Good 3D Mobile Gaming Practice

A product development project

Tony Ross Wainio

Bachelor's Degree Thesis
Film & TV / Online Media & Art Direction
2014

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Film & TV
Identifikationsnummer:	5311
Författare:	Tony Ross Wainio
Arbetets namn:	Good 3D Mobile Gaming Practice
Handledare (Arcada):	Owen Kelly
Uppdragsgivare:	-

Sammandrag:
Detta produktutvecklinsarbetet inom mobila applikationer ska ge en förståelse om vad man kan och vad man bör göra och tänka på medan man utvecklar 3D spel för mobila platformer. Temat för spelet är självreflektion och genomtänkande av dagens händelser. Problemen är att det inte finns klara riktlinjer för vad man borde ta i beaktande I utvecklingsprocessen. Metoderna som använts i arbetet har varit empirisk forskning och testande av tekniken samt forskning av tekniska begränsningar och hur de kan kringås. Resultaten var ett alpha-stadie av spelet som kommer att publiseras senare, samt en mäng riktlinjer och viktiga punkter som är till nytta I utvecklingsprocessen.

Nyckelord:	Spelutveckling, 3D, mobilapplikation, flerplatform, pekskärm, flerplatformskod, användbarhet, användargränssnitt
Sidantal:	23
Språk:	Engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Film & TV
Identification number:	5311
Author:	Tony Ross Wainio
Title:	Good 3D Mobile Gaming Practice
Supervisor (Arcada):	Owen Kelly
Commissioned by:	-

Abstract:
This product development in mobile applications should provide an understanding of what you can and what you should do and consider while developing 3D games for mobile platforms. The theme of the game is self-reflection and reflecting on actions taken during the day. The problem is that there are no clear guidelines for what you should take into account during the development process. The methods used in this work have been empirical research and testing of the technology and research to technical limitations and how they can be circumvented. The results were an alpha stage of the game that will be published later, and a series of guidelines and key points that will help in the development process.

Keywords:	Game development, 3D, Mobile application, multiplatform, touchscreen, multiplatform coding, usability, user interface
Number of pages:	23
Language:	English
Date of acceptance:	

OPINNÄYTE	
Arcada	
Koulutusohjelma:	Film & TV
Tunnistenumero:	5311
Tekijä:	Tony Ross Wainio
Työn nimi:	Good 3D Mobile Gaming Practise
Työn ohjaaja (Arcada):	Owen Kelly
Toimeksiantaja:	

Tiivistelmä:
Tämä tuotekehitystyö mobiiliaplikaatiota varten antaa ymmärrystä siitä mitä voi ja mitä pitäisi tehdä ja miettiä 3D-pelejä mobiilialustoille kehittäessä. Pelin teemana on itsensä ja omien tekemisiensä tutkiminen ja ajatusten puhtaaksiajatteliminen rankan työ- tai koulupäivän jälkeen. Ongelmana on, että ei ole olemassa selkeitä ohjeita siitä, mitä on otettava huomioon kehitystyön aikana. Tässä työssä käytetyt metodit ovat olleet empiiristä tutkimusta, testausta sekä teknisten rajoitteiden löytämistä ja kokeilemista. Lopputulos on alpha-vaiheessa oleva peli, joka julkaistaan myöhempänä ajankohtana, sekä lista erinäisiä ajatuksia ja vinkkejä siitä mitä kannattaa pitää mielessä 3D peliä tuotettaessa mobiileille alustoille.

Avainsanat:	Pelinkehtiys, 3D, Mobiiliapplikaatio, monialusta, kosketusnäyttö, monialustakoodaus, käytettävyys, käyttäjäympäristö
Sivumäärä:	23
Kieli:	Englanti
Hyväksymispäivämäärä:	

INNEHÅLL / CONTENTS

FOREWORD

Developing good praxis to tackle a demanding gaming project, in 3D in particular, is something every aspiring game developer should want to strive for. Nowhere is this a more clear a goal, than in mobile gaming.

Mobile gaming has become a very large and respectable market now, all starting with the apps one could download on their iPhone Edge back in 2007. This new technology of capacitive touch-screens and very good overall performance in terms of hardware, coupled with seamless integration with the native OS and easy digital distribution really sparked the industry into working more closely with hardware manufacturers to be able to squeeze as many frames per second as they possibly could.

When Google's mobile OS, the Android OS came into the game in 2009, it became apparent that mobile was the way forward. No other mobile OS has had this big a market-share since Symbian 40 and Symbian 60 from Nokia back in the early 2000's. And, Google Android OS being free, and open source, meant that a whole new breed of game-developers have stepped onto the scene, forming companies that we couldn't have thought to exist some 10 years ago. Companies like Rovio, Supercell, Grand Cru Games, just to name a few Finnish game developer companies weren't in action 10 years ago. Now though, they're running the world it seems.

In this thesis, we're exploring the possibilities and limitations of one of the most widely used 3D mobile gaming engines = Unity3D. The goal is to compile a small compendium of information for anyone to keep in mind whilst generating 3D content on mobile platforms such as tablets and phones.

The project in question is a relaxing toy or curio for the working man or woman who just wants to spend a few minutes a day relaxing and self reflecting, and doing so by the means of gaming. The reason a mobile platform was chosen for this project, was because mobile phones are more widespread and constantly in use than personal computers.

1 PRELIMINARY RESEARCH

To be able to determine the needs of 3D game development, we need to look at a few different platforms and compare them. Some practical testing is also needed, in order to see why one engine might be subjectively better than the other.

The needs of this project are neatly summarized in three points: Easy to navigate GUI, at least one well documented scripting language capability and mobile readiness.

1.1 Adobe Flash with Away3D package and Citrus Engine

Flash was originally the first choice, because of courses previously attended, as well as because of the beforehand knowledge of its basic and some advanced features and possibilities. The Flash scripting language is called Actionscript, and as it stands, it's one of the most well documented languages in coding and gaming.
Whilst it might seem as the obvious choice, almost solely based on its well documented existence, it's got some drawbacks. One of the major ones is its sheer abundance of different engines (be it physics, graphics, or a mix thereof).

For the purposes of this project, the Citrus Engine was chosen for comparison, mainly because of its ease-of-use, ability to import some 3D models and formats, and portability. Also, it would take up a sizeable portion of this thesis to go into detail and explain all the possible Flash engines to be used, simply because, as stated before, there are so many of them.

1.1.1 Citrus Engine pros & cons

The Citrus Engine has a whole lot going for it, including but not limited to:

- A nice and well organized compendium
- A myriad of tutorials from basic to advanced features
- An active discussion forum with an active community

Citrus also has a few drawbacks, mainly:

- Support for multiple formats of 3D models
- Very resource intensive
- Not supported by all mobile OS-platforms
- Dependent on other plugins for 3D rendering

Verdict: Citrus, with Away3D, is quite powerful for desktop development as well as subtle and small projects, however it does not fit the needs of our project.

1.2 Native SDK development

The notion of using every mobile platforms own Native SDK was a tempting thought. It's the most versatile way of development per platform, it has the potential of being the least resource intensive and easiest to debug, and they have a very extensive support system. After careful consideration of all pros and cons for the SDK's, it was clear that this was not the avenue to pursue.

1.2.1 Native SDK pros and cons

The advantages of going native are clear:

- Full support for that system
- Incredibly clear and strict coding and scripting policies
- Possibility of being frugal on hardware resources
- Very easy to troubleshoot and debug

The disadvantages regarding development for this project are also very clear:

- Only one system can be chosen
- Very little tutorials, and the ones that do exist, are very intense
- Giving away partial software licenses to the platforms developers is not a very interesting or intriguing option
- Sheer amount of foundation to be built before actual development can start
- Not a very user-friendly GUI, in fact for the most part there is hardly any GUI to even delve into and cannot be considered very newbie-friendly.

Verdict: Native SDK development is probably the most rewarding in terms of the “when it's done”-principle, but the way to the end is rocky and perilous. This project is in need of something easier, more lightweight in terms of learning and getting to grips with the engine to be used, as well as multi-platform development. The Native SDK is a good choice for the bigger companies with millions to spend, but it is not the right choice for this project.

1.3 Unity3D – Open source, multi-platform development

When Unity3D first came to the market in 2005, it was meant for desktop and browser gaming. A free, open source game development utility run and built by the community, hence the name Unity. After Apple Computers Ltd. launched their iPhone Edge in 2007, there was a massive need for games and of course Adobe Flash was the first to the scene for independent developers. No one realized this would be the start of a mobile gaming revolution, except for Unity3D.

Unity Ltd. adapted quickly to encompass the whole mobile market (smartphones and touch-devices) with its tight knit community with avid developers and coders. The best part of Unity3D was being free, which gave independent developers a chance to show their prowess.

Touch-devices, smartphones and tablets have all developed so fast on the hardware side, that software developers have had a rough time keeping pace. So far, it seems that Unity3D might be the best development platform, according to preliminary research, for this project.

1.3.1 Unity3D Pros and Cons

Some of the advantages of Unity3D are pretty straightforward and clear:

- Free
- Fast and intuitive GUI
- 3D live preview
- Large community and decent documentation of functions
- Widely popular with indie-developers, which makes for quite a large library of functions and development questions and problems answered with a little bit of google-fu

The disadvantages of Unity3D are quite standard with all open platforms:

- Not as flexible as native SDK development
- Has limited functionality in some cases
- Requires you to learn one of two languages: C# or JavaScript

Verdict: Unity3D is probably one of the easiest ways to get started fast, is somewhat flexible and is free, and as you learn your desired language (C# or JavaScript) you are developing to iOS, Android and WP at the same time. And better yet, the same game can be exported without major modifications to all platforms, as well as Desktop and Browser environments. Definitely the best choice for this project.

2 PROJECT DESCRIPTION

In this section we're covering the meat of the potato as it were. The focus will be on the visual representation and graphics, as well as the coding of the game and how the project has developed during a few months time.

2.1 Project conception

The project came to be through a series of events and thought experiments as well as the realization that our mobile devices are constantly on our person and accessible in mere seconds.

The first idea was to develop a portable USB world, installed and run on a USB stick inserted in any windows computer anywhere at any time. It was supposed to run on Open-Sim which is a free and open version of Linden Labs open-world mmo-game Second Life. The thought of having a malleable game world where one could freely experiment and build whatever you wanted, was an intriguing one. After several experiments, numerous version revisions and a couple hundred hours of 3D modelling and testing, it was decided that this USB-stick malarkey was clunky and didn't fulfill the needs of the psychological self-reflection needed at the correct time.

The idea then developed into a mobile platform, because of the way the psychological benefit was perceived; that is to say during downtime. One does not want to wait for half an hour or more to get home to plug the USB-stick into a computer, start it up, load the world and then start exploring. Why not use the time on the bus, tram or any appropriate means of travel where you already have some downtime after work or a hard day at school to relax and reflect on your day while it's still fresh in memory?

This is the exact reason the mobile platform was chosen. Be it a phone, tablet or something else, you can use the time you would normally sit and wait, to reflect on your day while exploring a world full of semi-randomly generated scenery and items, that all react

to how you interact with them. Some items would increase your speed, some decrease it, and some might even change the colors of the world or make a sound.

2.2 Project development

Discussions in our thesis project group, together with our supervisor Owen Kelly, and thesis project designer Idamaija Pitkonen-Piguet, we decided that we need a set of parameters. These parameters need to be fulfilled in order to get the project done and for the game to be in working order as well as look and feel like a working product.

2.2.1 Project parameters

The instructions from the game concept design, or recipe as it were, are as follows:

- A game for mobile devices
- The game world is in 3D
- No instructions delivered to the end-user, just 3 sliders with which to control the look and feel of this particular gaming session. These sliders are based on colors that reflect on emotions. i.e. red for anger, green for happiness, blue for sadness. There is one additional, a 4^{th} slider, which decides the length of this particular gaming session.
- The game starts with the player flying in this colorful space and is in constant forward motion. Steering is done by leaning/tilting the device.
- The world is a loop, or a cylinder form, so that if the player flies constantly to the left, they'll arrive past the same point in space eventually.
- The game world has abstract objects that the player will interact with to change and alter the game world. These objects are randomly generated and placed around the gaming world.
- There will be a question at the end of the game, after you've achieved the goal (which is to come to arrest, slowing down enough to find your center and relax.)

The decision was made to start developing the game part first, and deal with the sliders and world altering objects later.

2.2.2 Game world coding and modeling

Since we had chosen Unity3D as our development platform, we had a choice of two coding languages; JavaScript or C#.

JavaScript in Unity3D isn't actually JavaScript.

> The very first question all beginning Unity developers have is C# or JS. Always go with C#.
>
> Now, right now you may be thinking, "But I don't want to learn C# and I already know a bit of JavaScript."
>
> But the misleading thing you need to consider is that **Unity JavaScript is not JavaScript.** It is a superficial veneer of ECMA syntax over .NET CLR. It does not do any of the cool things JavaScript does like anonymous enclosures and function references the way you would use them normally. It has it's own arrays that dont match up with normal .net arrays, and you'll encounter many pitfalls and limitations that you could've just avoided by using C#. Unity JavaScript is only a candy-coated layer of syntax designed to make .NET appear less intimidating, nothing more. It will do you more harm than good.
>
> (greenland, 2014)

And there are some very convincing reasons why C# would be the better choice for this particular project and for me personally as a coder. Main reason being: if you learn C# you have a less steep learning curve for the other C based languages (C and C++ among others) which are very widely used in software development.

> Unityscript is Unity's implementation of a javascript-like syntax. C# is simply C#. It's also a language that's used in a few other frameworks, so if you can take the time to learn the ins and outs of C#, it'll be more applicable outside Unity, and it's also faster. (zapdot, 2013)
>
> C# is a beautiful language, and if you ever have an inkling to further pursue programming any C derivative (C, C++, Java, etc.) will look familiar. Javascript, or I suppose unityscript, is not going to be any easier to learn and C# has a log of nice features that you can master the more experience you get with it. (Aberrations, 2013)

Quite clearly, the best choice here, is C#. This is the language to be used for this project.

Experimenting in Unity3D

Experimenting in Unity started, with good trials first on an infinite looping flat plain, and after that, the surface of a globe. Because we needed a cylinder format, it became a little trickier since there was a need for the cylinder to have coordinates and negative coordinates are not mathematically easy. Basic functionality of steering, flying, stopping and speeding up was relatively painless, and the first big problems with controlling the game came in the form of tweaking and looping.

When the player steers up, and finally passes the 90 degree angle at the top of his loop, the basic premise is that the player loops around and continues on this vector. For some reason, still unclear to us, the coordinate-system works based on gravity, and is not absolute, or tied to character heading. This means that when the movement vector which should loop around a circle and back down the other side, turns 180 degrees to face the other way, and forces the player to continue upwards and again hit the top of the circle, and the whole scenario repeats infinitely.

This means, that the player can't turn upside-down, even if he'd want to. General practice for quick game developing is to use as many ready-made pieces, add-ons or plugins as possible to facilitate fast developing. This is what we did, used the built-in Unity physics engine since building our own would take a considerable amount of time. This physics engine has constraints that were unaccounted for, and resulted in some problems with orientation. We found no way to fix this.

For testing purposes, the decision was made to not be able to steer more than 60 degrees negative or positive pitch, and a full circle of banking left or right in a loop. Since this made testing easier, we started experimenting with movement speed, and altering it with the help of fly-through or touchable items or objects floating around in the world.

Item generation and object pooling was relatively simple, and started working as intended quite quickly, but certainly needs more tweaking before a final version can be released. The idea was to have the items come out of fog-of-war gradually, and not appear by suddenly popping into view.

This fog-of-war was achieved by using the Unity built-in fog and smoke effects, as well as view-distance calculations automatically made in the unity-engine. Of course, this needed some tweaking, but worked very well in the end.
At this point the project had taken around 300 hours of research and development, and for some time the project was at a stand-still. In a sense it still is. In fact, development can't resume until we've had an initial round of testing with a test-group, and for that testing to occur, the game will have to become a bit more finished and polished and maybe even have a version 0.1 alpha-release.

Work on the project will continue, and I will be posting new and updated versions online. Once it's finished, you will be able to access the game at this address:
https://www.dropbox.com/sh/ng4juzjeuwp421p/AAC5z1BREf0T6oUpfl9xOluZa?dl=0

3 PROBLEMS AND SOLUTIONS

3.1 Problem 1 – Upside down flying and steering

The problem here was that the Unity-engine has some limitations on what gravity is and how it works an effects objects and players. The easy solution would be to disable gravity on the player characters, which is what we did for the objects in the world, but this creates another problem: all of the friction and air-resistance coefficients have to be re-measured and calibrated. This will take time and didn't seem to be worth the effort at the time.

The solution of course is to fix it in the future, and do all the recalibrations.

3.2 Problem 2 – Replay value

Replay value is a very important fact to consider when developing a game. If the game is built as a one-off then by all means you can forget about replay value. This game however was meant to be played several times a week, and thus needs to have a good deal of replay value. How this is going to be achieved is the hard part.

The suggested solution is already present: the questionnaire at the end of each gaming session, where you answer a question about your day/feelings/mood and this gets reflected back to you on your next gaming session. I would suggest finding something more, a bit more flavor to get the replay value up.

3.3 Problem 3 – Development team

The problem was not the team, but the lack thereof. The team had one designer, one programmer and one supervisor. Ideally there should be a couple, maybe three more people in the team to facilitate a faster development time, as well as a greater support-net for

when there's lack of time or effort, as well as a few extra pair of eyes for troubleshooting and error-reporting. Also, the discussion and communication that is needed for good developing and great general practice in the game-developing community, is essential.

Suggested solution: get some more people interested and recruited.

3.4 Problem 4 – How do we optimize for a wide array of hardware

This is a problem that only multi-platform developers face. iOS developers can relax, since they only have a handful of hardware to optimize games and applications for.
Android developers have to do extensive testing to be sure that the game or application runs smoothly on the low-end as well as the high-end systems. There are literally hundreds of different models of Android hardware-setups and to find a good balance, is hard if not in some cases even impossible.

Solution: either start developing for a single platform, or just make the decision of shutting out a piece of the market by having hardware or software restrictions for installation of your application. This will of course alienate those users.

4 CONCLUSIONS

4.1 Guidelines for 3D Mobile Game Development

4.1.1 Planning

Nothing is more important than planning. You need to really focus on planning most aspects, and have a clear game-plan for how you're going to tackle possible upcoming problems and think of solutions in advance.

4.1.2 Rethink and adapt plans

And now for something obvious: Your plan most certainly will not go according to how you planned it originally. This is completely normal, and you only need to think of it as a step in the progress.

4.1.3 Choose a platform that suits your particular project

The right tool for the job. This will help you make the most of your time in the whole development phase. You can also plan ahead and pick a development platform that you think could help you in your future project.

4.1.4 3D Models do not need to be extremely detailed

Mobile gaming is mostly done on small handheld screens with a resolution rarely over QHD or even HD, so those tens of hours used on one model probably won't pay off unless you are thinking of a desktop release up the road.

5 REFERENCE

Aberrations (2013) *Starting out in Unity: C# or Javascript?* [Online] Available from: http://www.reddit.com/r/Unity3D/comments/193m4p/starting_out_in_unity_c_or_javascript/ [Accessed: 28th October 2014]

greenland (2014) *Differences between C# and Javascript for Unity* [Online] Available from: http://gamedev.stackexchange.com/a/79773 [Accessed: 28th October 2014]

zapdot (2014) *Starting out in Unity: C# or Javascript?* [Online] Available from: http://www.reddit.com/r/Unity3D/comments/193m4p/starting_out_in_unity_c_or_javascript/ [Accessed: 28th October 2014]

6 APPENDICES

6.1 Summary of the main points in Swedish – Sammanfattning av huvudsakliga teman på Svenska

Examensarbetets tema och syfte är att bygga upp en kort lista på saker som man ska tänka på när man utvecklar ett 3D spel för mobila platformer som t.ex. telefon och pekplatta.

I arbetet jämförs och testas tre olika utvecklingsverktyg för mobila platformer och på basis av den testningen valdes Unity3D för detta projekt p.g.a. dess bra dokumentation, lätt att använda användargränssnitt, stor användarmängd samt aktiva diskussionsforums.

Spelets egentliga idé och mål är egentligen att lugna ner sig efter en lång arbetsdag eller skoldag, och samt lite reflektera dagens händelser.

Projektets planering hade börjat redan en god bit före själva utvecklingen för denna slutgiltiga produkten var i tankarna. Från första början var tanken att bygga en totalt modifierbar 3D värld inom en öppen spelmotor (Open-Sim, baserad på Second Life-motorn) men efter en tid märktes det att det var en dålig idé att ge så mycket frihet till användaren. Idén var också att ha denna värld på en USB sticka som man kunde plugga in i vilken Windows PC och starta spelet. Det var inte tillräckligt flexibelt eller användarvänligt.

Valet gjordes att flytta oss till mobile platformer på basis av diskussioner med målgruppsrepresentanter, och det var helt klart bättre valet. Man har tid att reflektera på sina dagliga händelser medan man sitter på bussen eller tåget på vägen hem från jobbet eller skolan, och kan då kanske ha nytta av en sådan applikation som nu utvecklas. Man hinner inte nödvändigtvis plugga in USB stickan hemma och ta tid för endast lite reflekterande.

Några problem kom fram under utvecklingens lopp. Upp och ner flygande var ett problem p.g.a. fysikmotorns begränsningar, eller snarare egenskaper. Detta löstes med att begränsa användarens mobilitet inom spelvärlden.

Återspelningsvärde var ett annat problem. Hur kan vi få användaren att spela spelet pånytt t.ex. nästa dag? Lösningen till detta var redan inne i spelet, nämligen ett frågeformulär i slutet av varje spelsession. Man tar några frågor om hur användare mår och känner sig före och efter spelet, och valen där reflekteras i följande spelsession med t.ex. byte av färger, komposition av prylar som flyter runt i spelvärlden osv.
Utvecklingsteamet, egentligen brist på utvecklingsteam var ett problem. I fortsättningen rekommenderas ett större team för att ta hand om olika aspekter i projektet.
Ett stort problem var också frågan om hur vi kan optimera spelet för ett så varierande urval av hårdvara, som t.ex. Android OS telefoner (av vilka det finns hundratals olika). Det finns ingen direkt lösning till detta, förutom att antingen sätta hårdvara/mjukvara begränsningar, men detta kommer ju att rensa bort en stor del av marknaden och därmed användare och potentiella kunder. En till lösning kunde vara att helt enkelt välja en platform som spelet skall utvecklas för, och då optimera det för all hårdvara som finns för den platformen.

De fyra punkter som summerar bra en 3D spels utvecklingsprocess till mobila platformer:

- Planera
- Återplanera och anpassa planen
- Välj utvecklingsplatformen noggrannt
- 3D modeller behöver inte vara detaljerade

6.2 GLOSSARY OF TERMS

MMO-game = Massively Multiplayer Online game
SDK = Software Development Kit
Google-fu = Ones ability to use google to its full potential as a search engine