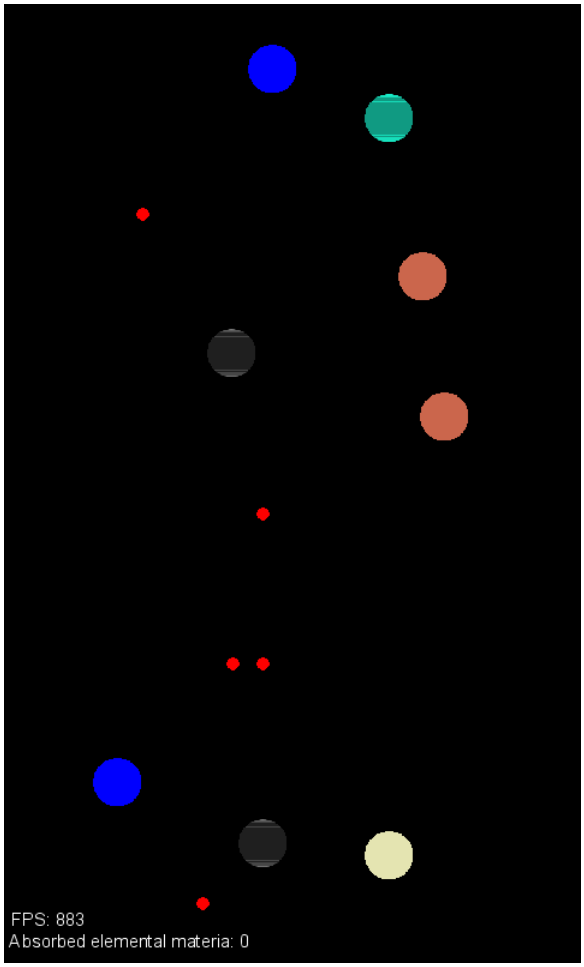


Figure 8. First appearance of elemental materia, as small circles. (left figure)

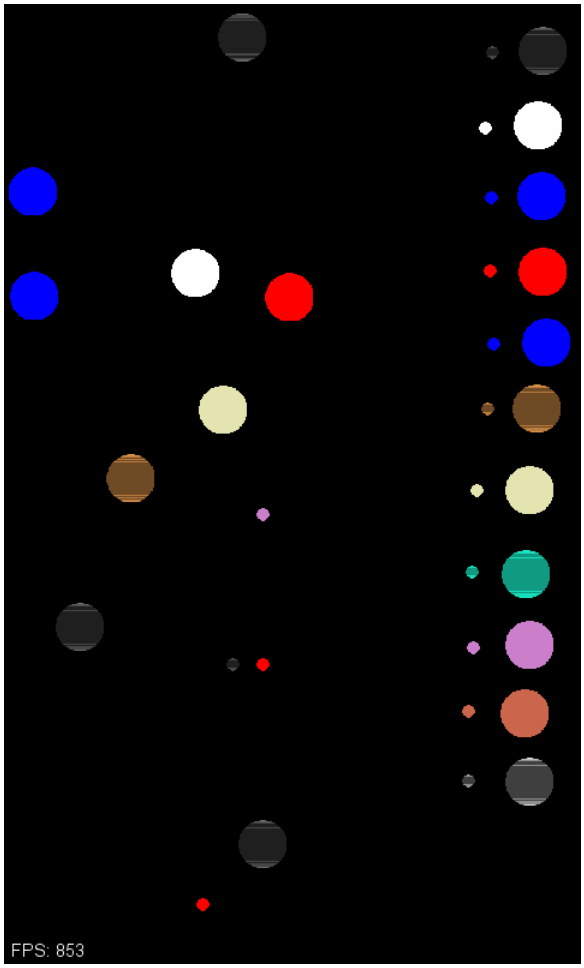


Figure 9. Marbles and their respective material lined up at right side. (right figure)

Fixed several bugs such as regeneration of sprites when resuming the game after it has been paused, a complication mentioned earlier. Made the different elemental Marbles have differently colored sprites. Not as easy as first thought, since there is a connection between the Marble object with a Box2D Body object, so the right Marble gets the right colored sprite and not just random flashing disco marbles.

General note: It is easy to crash the program by calling methods in the wrong order or at the wrong time, Box2D related methods in particular when previously inexperienced with it.

Implemented the ContactFilter interface from Box2D, which decides what collides and reacts with. Organized most of the physics related interaction into the Physics class. Materia sprites also needed to be recreated to prevent crashes. The lack of a streamlined method to always recreate sprites is another overhead the developer needs to keep track of.

Fixed InputHandler controls so it works better with an Android. Libgdx does not seem to have any streamlined way to differentiate between Desktop and Android control. Not for output either for that matter. A way to work around this is to include Boolean variables in the constructor when creating a desktop or an Android application. That Boolean would keep track on different input and output actions.

Performance seemed acceptable on a HTC Desire with about 50 objects and sprites. More optimizations need to be done though, since we only developed for screen-sized levels at the moment. Levels in the future might be larger, at least 3-4 screens. Refer to Figure 10 to view the objects we tested with.

Figure 10 shows that about 50 marbles can fit into a screen and not slowing down the game play on a HTC Desire. The marbles also got updated sprites, where sprites are created different for each marble but colored according to their element. The amount of active objects involved seemed promising for future development.

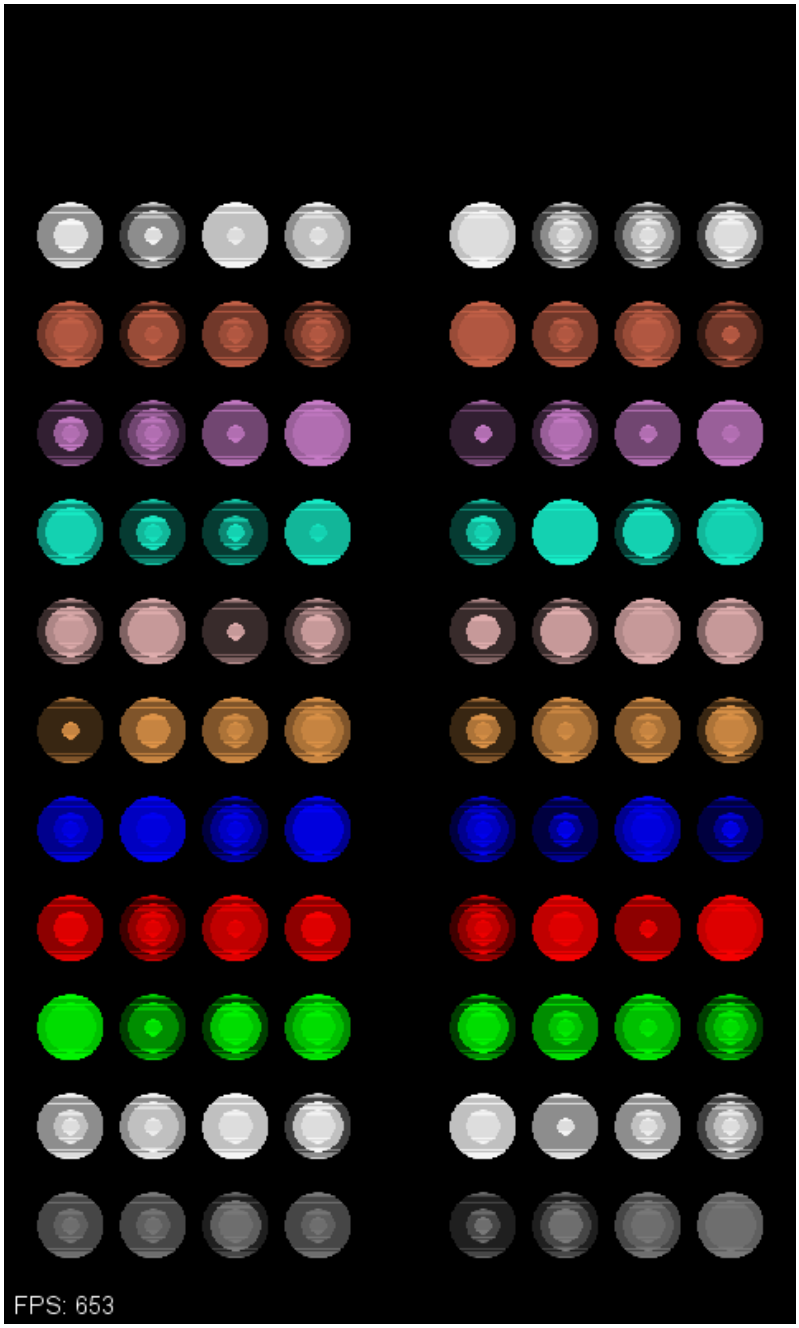


Figure 10. Ca 50 marbles on the same screen without slowing down.

3.3. Version 0.1 – Game objects design

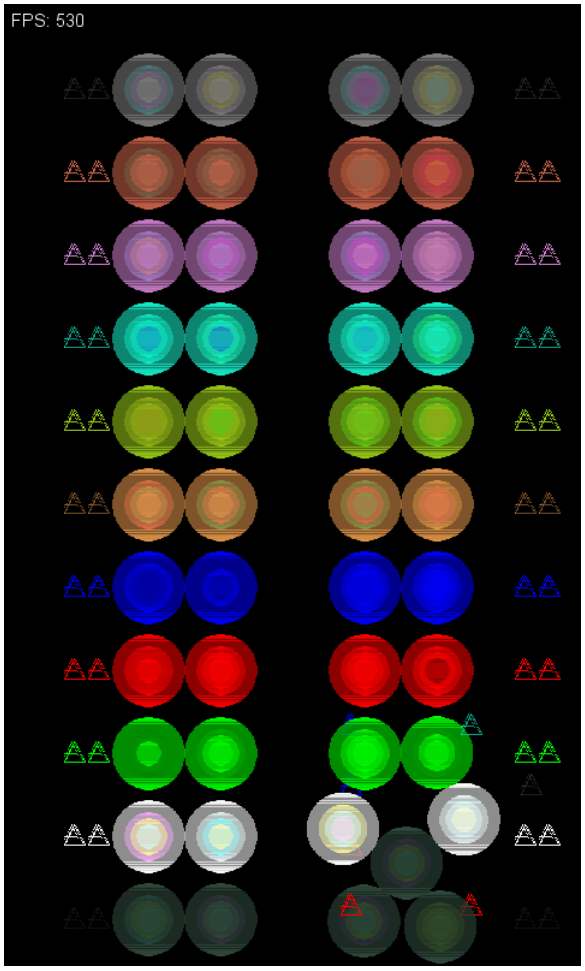
This version is about trying to make game objects for the game. Lots of different game objects and level features are needed for the stages. Many screenshots will be provided to show the different stages and various objects. The Tiled Map Editor support of Libgdx was tried out.

Performance regarding having many different objects is also tested. This version does not deal with many issues or features related to the evaluation of the Libgdx framework otherwise. Instead it is about implementing the different objects need for the tutorial stages mentioned in the concept document (refer to *Appendix* 6.1 *Magnificent Marbles – Concept Document*). The following game features were implemented in this version:

- New game objects:
  - Game object emitter (generator)
  - Holes
  - Barriers
  - Elemental Barriers

The main focus of the framework evaluation for this version was:

- Having a prototype runnable on an Android device without lag, i.e. performance issues.
- Trying out Tiled Map Editor support.



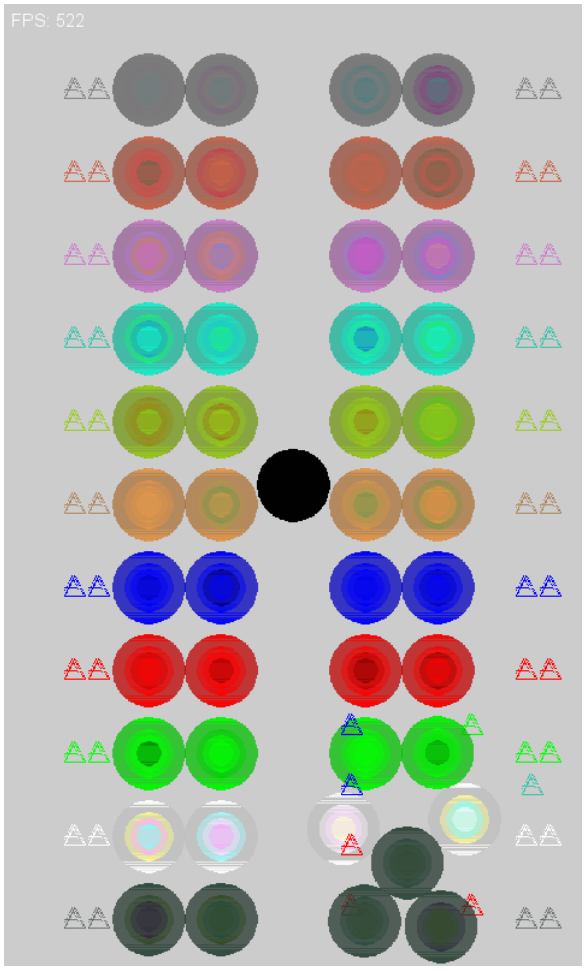


Figure 11. Marbles are bigger and materia is triangular (left figure)

Figure 12. Changed background and introduced a black hole that makes marlbes disappear (right figure)

Figure 11 and Figure 12 displays the contrast of having different background colors, a black respectively white background and the introduction of a black hole. This hole had to be implemented so it destroyed Box2D body usage related to the marble it swallowed.

Implemented very basic loading and creation of some types of game objects from a Tiled .tmx file. However after testing the Tiled Map Editor we did not like its workflow. The most commonly done actions are inconveniently located and do not have shortcut keys. There are many similar UI details in the Tiled GUI, which makes it slow and not very pleasant to use.

It is hard to define reusable objects to place, the properties for each object should be in a sidebar for quick access but instead switching to the select tool, then right clicking the object to edit and choosing an option in the right click menu is needed. Constraints cannot be defined for using on the values of object properties or set some properties as required for all objects.

Since Libgdx has support for Tiled (.tmx) map files we do not have to write the code to parse the XML it contains, instead we can access the objects and their properties as well as other things in the map file in an object-oriented way. However the code for creating the game world with its objects from the map file has to contain a lot of checking that properties exist and that they are not null. This makes it tedious to write and later read the level loading code.

In general the Tiled editor was not a good fit for our project since we did not have a rigid tile-based grid but instead more freely placed objects. Another option that surfaced for having a level editor is the Ogmo Editor[26] or maybe making a custom in-game editor instead. The advantage of the Ogmo Editor is that game objects can be defined in a XML file beforehand, with certain constraints on the properties each object should have and which values that property is allowed to be set to.

There is a clear advantage of having these game objects placed using the Ogmo Editor GUI. The advantage of this is that it minimizes errors in the level construction process and that the code for loading the levels does not have to check for properties that does not exist nor has incorrect values meaning shorter and simpler code.

We tried to put the graphics and levels folders in the assets folder in the Android project and linked the folder in the Desktop project so they work in both projects without needing to duplicate them all the time like in a tutorial mentioned in the Libgdx blog[27] but we could not get it to work. So for now we have two copies of the files, one in each project and have to keep them synced, which is another overhead for us.

[26] Ogmo Editor. <http://www.ogmoeditor.com/> [Retrieved 2011-03-31]
[27] Linking assets between desktop and android project in Libgdx […] <http://www.badlogicgames.com/wordpress/?p=1537> [Retrieved 2011-03-31]

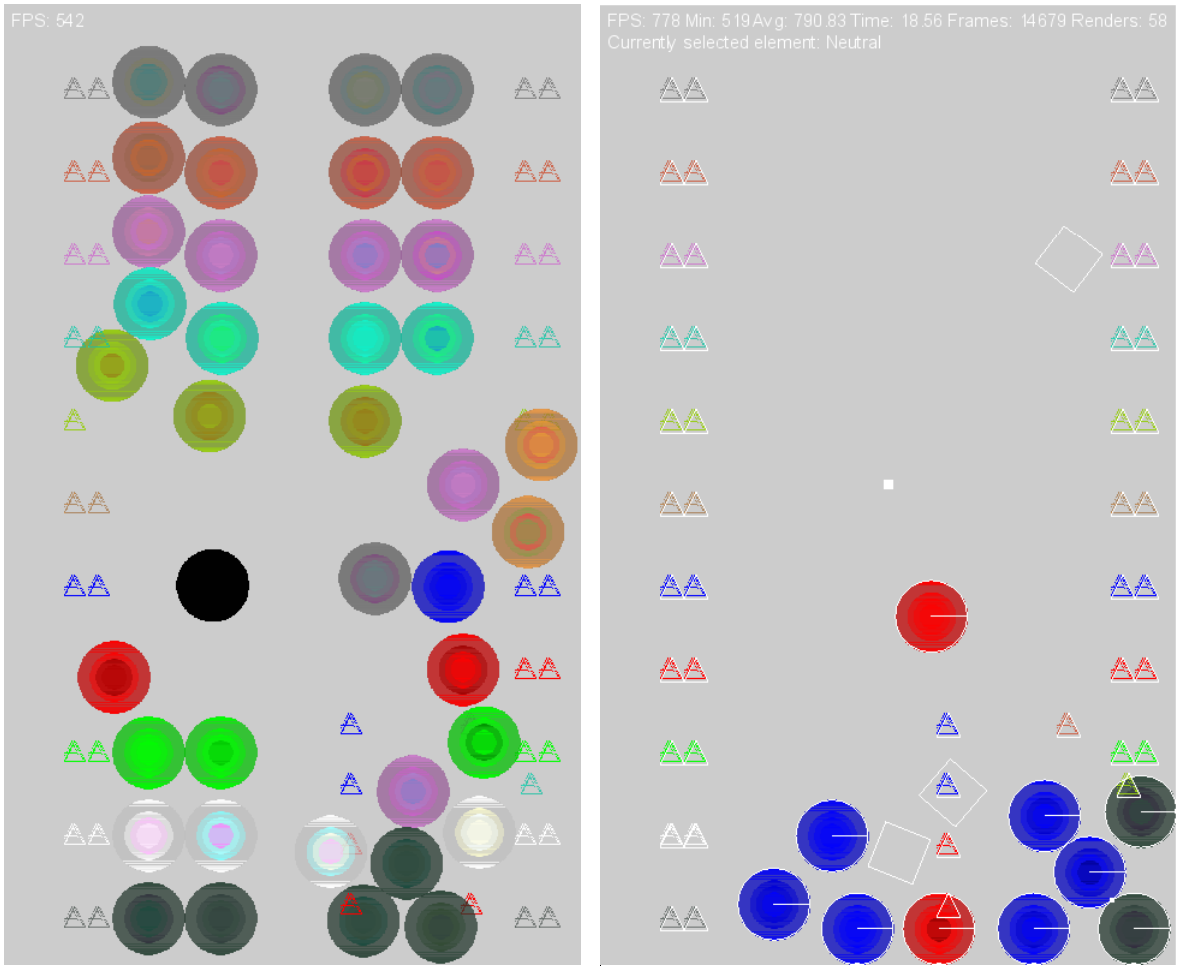


Figure 13. Invisible (without debug graphics) bodies that moves the marbles around. (left figure)

Figure 14. Object emitter repsented by a small solid white square in the center. It emits different objects according to level (right figure)

Figure 13 and Figure 14 displays the status of the game development. More game objects have been implemented and are shown. After the initial implementation of Box2D with the marbles, adding more game objects became more streamlined. Interaction between different game objects were also made easy due separation of physics with Box2D

The framework evaluation concern was the performance issue of having many different objects at the same time and also multiples of them. The FPS is around 30 on a HTC Desire, which is acceptable at this stage. Note that the FPS numbers shown in this report are based on the desktop version of the game, which is enabled with Libgdx.

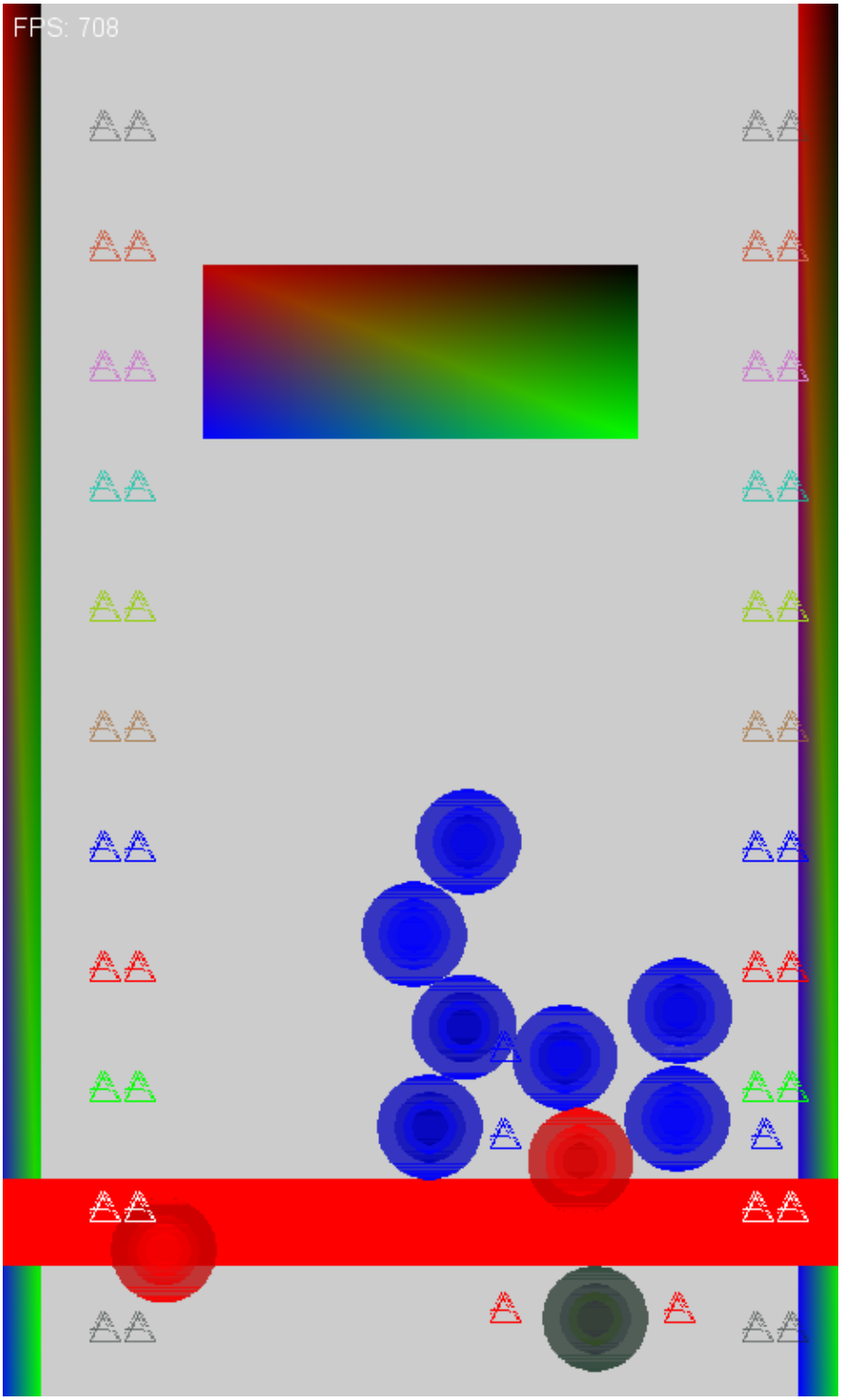


Figure 15. These game objects concludes all the objects needed to start making interesting levels.

The final development stage for this evaluation is shown Figure 15. Added the red elemental barrier (rectangle) near the bottom of the figure and the three other gradient walls, two at the sides and one near the top. This is not the final version of *Magnificent Marbles though*. More features and usage of Libgdx is to come concerning the game but those issues are outside of this reports scope.

3.4. Version 0.2 – Level Design

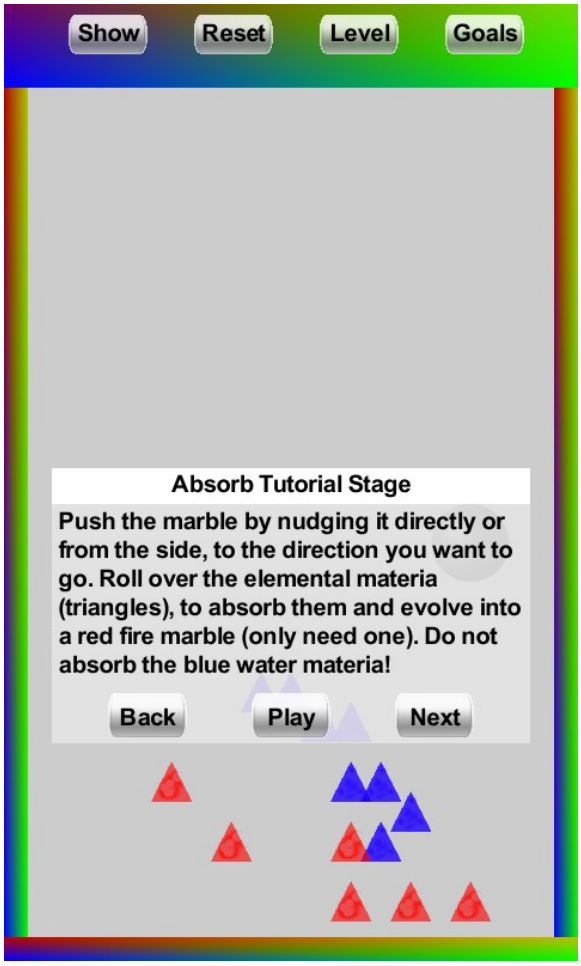
This version is about creating levels using the previously implemented game objects. This version does not deal with the evaluation of *Libgdx,* instead it is purely concerning the game design. As stated in the Concept Document (see Appendix), the focus of the game development at this stage would be on the beginning levels, also mentioned as tutorial stages. Other levels will also be implemented.

A level is only made up of one screen, so you can see everything at once without scrolling around, giving the player a better overview. A level comes with some instructions telling the player what to do, along with the level's goal. A level can also have a time or touch count limit.

Tutorial stages for the different aspects of the game are created, for giving the players an introduction to the various game elements without overwhelming them at once. The game starts with these tutorial stages and then advances to the regular levels. The levels are the following, in the order they appear:

Level Name	Description
1. **Absorb Tutorial Stage**	Learn the absorb aspect of the game by letting a simple neutral marble absorb fire materia and evolve into a fire marble.
2. **Multiply Tutorial Stage**	This stage is about multiplying your marbles by letting them absorb several elemental materia of the same type as the marble itself.
3. **Evolve Tutorial Stage**	This stage is about evolving fire and water marbles into steam marble by having them interact with each other or elemental materia.
4. **Barrier Tutorial Stage**	Move your way through to the hole.
5. **Hole Tutorial Stage**	Create many marbles and let the holes swallow them.
6. **Elemental Barrier Tutorial Stage**	Create the right type of marble to get pass the elemental barrier and to the hole.
7. **Moving holes**	Watch out for the moving holes and survive.
8. **Keep it burning**	Keep your fire marbles safe by avoiding the water materia.
9. **Metal Marble**	Create a metal marble by using the knowledge of marble mixing you have acquired earlier.
10. **Evovle Three**	Evolve into three different elements, so you learn which elements evolve into what.
11. **Muddy weather**	Create mud marbles and pass them through the elemental barriers.
12. **Evolve Three 2**	Evolve into three different elements again but other starting and goal elements.

Table 3. List of levels for Magnificent Marbles.



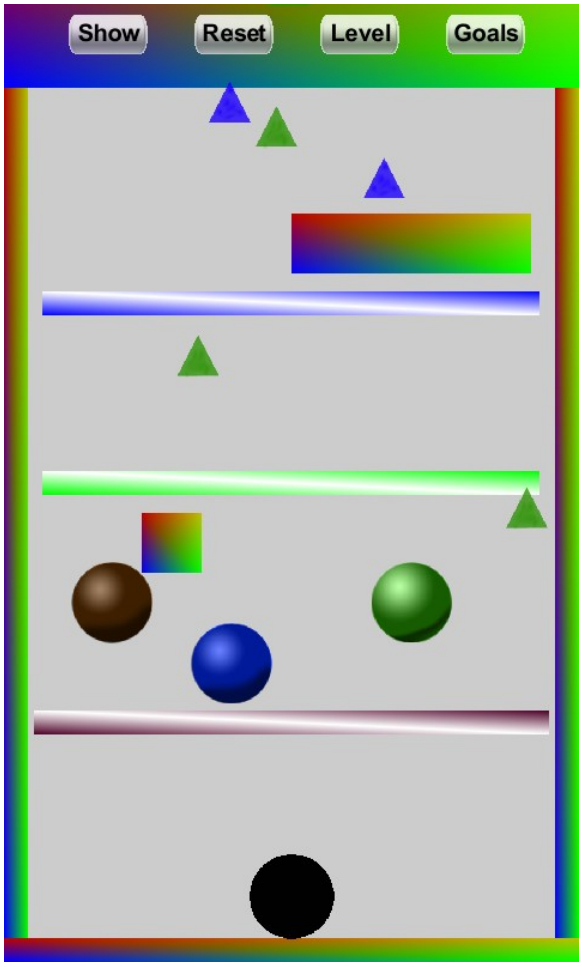


Figure 16. Displaying the dialog box with level instructions at the start of a level. In this case level 1 – Absorb Tutorial Stage. (left figure)
Figure 17. How a level (level 8 – Muddy Weather) looks without the instructions dialog. (right figure)

The first level of the game, Absorb Tutorial Stage, is shown in Figure 16 , which is very simple even if there is a some text to read. The level is about introducing the absorb aspect of the game, hence the goal is to absorb some materia, fire in this case, and evolve your neutral marble into a fire marble. The following tutorial stages are in a similar fashion, being as easy as possible.

Muddy weather is a later level when the player is more accustomed to the game. It is pretty difficult since you need to avoid the moving elemental materia that is emitted from the top. Since water and earth can both become mud and vice versa, you need to absorb the right materia at the same time avoiding the wrong ones. There is also three elemental barriers, limiting the movement of different marbles. Notice the buttons at the top as well. The buttons sprites and GUI layout were implemented during version 0.3 – GUI Design (see section 3.5)

Each level has its own goal or a set of goals that the players must achieve. These goals are described in-game with level description text or with a list of goals. A goal is made of several components resulting in many different combinations of a single goal, and a level can have several of these combined goals. The different components of a goal consist of:

- Amount of marbles (1-30)
- Type of element (11 choices)
- If it should apply to alive marbles or marbles swallowed by holes (2 choices)

The goals are implemented in a similar fashion to the Builder pattern[28]. You create a goal object in the level class, give the goal its name and description and then start to build it by setting the different attributes mentioned above. The main different from a regular Builder pattern, is that there is no private builder inner class of a Goal, instead each time you call a set method for a goal, the set method returns the Goal itself, thus enabling you to continue building the goal. All the different combinations of goals is then handled within the Goal and its checkGoal() method.

Different goals can then in each level be categorized into four types of goals by adding them to the respective list. The Level object checks these lists and judges them accordingly to decide if the goals have been reached or not. The following types of goals exist:

- Regular goals: This is what you have to reach to fulfill the goal(s) at any time during the level.
- End-goals: This is what you have to have at the end of a level when the time runs out.
- Non-goals: These are "goals" you should not reach, if you do then you have to restart the level.
- Ever-goals: These are goals you should keep fulfilled at all times; otherwise there will be a reset.

To have more advanced levels, instead of only using generic combinatorial goals, one can code the levels so certain goals and objects are linked with each other, by having these objects as a field of the level. For instance in the "Elemental Barrier Tutorial Stage", once you get your first marble into the hole, more elemental materia is emitted making it more difficult for the player to create the right marble to pass through the elemental barrier. Timers can also be introduced to set certain object on after a set time, for example in the "Keep it burning" level, more water materia is emitted after a while. All these level design methods makes it possible to create customizable and different levels given enough development time.

[28] Builder pattern – Wikipedia […] < http://en.wikipedia.org/wiki/Builder_pattern >
[Published 2011-05-26] [Retrieved 2011-06-14]

3.5. Version 0.3 – GUI Design

This version focused on making the game more user friendly by adding a GUI that displays help screens and other information to the player. Previously the game were tested on the PC using keyboard and mouse input.

We created our own simple dialog boxes for showing text to the player (as seen in Figure 16), this was easier than using a full-blown GUI library such as TWL[29] which Libgdx also supports. Libgdx itself provides only a few very simple classes for GUIs, a text label and a button.

Other GUI elements that we added was a little text that appeared near the top of the screen when something happened that we wanted to inform the player about such as when two marbles merged or a marble evolved into another type. Several of these could be shown at the same time forming a vertical list but making sure not to show too many of them at the same time blocking the view of the game too much.

The perhaps most important GUI element is the new bar at the top of the screen (see Figure 19) with four buttons on it that is always available. The "Show" button shows a screen with an overview of the different types of marble elements in the game. The "Reset" button takes you back to the beginning of the level if you're stuck. The "Level" button shows the same dialog as at the start of the level with a text describing what to do in this level. The "Goals" button shows a list with the exact goals of the level.

So all of these buttons are for providing help for the player, in one way or another. There was however some issues with it, many players didn't use it much on their own instead trying to get along without it for some reason. One reason could be that they focused on the gameplay when there was a lot happening on the screen at the same time, making it hard to concentrate at the GUI at the same time.

Much time was spent making small changes to the positioning of different GUI elements in the code and then running the game to see what it looked like. This would likely have been easier with a graphical tool for creating GUIs, such as the one that comes with TWL. This is something we might try in the future.

The GUI was one thing that needed much testing on an actual Android device, running it on the desktop was not enough. This was because of things like text readability and the size of buttons to make them easy to press that was very different on a computer with a large screen and a mouse compared to a smartphone.

[29] TWL - Themable Widget Library < http://twl.l33tlabs.org/> [Retrieved 2011-06-14]

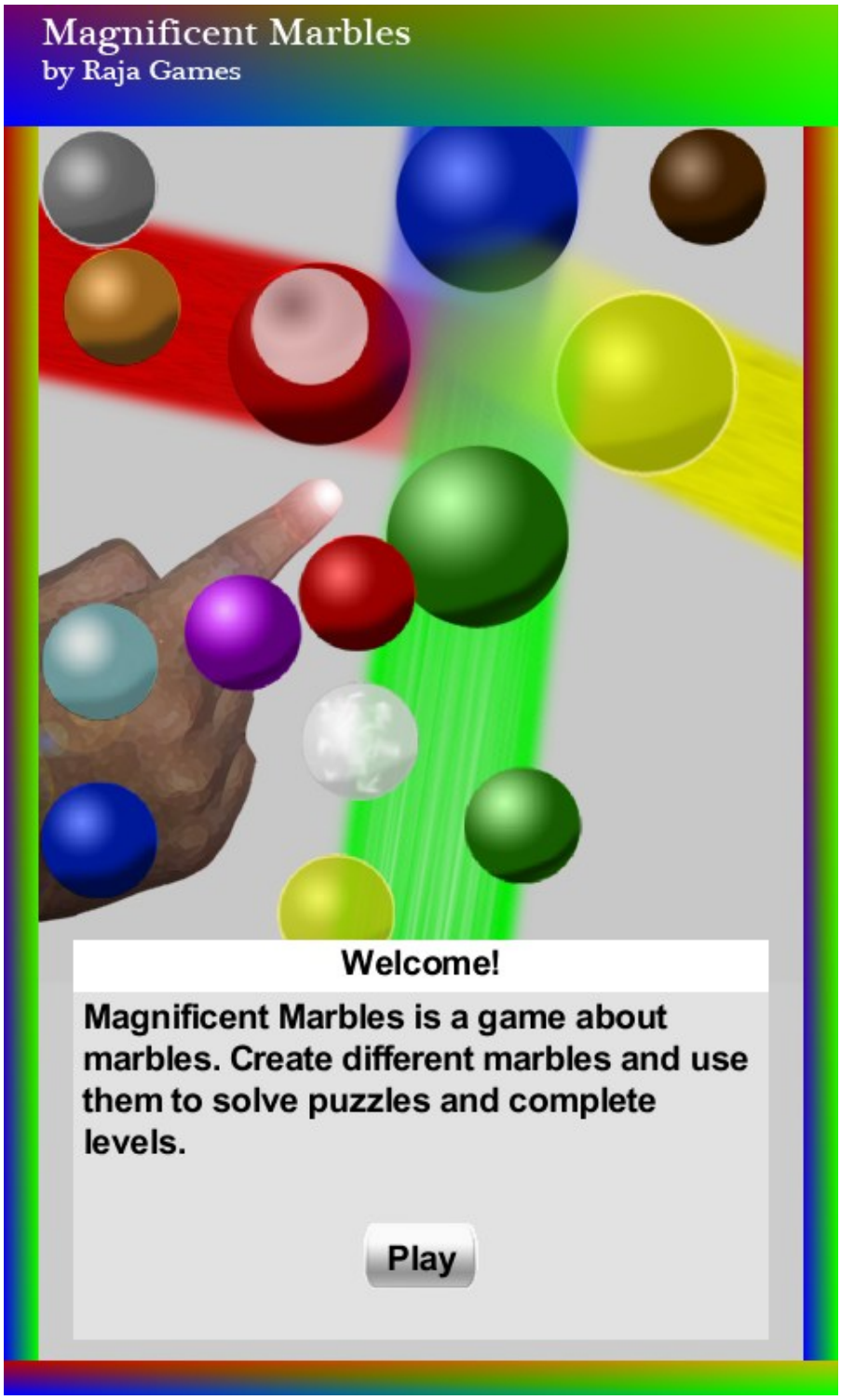


Figure 18. The looks of the splash screen at the end of GUI design. Shown when the application is started.

3.6. Version 0.4 – User testing

This version describes how the demo was developed with the help of user testing. The test consisted of letting five different people play the game for 10-20 minutes, while the developers noted how the players played and responded to different game elements. The demo also evolved from test to test when implementing the suggested improvements from the players. The players were chosen according to the "hallway testing" method[30], in other words anyone willing and available. Briefing from each test session regarding issues we could improve is displayed below in a chronological order:

Test person BC	
Gaming experience:	Years of experience with mainly PC games.
Mobile/smartphone game experience:	Limited
Player feedback	**Our response**
The reset button should have a confirmation button.	Maybe, but might be as well as annoying to have to confirm all the time when you actually want to reset. Optimal would be to have an option for it.
Be able to start "touching" upon barriers.	Fixed this bug.
Be able to drag marbles around.	Implemented this new feature.
Moving holes in level 7 is too fast.	Adjusted, made the holes move slower.
Show remaining touches and time left.	Implemented.
Have a list of the actual goals for each level.	Implemented.
Add a level so you can learn which elements mix with each other.	Added levels 9, 10 and 12.

Test person ME	
Gaming experience:	Very limited
Mobile/smartphone game experience:	None.
Player feedback	**Our response**
Confusing which color represents which element.	Added a show screen which displays all the marbles, materia and their element name.
Show the respective goals at the beginning of a level	Added a button for showing goals due screen space restrictions.
Text at the bottom is not visible	Made the text larger and moved it to the top.
Moving holes in level 7 is too fast.	Adjusted, made the holes move slower.
Did not understand that marbles can both mix and evolve	No easy solution as for now. Maybe more tutorial stages in the future.
Did not use the buttons at the top until notified by developers.	Is probably because of lack of gaming experience.

[30]Usability testing – Wikipedia […] <http://en.wikipedia.org/wiki/Usability_testing#Hallway_testing > [Published 2011-05-29] [Retrieved 2011-06-14]

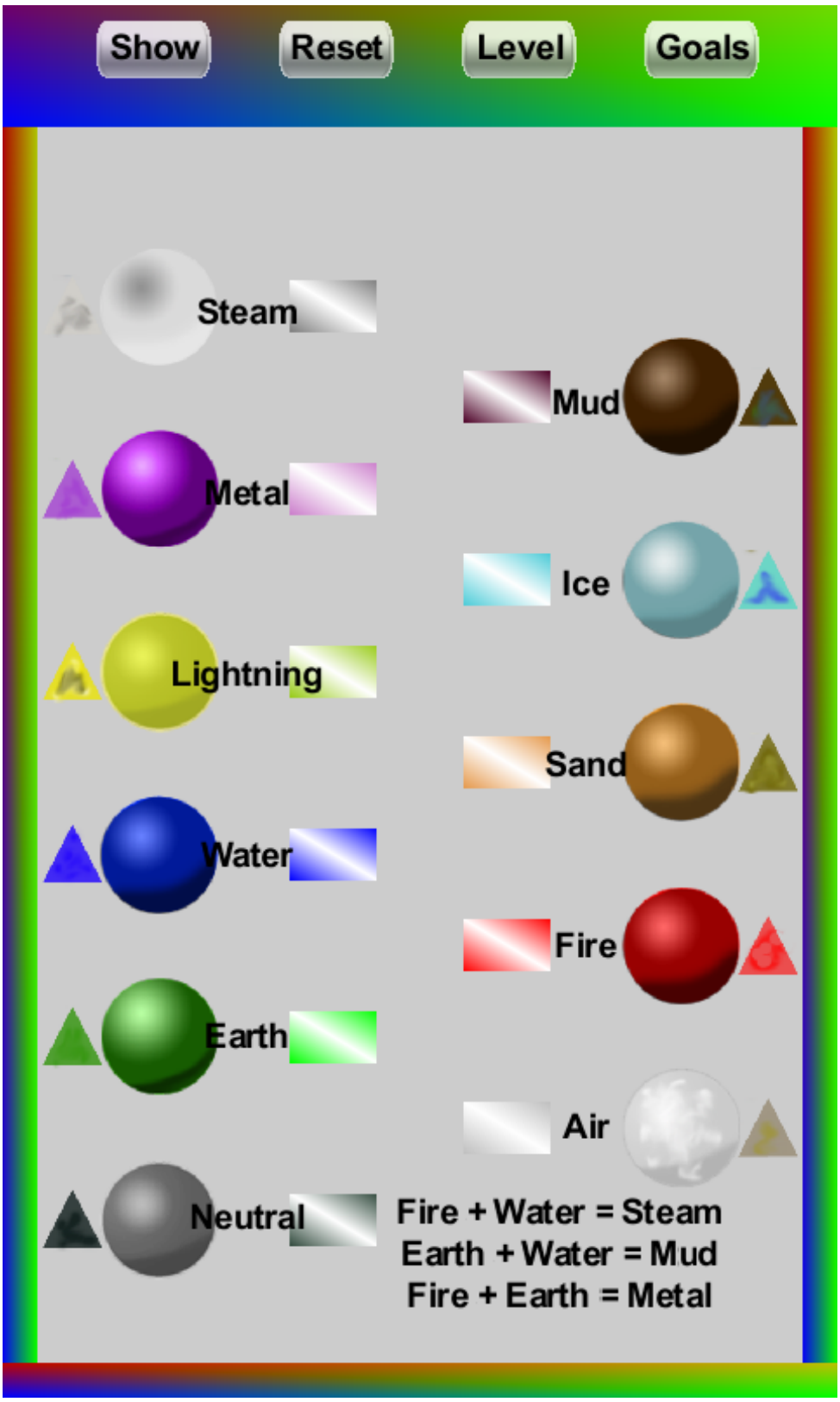


Figure 18. A Show screen was implemented to help the players recognise the different elements and some mixing combinations. It is accesibale with the “Show”-button at the top.

Test person ÅB	
Gaming experience:	Years of experience with mainly console games.
Mobile/smartphone game experience:	Experienced with handheld consoles.
Player feedback	**Our response**
There should be a Next button for the tutorial levels.	Implemented.
Some level description is having a lot of text without spaces (enter)	Could not change at the time due screen space restrictions.
Confused about the many different marbles that appear at once	Going try to limit each level to only involve 2-4 elements.
No understanding of how marbles were created.	Improved this by adding toast messages that popups when a marble is created.
Would appreciate if there was a "Tips" button.	To be implemented in the future.
Level 4, 6 and 8 is fun and should turn up sooner.	Changed the level order for the next test.

Test person COE	
Gaming experience:	Years of experience with games and also a game developer.
Mobile/smartphone game experience:	Experienced.
Player feedback	**Our response**
There is too much text for a casual player.	Will try to reduce amount of text or present the levels in smaller blocks of text.
Lots of different game mechanics at once makes it confusing.	Only introduce one game mechanic at a time and let the player get accustomed to it with some levels before introducing a new mechanic.
Levels with non- or ever-goals should alert the player about them.	To be implemented.
Change the image for good and bad holes.	To be implemented.
Remove the element aspect and only have colors and let them mix.	To be implemented.

Test person SK	
Gaming experience:	Years of experience with games and also a game developer.
Mobile/smartphone game experience:	Experienced.
Player feedback	**Our response**
Let players receive instructions through images instead of only text.	To be implemented.
Playing at the moment is mainly trial and error.	Going to try to make smarter and more fun levels.
Reading the level instructions should not take longer than playing the level.	Redesign the tutorial stages.
Change the image for good and bad holes.	To be implemented.
Remove the element aspect and only have colors and let them mix.	To be implemented.

3.7. Example code

This section's purpose is to display some example code and is not involved in the evaluation of *Libgdx*. An overview of the code is also shown with a class diagram, describing the code in a top-level manner.

3.7.1. Program overview

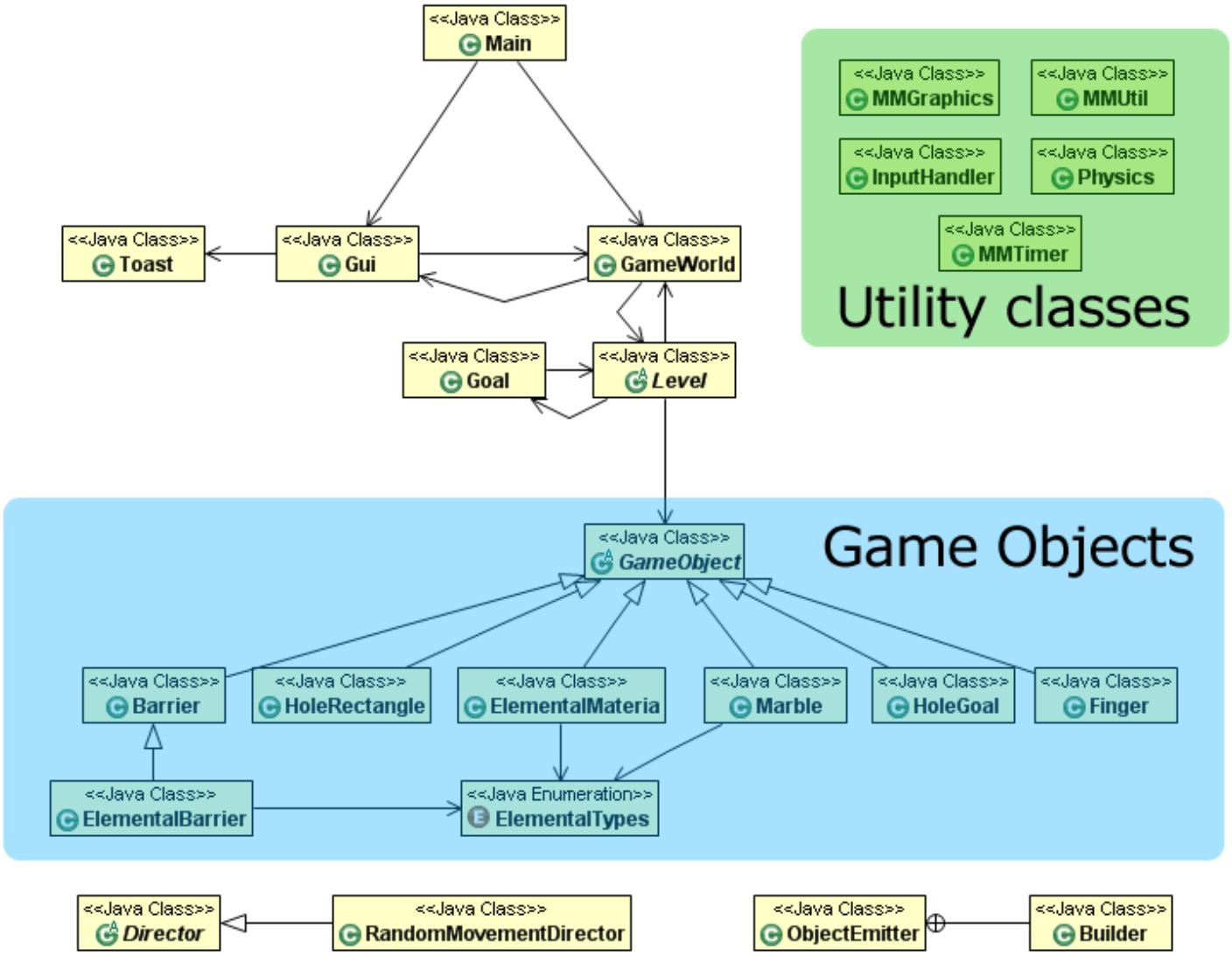


Figure 19. An overview of the most important classes and the most important relations between them.

3.7.2. Builder pattern in Goal and ObjectEmitter

ObjectEmitter uses the Builder pattern[31] to its full extent, since ObjectEmitter need to be able to build different objects, which it can do by having an inner Builder class. Different properties need to be set for the ObjectEmitter to work. Required ones are the GameWorld parameter and Vector2 parameter for location. Other parameters are optional and change the default values for when and how to emit an object. ObjectEmitter with the inner class Builder looks like:

```
public class ObjectEmitter {
    private final Vector2 location;
    private float minSpeed;
    private float maxSpeed;
    private GameObject[] emittableLookup;
    [...]

    private ObjectEmitter(Builder builder) {
        location = builder.location;
        minSpeed = builder.minSpeed;
        maxSpeed = builder.maxSpeed;
        [...]
    }

    public static class Builder {
        // Required parameters
        private final GameWorld gameWorld;
        private final Vector2 location;
        // Optional parameters
        private float minSpeed = 10;
        private float maxSpeed = 30;
        private List<GameObject> emittables = new ArrayList<GameObject>();
        [...]

        public Builder(GameWorld gameWorld, Vector2 location) {
            this.gameWorld = gameWorld;
            this.location = location;
        }

        public Builder speed(float min, float max) {
            this.minSpeed = min;
            this.maxSpeed = max;
            return this;
        }

        /**
         * This method can be called multiple times to add the different
         * GameObjects this emitter should emit. The priority value defines
         * how often this object should be created related to
         * the other objects in this emitter. [...]
         */
        public Builder addEmittable(GameObject obj, int priority) {
            [...]
            return this;
        }

        [...]

        public ObjectEmitter build() {
            [...]
            return new ObjectEmitter(this);
        }
    }
}
```

[31] Joshua Bloch (2008), *Effective Java 2nd edition.* Prentice Hall. p. 11-16.

The code is used by first creating an instance of ObjectEmitter.Builder and then calling methods on it for setting the fields you want. Then call the build() method to create the actual ObjectEmitter. The method addEmittable() determines which objects to emit.

The reason for using the Builder pattern is to have client code (the code creating the ObjectEmitter) only need to specify the parameters which they want to define without needing a large number of constructors for classes with a large number of optional parameters. Some languages such as Python has this in the language in the form of named method parameters where you can call methods like this: ObjectEmitter(minSpeed=17, maxSpeed=4711) instead of relying on the order of the parameters to determine which value goes into which variable. This also makes the code more readable, not needing to look up the method definition to see what a certain value in a method call is about.

As mentioned earlier in section 3.4, the Goal class borrows some ideas from the Builder pattern. The difference is that a goal does not include the final build step, instead the combinations of properties is checked elsewhere. Here are the different set methods that configure a goal:

```
public Goal setGoalAmountOfMarbles(int goalAmountOfMarbles) {
        this.goalAmountOfMarbles = goalAmountOfMarbles;
        return this;
    }

    public Goal setGoalElement(ElementalTypes goalElement) {
        this.goalElement = goalElement;
        return this;
    }

    public void setApplyToHole(boolean applyToHole) {
        this.applyToHole = applyToHole;
    }
```

Since the set methods also returns the Goal object itself, they can be called continuously. A goal in the Absorb Tutorial Stage, where you need to create a fire marble, can be created in this way:

```
testGoal = new Goal(gameWorld, this, "1 fire marble", "Have at least one fire
marble");

testGoal.setGoalAmountOfMarbles(1).setGoalElement(ElementalTypes.FIRE);
levelGoals.add(testGoal);
```

The goal is added to levelGoals of the Level object, which are the regular goals of a level. Other goals, such as end-, non- , and ever- goals are added in a similar way, they are added to another list of goals, that is separately treated by the Level object. For the same stage, a non-goal of not having a water marble can look like:

```
testGoal = new Goal(gameWorld, this, "1 water marble", "Do not have water
marbles");

testGoal.setGoalAmountOfMarbles(1).setGoalElement(ElementalTypes.WATER);
levelNonGoals.add(testGoal);
```

4.Discussion

4.1. Why frameworks

With the use of a framework we were not required to learn the Android way of programming as explained in 2.2.3 Development without frameworks. This could be viewed as a drawback, being dependent on a particular tool and not having knowledge of an actual development platform. Setting up a more complex game project without frameworks requires additional comprehension when only working with Android programming.

Adversely, the advantage is to avoid learning a new platform and way of designing applications. In our case, learning a framework based on a programming language (Java) was more advantageous than learning the Android way of programming, particularly considering the time constraint. For example, handling more advanced graphics is more complicated without a framework, as stated in *Pro Android Games*[32]. Classes for polygons and sprites had to be created explicitly in that case.

4.2. Motivation for choosing Libgdx

We chose Libgdx since it has desktop development support and supposedly good performance. Desktop support enables us to develop faster as we do not have to rely on mobile devices or slow emulators. Better performance will make our game run on slower budget phones and not restrict us to expensive high-end devices, thus reaching a larger audience.

Libgdx also has better low-level graphics support if we would like to use it for further projects or more advanced graphics features. The downside is that Libgdx is said to have a higher learning curve but it weighs up on that with more development opportunities.

4.3. Libgdx evaluation

Much of the information we gathered about Libgdx, in our quick study regarding different frameworks, was accurate. Libgdx gives a fast way of developing since it has desktop support, which is unique regarding open source Android game development frameworks. This support enabled us to quickly get started with an application, without having to worry about Android-specific issues.

4.3.1. Advantages

We were able to quickly implement the fundamental concepts of the game, Version 0.0 – Pre-Game, and test our game concept. Doing so, with purely Android programming, would take us much longer since we would have to learn at least Android development and maybe coupled with OpenGL knowledge too. Because we used Libgdx, our previous Java knowledge sufficed, learning some minor new subjects on the go when needed.

The test-debug cycle is also shorter since launching the game in the emulator takes several times longer than launching it as a desktop application. This matters when making many small changes and testing them by running the program such as commonly done when trying out a new platform.

4.3.2. Disadvantages

Since Android and Libgdx uses OpenGL ES for its graphics implementation, we had to learn to implement graphics in a new way as well and not rely on our previous knowledge. Libgdx did not have all necessary documents in one place regarding different types of graphics, except for a few basic tutorials. Guides or documentation regarding menus, GUI, et cetera would be an appreciated addition to their guide library.

[32] Vladimir Silva (2009). *Pro Android Games*. Apress, page 81.

Even though Libgdx provided us with a fast way of starting the development, we quickly realized that more Android knowledge was needed. Our early test on an Android device (HTC Desire) showed that there was something wrong with the Android implementation, since it swiftly drained battery power.

4.3.3. Neutral assessments

Libgdx is a framework that enables to quickly get started with development, after the initial setup. Although Libgdx supports desktop development, do not be naive by thinking for example only a little Android programming or other needed knowledge is required. Take into consideration the learning of displaying graphics with OpenGL and its complications.

Libgdx should not be the first choice when doing a simple 2D game. For more complex projects, where the developers have time to learn or already possess the various knowledge needed, Libgdx can be the best tool thanks to its desktop support, good performance and other advanced features.

For the time constraint of this report, the complexity of Libgdx was not suitable, but for future development regarding *Magnificent Marbles,* the advanced features are attractive and need further investigation.

4.3.4. Criteria and purpose fulfillment

Libgdx fulfills nearly all of our criteria well or very well. See section 2.2.4 Frameworks for a list of our criteria. In particular it fulfilled the following criteria:

a. It is free of charge to use and the source code is available.
b. It is actively maintained with some very active developers as well as a rather active community.
c. The Javadoc documentation covers most of the essential classes and methods. There are also a number of small test programs and larger game demos available, including source code.
d. The performance of Libgdx is good but it does require some work and knowledge from the developers using it. It supports the functionality we want including a 2D physics engine as well as support for GUI components.

Our purpose in finding a framework for developing a mobile game for the Android platform (as specified in section 1.2 Purpose) was certainly met. Libgdx is a framework which works for developing mobile games in a familiar way. The drawback with Libgdx is that it is not tailor made for developing simple 2D games. Nonetheless, the advantage of Libgdx lies in the abundance of advanced features that enable more complex games. Learning Libgdx now for this project might have taken a little longer than using a simpler framework but that knowledge is likely to save time in the future if more advanced projects are attempted.

4.4. Android issues

Android and other feature-rich smartphone platforms are relatively new and do not have a shared standard on how to develop; each has its own standard. Android has its life cycle model, iPhone has a different one. Disregarding this new way of thinking about applications, activities and lifecycles can be detrimental, even if Libgdx has desktop development support. It can be a nice lure to have the desktop support that Libgdx provides, but that does not get rid of the complications regarding Android life cycles.

Android programming knowledge is a recommendation no matter what framework or tool that is used or what previous programming language that was mastered earlier. A new platform should be viewed independently and studied well, instead of relying on old experience.

4.5. Game changes and features

This section concerns the changes and features planned for the game and are not involved in the evaluation of *Libgdx*.

4.5.1. Level descriptions

The improvements and changes regarding the game design have already been addressed in section 3.6 with the help of user testing. The main issue of the game at its current stage is that it is too much of an effort needed for a casual player to fully understand all the game mechanics at once. The level descriptions, especially for the tutorial levels, are too lengthy for a player with short attention span to bother with. Images accompanied with one or two lines of text should be enough to describe a function in the game.

4.5.2. Tutorial overhaul

With regards to this, the tutorial stages need a major overhaul so the different game mechanics gets easier to understand. For example instead of having all the tutorial stages after one and another, there could be 3-4 levels with only the previously introduced mechanics, between each tutorial stage, and the introduction of a new game mechanic. This would give the player time to understand each game mechanic.

4.5.3. Memorizing

Another issue was that at times there was too much a player needed to remember, including doing different tasks in a certain way. The players needed to read the instructions several times to understand or memorize them. One way to solve this is to make levels more interactive, so you do not get all the goals at once; instead they get to be introduced one after another. This is mainly a concern related to level design and would be solved by investing more time to create an interactive level that is playable for a longer time.

4.5.4. Elements and colors

An issue that was more or less solved during the user testing was the need to memorize different elemental combinations. The solution was to have a help screen that showed different combinations. In the future instead of having elements, marbles would simply be categorized by colors, as remembering different color mixes is more easier and natural. See Figure 20 for reference.

The idea behind having different elements was that there were already games where the entire purpose is to mix different elements, for example Doodle God or Alchemist for Android. The difference being that in these games, you did not have to remember the combinations and could just mix and match freely. Since our game never got to the stage to implement different elemental properties as mentioned in the Concept Document (see Appendix), simple colors would be better.

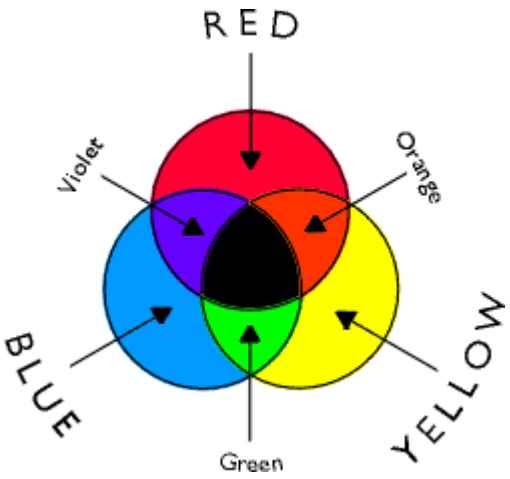


Figure 20 This shows how colors can be mixed and in which way.

4.5.5. Miscellaneous

Another feature to implement is audio, by having sound effects and some background music. Graphics and the GUI could also be updated to give a cleaner look, instead of having Barriers at the edges of the screen. Using TWL is viable option for having a better GUI. An options menu and a level select menu would also be good to have.

4.6. Future improvements

The area in which Libgdx, according to us, could be improved is mainly more documentation for people unfamiliar with OpenGL and Android programming. We think that Libgdx will continue to develop as an open source project, as it has in the past year, and these improvements will probably be implemented.

The improvements we could make for ourselves are continuing the exploration of the Android platform and learn it thoroughly before venturing into more complex projects. Learning basic OpenGL would be useful for future games as well.

4.7. Conclusion

The conclusion we can draw from this short evaluation of Libgdx is that Libgdx could be a great tool if used by the right people. By that we mean developers with knowledge of the Android platform, OpenGL and other related areas. The desktop support provided by Libgdx is highly attractive, especially for developers inexperienced with Android programming. Libgdx could be used to a great extent to create prototypes of games quickly and then easily run them on Android devices.

Using Libgdx in the future for other game projects are highly likely now that we know what is required to develop a new game. *Magnificent Marbles* will continue its development with Libgdx at the same time as we learn more about the required areas mentioned earlier, although this lies outside the scope of this evaluation.

5. References

(In order of appearance)

1. Smartphones Provide Extra Mana for Mobile Games Industry […]
   <http://www.comscore.com/Press_Events/Press_Releases/2009/1/Mobile_Gaming_Grows>
   [Retrieved 2011-04-08] [Published 2009-01-30]

2. iPhone "most successful" mobile games OS.
   <http://www.gamesindustry.biz/articles/iphone-most-successful-mobile-games-os>
   [Retrieved 2011-04-08] [Published 2009-03-12]

3. HTC's Teeter shows off how physical feedback can immerse you in a game.
   <http://www.zdnet.com/blog/mobile-gadgeteer/htcs-teeter-shows-off-how-physical-feedback-can-immerse-you-in-a-game/1168>
   [Retrieved 2011-02-20]

4. Youtube video image of Teeter.
   <http://img.youtube.com/vi/l2o_YKlS6Ns/0.jpg>
   [Retrieved 2011-02-20]

5. Air Control Game Review by AndroidTapp.com.
   <http://www.androidtapp.com/air-control/>
   [Retrieved 2011-02-20]

6. Flight Control.
   <http://firemint.com/flight-control-choose-your-landing-platform/>
   [Retrieved 2011-02-20]

7. Air-Control-in-game_Play-6.jpg
   <http://www.androidtapp.com/wp-content/uploads/2010/04/Air-Control-in-Game-Play-6.jpg>
   [Retrieved 2011-02-19]

8. What is Android? - Android Developers.
   <http://developer.android.com/guide/basics/what-is-android.html>
   [Retrieved 2011-01-29]

9. Android Developers, 2011, Platform Versions.
   <http://developer.android.com/resources/dashboard/platform-versions.html>
   [Retrieved 2011-02-13]

10. Installing the SDK - Android Developers.
    <http://developer.android.com/sdk/installing.html>
    [Retrieved 2011-02-13].

11. Hello, World - Android Developers.
    <http://developer.android.com/resources/tutorials/hello-world.html>
    [Retrieved 2011-02-13].

12. Notepad Tutorial - Android Developers.
<http://developer.android.com/resources/tutorials/notepad/index.html>
[Retrieved 2011-02-13].

13. Vladimir Silva (2009). *Pro Android Games.* Apress.

14. Edward Burnette. (2010). *Hello Android*. Pragmatic Bookshelf.

15. Guiran Chang, Chunguang Tan, Guanhua Li, and Chuan Zhu. (2007)
Developing Mobile Applications on Android. Springerlink. LNCS 5960, pp. 264–286.

16. Top Ten Open Source Android (2D or 3D) Game Engine […]
<http://www.cuteandroid.com/ten-open-source-android-2d-or-3d-game-engine-for-android-developers>
[Retrieved 2011-02-12] [Published 2010-10-05]

17. OpenGL ES.
<http://www.khronos.org/opengles/>
[Retrieved 2011-02-23]

18. Box2D – Home.
<http://www.box2d.org>
[Retrieved 2011-03-01]

19. [android-developers] Libgdx vs. andengine.
<http://www.mail-archive.com/android-developers@googlegroups.com/msg133167.html>
[Retrieved 2011-02-19]

20. Libgdx, Andengine & Rokon Micro-Benchmarks.
<http://www.badlogicgames.com/wordpress/?p=803>
[Retrieved 2010-02-15]

21. Badlogic Game – View forum - Showcase
<http://www.badlogicgames.com/forum/viewforum.php?f=16>
[Retrieved 2010-04-08]

22. HTC – Products – HTC Desire – Overview.
<http://www.htc.com/www/product/desire/overview.html>
[Retrieved 2011-01-28]

23. HelloWorld
<http://code.google.com/p/libgdx/wiki/HelloWorld>
[Retrieved 2011-02-14]

24. MyFirstTriangle
<http://code.google.com/p/libgdx/wiki/MyFirstTriangle>
[Retrieved 2011-02-14]

25. Issue 151 - libgdx - Pixmap.FillCircle dont work
<http://code.google.com/p/libgdx/issues/detail?id=151&can=1&q=drawcircle>
[Retrieved 2011-02-21]

26. Ogmo Editor.
<http://www.ogmoeditor.com/>
[Retrieved 2011-03-31]

27. Builder pattern – Wikipedia […]
< http://en.wikipedia.org/wiki/Builder_pattern >
[Published 2011-05-26] [Retrieved 2011-06-14]

28. TWL - Themable Widget Library
< http://twl.l33tlabs.org/>
[Retrieved 2011-06-14]

29. Linking assets between desktop and android project in Libgdx […]
<http://www.badlogicgames.com/wordpress/?p=1537>
[Retrieved 2011-03-31]

30. Usability testing – Wikipedia […]
<http://en.wikipedia.org/wiki/Usability_testing#Hallway_testing >
[Published 2011-05-29] [Retrieved 2011-06-14]

31. Joshua Bloch (2008), *Effective Java* 2nd *edition.* Prentice Hall, pages 11-16.

32. Vladimir Silva (2009). *Pro Android Games*. Apress, page 81.

6.Appendix

6.1. Magnificent Marbles – Concept Document

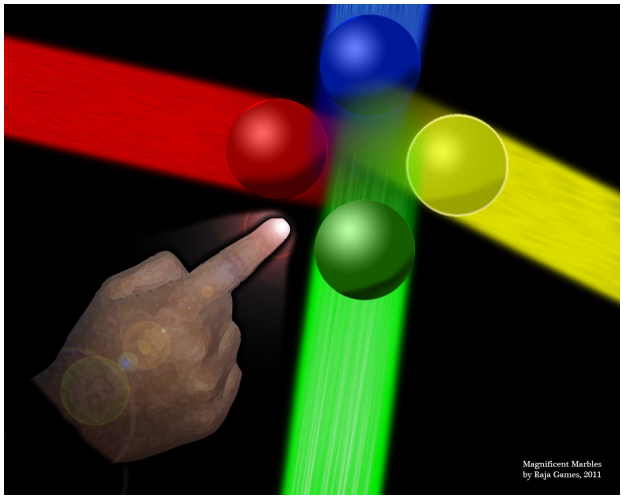
The purpose of this document is to be used as a pitch document, to sell or present the game idea behind *Magnificent Marbles.* It was committed to the SGA in order to receive feedback. The document is only a concept and does not state what the end version will be. It is mainly used for getting inspiration for the developers and also interest potential stakeholders.

Magnificent Marbles

by Raja Games

2011-02-28

Tagline:	Touch and Roll
Genre:	Puzzle
Platform:	Smartphone, Android (2.1+)
Target group:	Casual gamers; plays games for short moments while on the train or bus on their way to school or work. Might have played with marbles as a kid.
Goal of the game:	Touch, roll and create magnificent marbles. Use the marbles you have created to solve puzzles.
Requirements:	Android smartphone, mid-range.



Introduction

Magnificent Marbles intends to bring back the nostalgic marble playing experience you had when you were a child, with a magnificent twist! This twist consists of using the modern smartphone's touch functionality to emulate some hands-on play you had as a kid.

The main concept behind Magnificent Marbles is to have different marbles and move them with a flick of your finger, just like in the playground of young times. The view will be a top-down bird's view with 2D graphics. The magnificent part is that the marbles have different properties that are used for special purposes. There can exist fire marbles that are able to burn through wood or water marbles that could extinguish fires. A marble can interact with other marbles in different ways, such as simply pushing them away and absorbing, merging or multiplying with them to create other marbles (*Marble Mixing*).

Tutorial level

To not get overwhelmed with all the features of the game, the player will start with a simple and short tutorial that will give an introduction to all the basic aspects of the game. In the beginning of the tutorial you start with one neutral marble and your first goal would be to absorb elemental materia to be able to evolve to a special marble. For instance collect enough fire materia for evolving to a fire marble and burn through the wooden walls to get access to the next stage.

The second stage would involve other coloured marbles and your goal would be to push them to their coloured holes. The third stage would introduce the player to the merge mode and the *marble mixing* aspect of the game. For instance if you have a fire marble and you let that marble merge with a water marble, you would get a steam marble that would be able to float to, until now, an unreachable part of the level.

For the last two stages things will be a bit more complicated. You will need to deal with multiple marbles and AI controlled marbles. For the fourth stage you will need to multiply your marble by absorbing materia or merging with other marbles. For instance you might need several marbles to perform specific tasks in a timely fashion. Put the first marble in hole A, to open up hole B for two seconds for your other marbles.

For the last and sixth stage of the tutorial, your marbles will meet up with other moving marbles. The AI marbles can move in different ways, such as circling a special area or hole or simply moving freely. It will be up to the player to figure out what needs to be done and how.

Target audience, game goals and beyond

As a casual puzzle game, the intended target group for this game is people who want to spend some minutes on a game or level while they are in between other activities. Drop-in play that can happen anytime or anywhere. There is also room for a more competitive approach if a player wants to keep beating high-scores in a particular mode or level.

The goal of the game's tutorial is to get familiar with the gameplay that comes with moving magnificent marbles around. Since it is a casual game there is only an underlying short term goal. That is to have the freedom to be able to experiment with your marbles and see what mixing them or interacting with them in different ways will lead to.

Currently the main focus of the project is to make a well-made tutorial level. The ways forward for the game would involve other game modes after the tutorial is done. One example would be to have an infinite arcade mode, where the level content would be randomly generated and the player would have to survive with his/her marbles as long as he/she can. Another classical option is to have different levels with specific layouts and missions.

Appendix

Summary

Different stages of the tutorial for introducing the player to the different modes and aspects of the game:

1. Absorb - Collect elemental material to evolve to a special marble and use its ability.
2. Push - Push other coloured marbles into their respective holes.
3. Merge - Collide with other marbles to evolve into another special marble.
4. Multiply - You need X amount of marbles to complete a certain task.
5. AI - Time to interact and fight off the AI marbles.

After this tutorial level the game could have other modes such as:

- Infinite: An arcade mode where the level layout, materia and AI marbles would be randomly generated. Score is kept to have a high score list.
- Level: Levels with special goals, marble types and terrains.
- Campaign: Heard the story of one brave marble that magnificently saved the world?
- Duel: Multiplayer 1 on 1 real time action. Decide who the best marbler is!

Concept art