Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Implementing touch interaction in a casual mobile game

by

Gustav Andersson

LIU-IDA/LITH-EX-G--14/057--SE

2014-06-03

Linköpings universitet

Linköpings university
SE-581 83 Linköping, Sweden

Linköpings university
581 83 Linköping

Final Thesis

Implementing touch interaction in a casual mobile game

by

Gustav Andersson

LIU-IDA/LITH-EX-G--14/057--SE

2014-06-03

Supervisor: Erik Berglund

Examiner: Anders Fröberg

SUMMARY

This thesis is to help developers in the touch design of casual games. The thesis work was made on the development platform Gideros Mobile and much of the thesis theory and methods will be based on that work. The purpose of the report is to show what one should think about when making a casual game in a touch design perspective, like taking into account that most people are used to one type of input design (like swiping or tapping), that you should design the game with the purpose to reach out to as a large audience as possible (taking into account people with disabilities) and some small notes on what should be avoided so not to deter people from wanting to play the game. It is important to note that this thesis is built for the purpose of mobile casual games, others will probably not find this thesis relevant.

INTRODUCTION

Motivation

It may seem simple to sit down and develop a game out of your own thoughts, but in reality it's very simple to get sidetracked and develop the right features for the wrong game or just simply the wrong features. You get an idea that sounds good in your head, but when you start to implement it, it will turn out to be incredibly un-adapted for this type of game genre or console, or that you are trying to expand on an already existing feature, just to find out that it now became unnecessarily advanced for a casual game.

How then can a mobile game with a touch interface be designed to attract a casual-gaming audience? That's what this report will focus and investigate further upon. What should be avoided, what should you should yourself on instead and what should be thought of before you start to implement new or improve old features.

Purpose

The purpose of this thesis is to give tips and pointers on how to develop a user friendly touch interface application and what you should refrain from doing. The thesis is based off the work I did on the mobile game application "Gravel", made in Gideros Mobile[1] in which I was responsible for interaction design, bug fixing, feature implementations and optimization.

Research question

How is a user friendly touch-interaction interface designed for a casual game?

Casual mobile gaming development is quite different from normal touch screen application development or PC application development. So what should you think of before starting to develop such an application and what should you refrain from doing?

Scope

My work on this thesis is limited to the following parts:

- Interaction design for Casual games
- Gravel
- Gideros Mobile

BACKGROUND

The thesis is built upon my work I did on the mobile game application "Gravel" made for a casual gaming community. We were three students working in a group with different thesis directions, I initially started my work by working on the game engine in general, removing and fixing bugs, adding new features, improving or removing old features and optimizing the game to make it run smoother.

Gravel

Gravel is a physics-based puzzle game with a space setting (see figure 1). The player can place gravitation nodes that will move the ship forward or sling it in other directions, so that the player can reach a goal. In between the player and the goal there are obstructions like asteroids, pre-placed gravitation nodes and planetary objects. The goal of the game is to reach the goal using as few gravitation nodes as possible. The game also has an editor mode where the player can create his own maps and share via Facebook to other players.

At the beginning of the project, the game was designed to have gravitation nodes that dragged the player into its center and repulsion nodes that pushed the player away. The repulsion nodes, however, were removed at a later stage. The game also had a set amount of gravitation nodes that the player could place on the map. This was remade so that the player now have an unlimited amount of

[1] http://giderosmobile.com/

nodes, but each time the player uses one, the score gets affected, so using less nodes will improve your final score.

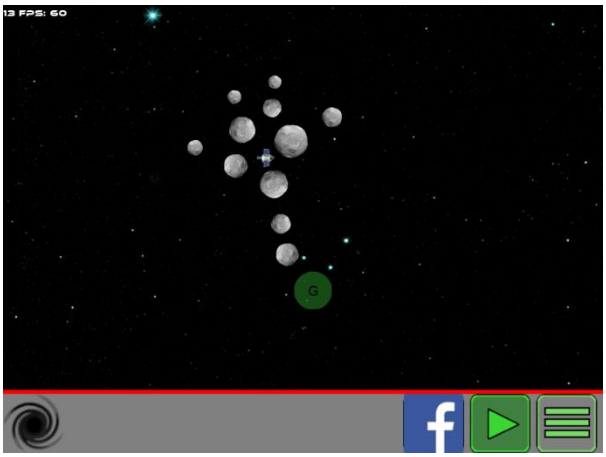

Figure 1. In-game screenshot of Gravel.

My work

Later in the project as we started closing in on a more finished product, my work got more focused on playability on mobile devices. Since the development was made on a computer we did much of the unit testing on a built in emulator that Gideros Mobile has. This led to that most of the testing was done using a mouse as the point and click device, instead of a finger as it should. So when we started doing unit testing on actual mobile devices we quickly saw current designs that did not work well with finger precision contra mouse precision. So I got more involved into making the game focus more on finger precision, making hit-boxes for various objects bigger, changing menu handling so it is easier to navigate and use objects, adding features for visual help since it can be hard to see when your finger is in the way and also removing features that were too advanced for a simple mobile game but worked well on a PC.

I also had to redesign it for casual gaming, simplifying various things in the game to make it more accessible to play on the go. The game had to be easy to get into and easy to understand without having a tutorial-screen, so the game had to be redesigned quite a bit from the original concept to make it happen.

Casual games

It is important to have your mind set on what a casual game actually is before you design its interface and such. A mobile casual game stretches to as wide an audience as it can and tries to keep a simple and easy-to-get into gameplay. Usually the genre and gameplay type is unimportant as long as it keeps to the wider audience, so violence, harsh language, sexual content and destructive content might deter certain people from playing a game [4]. The game is also supposed to be easily accessible, not only from a purchase point of view but also from the playable point of view. This means that you should be able to start up a game and be able to play directly within seconds rather than minutes [4].

THEORY

User knowledge

Most people are used to certain gestures on mobile devices and deviating from these can produce confusion among the users [2]. These are some common inputs most users recognize [7]:

- **Tap**: To press or select a control or item
- **Drag**: To scroll or pan, that is, move side to side. To drag an element.
- **Flick**: To scroll or pan quickly.
- **Swipe**: With one finger, to return to the previous screen, to reveal the hidden view in a split view (iPad only), or the Delete button in a table-view row. With four fingers, to switch between apps on iPad.
- **Double tap**: To zoom in and center a block of content or an image. To zoom out (if already zoomed in).
- **Pinch**: Pinch open to zoom in; pinch close to zoom out.
- **Touch and hold**: In editable or selectable text, to display a magnified view of cursor positioning.
- **Shake**: To initiate an undo or redo action.

These input references are based of iOS UI Design Basics – Interactivity and Feedback.

General design

Limited space and simple interaction

Since most devices have relatively small screens, the design should be in focus for them [2]. It is probably better to avoid harmful operations such as deleting files or renaming files, application shut down and similar operations [3]. Advanced features that require several inputs are to be avoided. Fingers also vary in a wide spectrum of sizes, so design in the perspective of larger hands and fingers, having buttons too close to each other is

probably going to make some people annoyed, a hit target for a control button is recommended to have a size of 44x44 pixelpoints [2, 7].

Elevate importance and be consistent

Simplify important events by highlighting them in various ways [3, 2, 7]. An example can be to gray out the background when opening a menu (see figure 2) and keep everything consistent; if your main menu works in method X then all menus that has a similar use should also work in method X since that is what people will assume. Inconsistency will confuse people and they will probably spend time figuring out why it is inconsistent, rather than just playing the game [7].

Figure 2. Example on how to make an overlay menu and how to highlight its importance

Screen resolutions

Since not everyone on the planet uses the exact same device with the exact same resolution, it is important to make an application with the focus on various resolutions. Hardcoded sizes usually won't work well, like a button that is normal sized on a large device will be gigantic on a smaller device. Programming with the focus on resolutions will also keep true to consistency.

Design swiftness

Casual game users want swift and easy task completion. Design with the perspective of completing various tasks fast and simple [3, 5], like lessen repetitive tasks and reduce number of operations to reduce regular tasks.

Keeping it simple

Simple is the key factor for a casual game, meaning that if you have to stop and read to understand or having to navigate a large array of menus then you are probably doing it wrong. But it's not also advanced gameplay or bad menu-design that will ruin a casual game, it is also important to have a simple uniform design overall, like keeping colours to a minimum. Too many colours (especially on a small screen like a smartphone) will most likely confuse people and destroy the functionality of the application [3, 4].

Colour blindness

About 8% of all males and 1 in 200 of all females have some form of colour blindness. Colour blind does not mean that people with the disability see everything in black and white, normally it is red green or blue light that they are unable to fully see[2] (see figure 3). One thing to think about when designing for colour blind persons is that people with this disability can still see clear differences in contrast, hue, brightness and saturation [8]. The easiest way to select colours is often to look at a colour wheel and choose one colour then the direct opposite on the other side, searching for colour scheme designers will lead to tools usable in this area[3]. Try keeping to a low amount of colours as well since using too many colours on a small space is prone to confuse people, even people with normal sight. Taking colour blindness into consideration will most likely help with ordinary design as well, as good contrast makes for good visuals, keeping different colours at a low has also a chance of increasing consistency overall (unless the game is designed in such a way that a multitude of colours is needed).

[2] http://www.colourblindawareness.org/colour-blindness/

[3] http://colorschemedesigner.com/

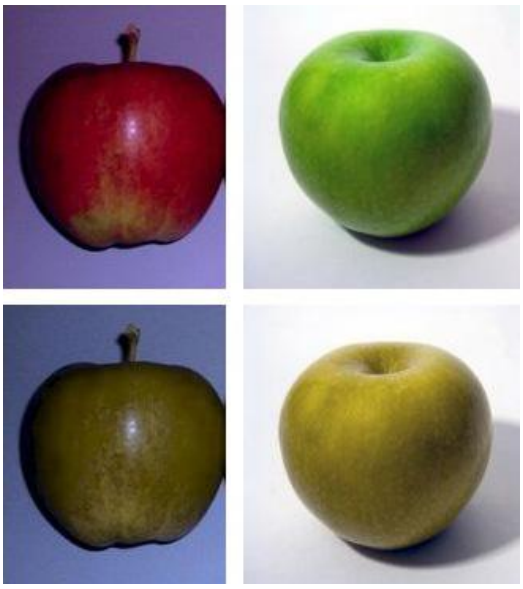

Figure 3. How a red and green apple looks like to a person with dichromatic perception[4] (above picture depicts red and green coloured apples, beneath is their counterpart when seen through dichromatic perception, both appearing yellow in colour).

METHOD

Read and learn

When implementing a new design, one has to do more than just listen to the feedback and redesign the old feature. One should also read a little bit on what designs are common and expected, otherwise it is easy to just snowball back and forth between new concepts that never really work. This means searching for articles with similar interests and searching the internet in general for design layouts like iOS Human interface guidelines and Android's user interface API guide [6, 7]. On this project, some time was spent figuring out what motions and designs most people are used to, so that it could later be implemented into the project.

Work method

We worked agile[5], meaning that we had a lot of meetings and talked a lot with our customer, this lead to quick and many changes for the game. Some systems went through several iterations before reaching its final design and some were scrapped altogether. This working method was best suited for us since we were such a small group and could have lots of interaction between each other and the customer.

Detection and implementation

The problem with this project was that all core features had already been designed and implemented. Leading to that many bugs was and design issues were in the core system and that had to be redesigned. This chapter will go more into detail about some of those core features and how they were improved.

Bottom interface

The interface at the bottom of the screen (see figure 4) in the games editor mode could be swiped and items could be selected and dragged at the same time. However, if the item you started dragging overlapped another item in the interface, it got deleted. So a feature was made that locked the swiping of the menu when an item had been selected, but if the item is dragged too much in the x direction the item is deleted and swiping is instead activated. If you drag the item up and out to the map however, then the swiping will never be activated. This fulfilled the feature of both being able to swipe and drag out items at the same time without them colliding with each other too much, as well as keeping a simple interaction. A "hidden" feature was also made so that if the item is held longer than a selected amount of time the swiping method is also disabled all together, letting you move the object around in the interface without having to worry about activating the swiping method.

In a relevant note, the swiping menu was locked when playing the game normally (when not in editor mode), this was added because previously the game could still activate the swiping method when playing, but it made no sense because there is only one usable item when playing. So if the player dragged the object to the right inside the lower menu, the item would get deleted and the menu would active its sweeping method without being able to sweep. So it got changed so that the player can just select the item without having to be worried about activating the slider.

Items selected in the interface also get highlighted, suggesting that they are being used. Before nothing happened other than you started to drag out the object you chose, but you had no indication that you had actually selected it.

[4] http://en.wikipedia.org/wiki/Dichromacy

[5] http://en.wikipedia.org/wiki/Agile_software_development

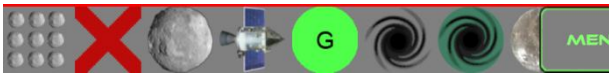

Figure 4. The lower interface and its objects

Menu interaction

Opening menus inside the game never really froze anything. Buttons behind the menu could still be activated and objects moved. So an easy fix of freezing all movement behind an overlapping menu fixed these problems as well as removing the risk of unseen harmful tasks that could have been activated, such as exiting the map, deleting objects on the map or placing out new items.

A pop-up menu (see figure 5) also lacked scalability for different resolutions and sizes of mobile devices, meaning that if a smaller device opened the pop-up menu, it wouldn't be able to see all items. So a swiping method was added to that menu system. However, we reduced the number of items in that menu so this particular problem probably will not arise too often.

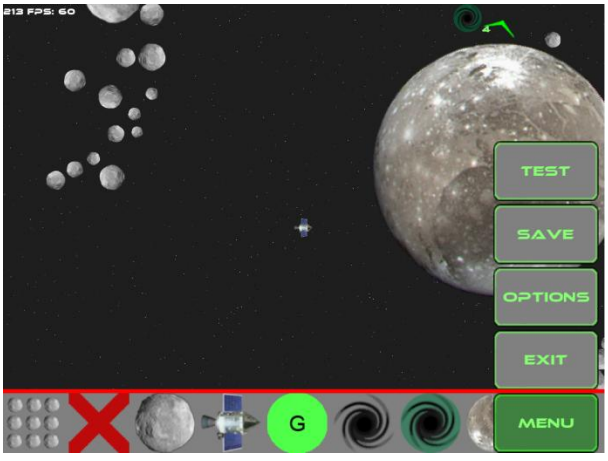

Figure 5. Picture of the pop-up menu

Menu-closing got an improvement as well; before you could only close a menu by clicking on the same button that opened it, however now you can close it by just pressing outside of its borders, making it easy to quickly get back into the game without precision targeting the menu-button as before.

Reducing repetition

Object placement

Asteroid placement and deleting objects were improved by getting a brush function. Instead of placing an object one at a time, or deleting an object one at a time, a brush feature was added so you can quickly place or delete hundreds of objects in positions of your choosing in a matter of seconds, lessening the repetition of tasks. The delete-brush also got a drawing area, so that you could see exactly which items would be deleted. There were some old testing by having a rectangular box you could place; this box could change size by double-clicking it and then dragging your finger outwards to increase its size, when triple-clicking the box would either create asteroids or delete all objects within the box depending if you were placing asteroids or deleting objects. However, this was scrapped because it was much too difficult and time-consuming to use on a mobile device, it got replaced by the brush system.

Starting the game and menu movement

Initially the game started with a main menu (see figure 6), where you selected to play a new map or if you wanted to create a new map in editor mode. This was changed at a later stage so when you start the application you are sent directly to the game with a map already started. Inside this game you can then open a pop-up menu that lets you open editor mode or select other maps, lessening the amount menu-movement (tasks to achieve a goal) [1, 3, 4, 5].

Figure 6. The old main menu system

Testing

To actually know if a design was flawed or not it had to be tested on an actual mobile device. This

was done privately within the group and also shared to our customer, so that he could give constructive feedback to us, it is quite easy for developers to miss if the design they have made for an application is good or not since they already know how things work, something the customers does not. We in the group later analyzed the feedback we got and redesigned whatever feature had gotten complaints on so we could do a new testing of it to see it worked.

RESULT

Work method

The agile method of working proved very good for this project. Lots of changes were needed and lots of testing was needed, the agile work let us achieve this.

Detection and implementation

Bottom interface

With the new locking system for the bottom interface, there is a lot less inconsistency with how items are handled (deleted and moved), so the player should be able to use and understand those mechanics with less frustration.

Menu interaction

Compared to its original design, the new system is greatly improved. Before it was very confusing and frustrating to use the menu systems since all buttons behind overlapping menus still worked, but with the new it became much more user friendly. It could be very confusing since the lack of button-locks did not only apply for local menus opened in the normal game; it also applied when navigating the menus at the start. By locking all buttons not currently used in the currently opened window, the game became much more consistent and useful.

Reducing repetition

Object placement

The brush function was a big addition to making the game more casual. The old features of drawing a rectangular box was not only complicated for mobile systems, it also looked ugly and did not fit with the core concept of casual gameplay and touch interaction. The new system is more in tune with mobile gaming and it enables the player to delete or add objects to the map in a much faster rate than the old rectangular drawing system did.

Starting the game and menu movement

The new system of directly starting with a playable map lets the player skip all the boring parts of menu interaction before being able to play the actual game. Now the player can start the game application and play the map in a few seconds and if the player does not want to play the auto-selected map, then he can just choose a new one by quickly opening the menu at the bottom interface. This is all to reduce meaningless tasks, there is no real time-loss in starting the game with the wrong map and starting the game with menus from where you then choose the map you want (the time to load a map is quite insignificant).

Testing

The testing was adequate, since it was only composed of the programming group and the costumer. Information regarding different designs was criticized in such a way the programming team did not have any insight in and so lead to an overall improvement for the game. It also lead to extensive debugging, letting the team find out what worked and not.

DISCUSSION

Result

Bottom interface

The interface in its current design is more true to consistency and reduces confusion more than how it was in the beginning. A lot of operations made in its pre-redesign made no sense, as the lack of highlighting. Often the item you had selected is underneath your finger and hard to see, so it was hard to actually know if you had selected and started to drag it. The addition of highlighting the selected item removed that confusion all together.

Its old method of removing the selected object if it passed over another object in the interface also made no sense to the user. There was nothing to indicate what caused it and there was no apparent indication that a change had been made. With the current feature that lets the object stay in your possession until you activate the slider lets the user at least know what caused the item to disappear (the start of using the slider), it is not a perfect solution, but the lack of screen space made it hard to implement some form of buttons that could replace the swiping method. The swiping method is probably more expected on a mobile device

anyway, rather than to have two buttons that you press to show items that are hidden due to lack of space.

You can also hold the object down for a small time to disable the swiping method, however this lack any form of indication at the moment and will probably not have one either. This is both good and bad, good because the timer would only be shown for fraction of a second, probably not showing much information for the player and most likely confuse the player and forcing him or her to repeat the task over and over to actually see what is happening. And bad because there actually is no indication, in one instance you moved the object too fast to the side and activated the slider, but in this instance the slider didn't activate and the player is not sure of what caused it, making him or her repeat the task to figure out why. However, since it's not such a huge time consuming operation and that the player will probably figure it out quickly it is a matter that is not worth (for this project) looking into too much.

With the normal play-mode I disabled the swiping method all together; having the object be removed because you moved it slightly to the side will only confuse the player and not serve any good purpose since the interface cannot be swiped anyway. It also makes it easier to play. In an earlier build the selected object (the gravitation node) got deleted when moving too much to the side. This made it hard to quickly place an item on the map since you had to move it straight (with a little margin to the sides) up and out of the interface before being able to move it freely. With the new system the player can instantly drag out the node without having to worry about it accidently being deleted.

Menu Interaction

There were three ways of handling the pop-up menu that were too big for some screens (and didn't show all items).

1. Shrink the buttons so the menu fits the screen
2. Add "up and down" buttons that you can press to move the menu.
3. Add a swiping method to move the menu.

The first alternative would probably have been the worst. On bigger screen application this would probably not have been a problem, but since the buttons already are on the smaller scale (as small as we dared to make them without making them hard to press), shrinking them even further would have made it too hard to hit them or too hard to read the text they encapsulate.

The second alternative would just add two new buttons, taking up even more space and since there is no other such feature in the entire game it would most likely only look weird to suddenly have two buttons that you use to iterate through menu items.

The third alternative seemed most native to the game; since the bottom interface already has a swiping feature it would only be natural for other elements to have it as well, it would also not take up any other space on the screen.

The addition of being able to close the pop-up menu by pressing anywhere outside of the pop-up menu might also be debatable. On one hand you might accidently press the outside of the pop-up menu borders and close the menu, continuing the game. But since the game is made in such a way that the maps are small and can be done very quickly, I deemed it a lesser evil compared to being able to get quickly into a game again by not only being able to close the pop-up menu by pressing that single button. It might add to a small amount of confusion as to why the pop-up menu closed when you do it the first time or so, but the player should be able to figure out this feature instantly.

Object Placement

This is probably one of the major features to reduce repetitive tasks. Instead of placing one asteroid at a time the player can now just select a brush and instantly draw out hundreds of asteroids, this particular feature was not made by me but I adapted it to the delete brush, so that the player can delete hundreds of objects with good precision in a matter of seconds. Before this the player was forced to select items one at a time and move them down into the interface for them to be deleted again, or they had to create a new blank map. I later added so that you can see the deleting area, so that you might not accidently delete things because you could not see the exact area of deletion. The delete area also depends on zoom level of the game and how fast you are swiping back and forth with the delete brush. Without the addition of the delete area drawn it would have been quite hard to understand the function of how big the delete area is.

Starting the game and menu movement

This idea was brought up by our customer to let players quickly get into the game without having to navigate through a bunch of menus, so now the player will be sent directly to a started game and from there he or she can open up a pop-up menu and navigate to other parts of the game, like the editor menu where the player can create new maps or the level menu to select other playable maps. Since the game is focused on being able to play fast short games, the old design was a step backwards in that direction since the player had to do several menu interactions before being able to start a game. Now the player can either start directly with the already started map, or open the pop-up menu to quickly select a new map.

Method

Read and learn

It actually was a bit hard to find good information, some information like Androids and iPhones design sites gave proper and easy way of understanding on how things should work, but overall there is not that much written about this subject. It can be quite hard to find concrete information on how one should design an application; it is often hidden in texts regarding other issues. So I found this part a bit hard, but I still found good information regarding the subject.

Work method

There is not much to discuss about the work method, agile working is what we are used to and have used a lot. Our customer also let us work freely without much supervision, so the agile working method just fell into place. A waterfall method or a strict supervisor method didn't really fit as well, mostly because the waterfall method is (according to me at least) bad and since we didn't have a supervisor or that anyone in the group wanted to be a group leader, the agile working method was most natural for us to begin with.

The implementation of the systems

In the essence, how we improved the systems was by finding issues with current designs and then changing them. Given more time we might have redesigned some systems wholly, rewriting them from scratch, but given the limited time and that we did not find certain issues after a while, the method we used was most likely the most useful for us. The changes listed in the previous chapters are those changes made that I found most fit, there might possibly be better ways to design them but if so, I have not discovered them yet. There still are designs that I would have wanted to work in different manner but due to existing code and how Gideros Mobile works, sometimes I am forced to let certain things slip by.

Testing

Testing out our functions was probably the most important part even though we in the group did not really list testing as a form of working criteria (we just did a lot of testing as we worked). Since all the functions and gameplay elements we created were from scratch without any real templates to work from, testing was the only way for us to know if what we had created was actually good or not. Actually, a lot of features we created were scrapped because they were deemed too complex for the game, or it did not fit into the game from an aesthetic viewpoint. Letting our customer test out the builds also gave us important viewpoints from a perspective we impossibly could have seen since we already knew exactly how everything worked, so having people testing it that don't know the project was really helpful and it gave us the insight of which gameplay materials ruined the game overall. However, we could probably have gotten more and better feedback from if a greater testing group could have been gathered, though that would have been time consuming and almost worthy of a thesis of its own.

Reading up about design and looking at other games was really helpful as well, of course it is good to do new things but there is also a reason to why the other games are so popular. So it is always worth taking a good look at them and analyzing them, so that you can incorporate popular elements of those games into your own. It does not specifically have to be gameplay related elements but it could also be how they handle marketing, menu systems or viral sharing. I for one looked at how iPhone and Android designed their UI, of course it is hard to try and push that into an existing game but there is always some improvements that can be made, either that or it might be useful to redesign an element all together.

My thoughts about making a casual game

Making the game for colour blind people wasn't really specified anywhere, however the game coincided with being accessible for them automatically when designing the game. Since it is in space we have a black background (with stars)

and there aren't many overlapping textures since almost everything have physics that make them bounce off of each other. So the game automatically had somewhat bright colours on a black background, meaning high contrast. The only time overlapping textures happen is when a player enters the gravitational area of a gravity node; however these colours have also been taken into consideration. Making a colour scheme should help keeping colours right.

Some of the hardest parts of making a casual game is keeping it casual. It is incredibly hard to not go overboard with the gameplay by adding new features because they look cool, expand systems so it becomes hard to interact with and making gameplay advanced by adding several ways to do things and suddenly you sit there with a game that needs several hints or help screens so that the player will know what to do. It might also become hard to jump back into the game if the player has been away from it for a time.

Consistency is also very hard to achieve, sometimes you make a menu for one part of the game but at another similar part of the game you must remake the menu system but with new elements, but this time you accidently redesign it a bit without thinking. So when the player gets to the new area with the new menu he or she won't be able to as easily navigate due to the confusion brought up. This could be seen when playing the game normally and testing out a map created in the editor. At the normal gameplay session the pop-up menu and some items in the lower interface looked in a certain way, but when testing out in the editor the lower interface and pop-up menu suddenly looked different and had items put in different places. The cause for this was how the game had been designed (code-wise) by the previous developer; the game lacked general classes for implementing a full interface, instead the interface was created at each instance for each map at full.

To prevent this, a design document could have been specified; writing down what the game should do and not do how menus are to be designed, what kind of graphics the game should use and the type of controls. So when creating new features for the game, or re-doing old ones in new places, you can look at the design documents and know exactly what it should look like, how it should work and where it should lead. However, due to already spent time and what work had to be done, making a design documents for this project at this time is somewhat unnecessary.

The work in the wider context

One problem about making games in general is that a lot of large companies try and copy already popular existing ones (an example is that World of Warcraft have set the standard MMORPG[6] gameplay since it came out 2004, making developers create clones of World of Warcraft rather than trying to make innovative ideas) or not daring to push new and innovative ideas because it is too much of a financial risk-taking. However with the mobile game industry it is different since it can be easier to make a popular game (compared to consoles or PC). The mobile industry is still in constant development due to frequent upgrading hardware and new mobile designs; this also makes games change their designs constantly. In my opinion, this makes it easier to develop popular casual applications for mobile devices compared to other platforms, due to how you can create and realize new and fun ideas never done before. Of course, this depends on how you look at it, for small casual game applications, a mobile device would be easier to develop for, but if one would want to create a story rich adventure game with advanced graphics, a PC or console would be the obvious choice.

CONCLUSIONS

In the end, Gravel got some big updates in its functionality towards interaction design. However, more could have been done if my thesis from the beginning would have had its focus on interaction design, since most of my work was more focused on bug fixing and feature implementation.

Research question

How is a user friendly touch-interaction interface designed for a casual game?

As depicted in this report, a big part on how to do this is research and preparing, since this thesis is built on an already started project and that my initial work was not in the focus of design this could not be adapted to my practical thesis work. However I do strongly believe that preparing before

[6] http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game

and specifying what should be done is important in making a good game design. Simplicity and consistency should be the guidelines for the game and using standard and known touch interactions should be the design for touch interactions (using motions and triggers that people already know how to use will make it simpler for them to understand the game). Organize a lot of testing with people outside of the programming group to actually know if the features added are easy to understand or not and lastly one should also not forget about people with disabilities. It's easy to forget about disabilities like colour blindness, arthritis, joint pain and such, but there still are a lot of people with these problems. So designing for them will reach out for a larger crowd as well as most likely making the overall design of the game simpler and more effective (lessen tasks, easier touch interactions and such).

The purpose

I believe that the purpose of this thesis has been fulfilled to such an extent that I could, the only problem being that I iterated through several research questions before arriving to the final one that I am using for this thesis. So some time was "wasted", in a sense, on spending time doing things on Gravel that was not related to interaction design at all. However that work was well needed anyway since the other two people in my programming group had their work to do as well, my initial parts of stabilizing the game did make it more playable.

Future work

I will work more on the project for a while after this thesis is done, completing things that still is not fully done, stabilizing the game a bit more and preparing it for release. Since the game is so far done and almost ready for release there is not that much that I want to do to the game that is not already done. On another note, I would have liked to do the design specifications and colour schemes, making it easier for future workers to continue develop the game but at this stage of the development it is not worth putting time on.

Final words

Even though my work for the thesis was not fully realized until a later stage of the project, I feel like I have learned a lot about what a casual game is and how to do its interaction design. It was quite hard to find information about this as well since it is such a relatively new genre; 2007 was the release date of the first iPhone, 2008 was the release date of the first Android smartphone (HTC Dream). Many popular games have had their release dates quite recently as well; Temple Run was released 2011, Candy Crush saga 2012 and Angry birds 2009 (which was a huge success). What I want to say about that is that there is not much research made for this kind of work; many games do not have many references to base their work on other than a few overly successful games as those listed. With this report I had hoped to fill in a gap that exists in the current generation for mobile games, making it easier for future developers to know what to think of before deciding to create a casual mobile game.

REFERENCES

1. Baharuddin, R., Singh, D., & Razali, R. (2013). Usability dimensions for mobile applications—A review. *Res. J. Appl. Sci. Eng. Technol*, *5*, 2225-2231.
2. Choi, J. H., & Lee, H. J. (2012). Facets of simplicity for the smartphone interface: A structural model. *International Journal of Human-Computer Studies*,*70*(2), 129-142.
3. Gong, J., & Tarasewich, P. (2004, November). Guidelines for handheld mobile device interface design. In *Proceedings of DSI 2004 Annual Meeting* (pp. 3751-3756).
4. Kultima, A. (2009, September). Casual game design values. In *Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era*(pp. 58-65). ACM.
5. Wasserman, T. (2010). Software engineering issues for mobile application development. *FoSER 2010*.
6. http://developer.android.com/guide/topics/ui/index.html
7. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/MobileHIG/index.html
8. http://www.stcsig.org/usability/newsletter/9910-color-blindness.html