# Design of a tactical turn-based game for mobile devices

Ricardo Silva
ricardo.a.silva@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2016

**Abstract**

With the mobile games market growing by the day, more and more gamers are shifting to mobile platforms. Most of the games, however, are very simplistic, which leads to non interesting gameplay for veteran players, or are ports of preexisting games, which leads to poor interfaces and user experience. This project focuses on the second problem, trying to understand what the best user interface/user experience is in tactical turn based games for mobile platforms.

**Keywords:** Tactical, Turn-based, Mobile devices, Strategy, Mobile game, User experience, User interface

## 1. Introduction

Even though there are more games being developed for the mobile market, the same cannot be said so much about tactical Turn-based (TB) games. While there are a few of them (even like XCOM and Final Fantasy tactics, that are games that both the public and press views as being very good) they're not really made *for* mobile devices and so they don't have the constraints in mind, are just ports of pre existing games.

This is a case of games that are good on consoles/computers not fitting the target audience, being a game far too complex for the system in question.

While there are handheld consoles on the market, that on the surface face the same challenges as mobile games, that is not the fact, as handheld consoles have both a different audience (comprised more of hardcore/regular than casual gamers) as well as different control schemes, since the consoles have some form of physical input, unlike mobile devices.

The objective of this project is, therefore, to create a game that can suit the mobile market well, both in terms of interface design as well as game mechanics that suit the target audience, with a greater focus on the interface/user experience.

## 2. Related work

In this section, we will define some important concepts in regards to turn based games (such as varied ways of implementing turns) and tactical games. Afterwards we will look at constraints that mobile development brings.

### 2.1. Turns based types

In order to design a TB game, it was necessary to know what defines a TB game and what types of TB mechanics exist.

A TB game is defined for having its game flow divided into individual phases, where the game is paused, called turns. Depending on the TB variant used the turns progression may vary, but overall turns are used to give players more time to think about their actions and not focus so much on keeping track and controlling several things at once in real time [6].

#### 2.1.1 Classic turn based

In its classic form, a turn ends when one of the players executes the actions of all of the elements that he controls. This allows for the player to have unlimited time to think about its actions, because the opponent only acts after the player acted. In multiplayer computer games, since the unlimited time system would be easily abusable, each player usually has a time limit to perform his turn (but it is usually more than enough so that the player can still think carefully about what he wants to do).

#### 2.1.2 Active Time Battle (ATB)

In this system, usually used in party based combats, each element of the party has an action bar that fills over time. When the bar is fully filled, that element executes the action that the player queued for him.

This system also allows the player to have time to think about his actions, because the action bar takes a while to fill, but gives some sense of urgency, because if the bar gets completely full and that character doesn't have an attributed command, it will just stay still, wasting time and giving more opportunities to the enemy to act.

#### 2.1.3 Simultaneous execution

In this system, both players decide what they want to do at the same time, and when the turn advances, both players' inputs are executed simultaneously. Even if this looks like a minor variation, it leads to one very important (and major) difference in the way of thinking about the moves. While in the previous variations the player knows how the state of the system will be when his actions are executed, it can plan them with 100% confidence that his actions will occur how he ordered them to.

In simultaneous execution, however, the state of the system when the actions are executed is unknown to the player, so he has to plan them while trying to guess what his opponents will do and vice versa, which can lead to much more complex train of thoughts.

### 2.2. Interesting mechanics

In addition to the different types of TB, there exist several mechanics transverse to any kind of TB game. Those mechanics, if used correctly, can easily make a game's strategic space increase (but one has to be careful as to not over saturate a game with mechanics, as that may make the game more complex than it is supposed to, leading to confusion amongst players).

All of the concepts presented in the next sections have the objective of making the game have the following qualities [9, 10]:

- Emergent complexity, meaning it creates complex gameplay out of a simple set of rules. This is opposed to having a game that's complex by having hundreds of pages of rules;
- Clarity, meaning the player has an immediate feedback and understanding of the result of its actions and decisions;
- Determinism, meaning that the game, even if it uses some sort of random, gives enough tactical tools and agency to the player, which results in proper strategies working the majority of the time.

#### 2.2.1 Variable terrain

Terrain should be diversified, both in terms of positive/negative buffs on specific spaces (a character on a fortress should have more defense than an uncovered one, and walking through mud should be harder than through flat terrain) and in its design (a narrow and long bridge creates a nice chokepoint, while a large plain creates space for spread out strategies).

If also coupled with classes specializations, more interesting dynamics can be created, by making certain classes have specific interactions with some terrain (maybe flying units can go over water but none of the others can).

However, this concept can be pushed even further by allowing the player itself to manipulate the terrain, by destroying/creating some paths (eliminating some passageway, creating choke points, etc) or by placing traps.

This would make it so that the player must think not only where to place his characters according to the terrain, but also how to change the terrain to place the characters where he wants them to be.

Games where the mechanic is used: Final Fantasy Tactics, Fire Emblem series and Advance Wars (different terrains have different movement and/or combat modifiers). XCOM series (some spaces can be poisoned/on fire, and height plays an important role on the combat).

#### 2.2.2 Directional facing

If the characters (both allied and foe) suffer more damage from the sides or the rear side, it forces the players to think where to turn their characters to, instead of just where to move them to.

This concept can be used both offensively and defensively, because the players must think where to face his characters so they don't take extra damage, but also how to place them so the enemy cant avoid taking extra damage from someone (pincer movement).

Games where the mechanic is used: Final Fantasy Tactics (units hit from the back or the sides take more damage), XCOM (units can take cover, and that cover is directional as it only covers the units from certain angles).

#### 2.2.3 Several controllable characters

The more characters a player has, the more it has to think about what to do with each one of them. Obviously, this creates more strategic diversity, because the number of actions a player can make increases exponentially with every new character available.

This, coupled with the space dimension, takes this concept even further, because the player has one more action to do with each character (where to place him).

### 2.2.4 Specialize the characters

This one ties up nicely with the mechanic explained previously. Having a lot of generic characters isn't interesting (nor useful to the game's depth), because there isn't a reason to chose one character over another.

However this differentiation shouldn't just be a case of the characters having different stats, like damage or hit points, and should provide meaningful choices to the player in the form of special actions only available to certain classes. For example in Fire Emblem, some classes like the Bard are so specialized that they don't even have the "basic" actions available to them (like attack) and are used with an unique purpose (in the case of the Bard, it has an action that allows an ally to act twice in the same turn).

On the same note, enemies should also be varied and specialized, making the players think which of their characters' classes are good versus the enemies'.

Games where the mechanic is used: XCOM, Fire Emblem, Final Fantasy tactics.

### 2.2.5 Resource management

If a tactical game has some sort of resource management, it immediately creates some form of risk reward.

This resource could be anything, from mana necessary to cast spells, to number of charges on abilities/weapons, or even available Action points (AP) per turn, strong attacks taking more, but having a high chance to miss. There can also be consumable items that the player can use for some immediate benefit but are gone forever afterwards.

Having such a system would promote a risk/reward gameplay, not only because heavy attacks have a higher chance to miss, but because players can't move afterwards, since it probably used all its AP.

Games where the mechanic is used: Final Fantasy series (mana), XCOM series (cooldowns), Fire Emblem series (attack charges on weapons).

### 2.2.6 Support multiple objectives

Multiple objectives, as in having multiple win conditions (defeat all the enemies, defeat the Boss, rescue someone, etc) already creates some nice strategic diversity and is pretty common place on most games.

However there is something else that can be done, and that is to give secondary, or optional objectives to the player. Those objectives can be something like "rescue a character in x or less turns to receive an extra reward", or maybe having some persistent stat on the players (like morale in XCOM [7]) that affects the characters' performance, and changes based on how the player handled the missions. Morale isn't necessary to completing the missions, but is a side objective in the sense that the player is trying to complete the missions objectives while also doing actions that boosts their character's morale.

Games where the mechanic is used: Fire Emblem (villages [4]), XCOM (meld canisters).

## 2.3. Mobile constraints

Developing games for mobile platforms is really different than developing a game (even when it is the same game, which is the case in ports) for consoles or computers.

There are multiple reasons for that, which will be described in detail in the following sections, but they are:

- Hardware capability;
- Display size;
- Control scheme;
- Interface
- Environmental constraints.

For those reasons, developing a game for mobile should be made carefully and have the design decisions be made with the constraints in mind, and shouldn't be thought as just developing a game that happens to be played in a smaller screen.

### 2.3.1 Hardware capability

With smartphones/tablets being such small devices, their processing power isn't their strongest point, and so, games have to have those factors in consideration, both through code optimization since the beginning of the development, and also by extensively testing in several devices with different hardware to see if the game runs without lag or fps drops. There is a plethora of services that provide testing in multiple devices and operating systems, to ensure that the game runs in as many devices and configurations as possible. In the Android's developer web page there are even a couple of requisites regarding the performance of the apps that go into the play store: "App loads quickly or provides onscreen feedback to the user (a progress indicator or similar cue) if the app takes longer than two seconds to load." and "With StrictMode enabled (see StrictMode Testing, below), no red flashes (performance warnings from StrictMode) are visible when exercising the app, including during game play, animations and UI transitions, and any other part of the app." [1] [1].

Also, the space required to both download and install a game should be taken into consideration, since most of the time the game will be downloaded over cellular networks. This is even enforced, for example, by Apple's App Store guidelines for content put up in the app store ("Apps larger than 100MB in size will not download over cellular networks (this is automatically prohibited by the App Store" [2]).

### 2.3.2 Display size

The mobile devices displays present two problems. First, in most cases, the resolution is fairly small, which limits the amount of content one can fit into the screen at a given time (which includes any menus and interface elements, in addition to the gameplay ones).

Secondly, the aspect ratio varies between different devices, ranging from 3:2 on the iPhone 4 to 17:10 on Samsung Galaxy Tab 7 and it is necessary to have them into account. This can have serious implications in the gameplay, for example on games that need the users to quickly react to an object that enters the scene, like runners. If two users are playing on different devices, one in a tablet (4:3 resolution) and another one on a phone (16:9 resolution), and if the game is poorly made, the phone user could have an advantage because he would be able to see farther then the one in the tablet.

The aspect ratio factor is also really important in a tactical turn based game, because if some users can see more of the battlefield than others, that creates a clear unfair advantage.

### 2.3.3 Control schemes

Controls in mobile games present a great challenge, because it can be really tricky to create intuitive controls with only a few possible inputs (the only three things recognized by a touchscreen are touch begin, touch end and touch move, while with a mouse there is also one extra "action", the hover. It is possible to derive some basic actions with those inputs (a touch begin closely followed by a touch end is a tap, a touch begin followed by a touch move followed by a touch end is a swipe, etc) but it is still hard to create a simple and understandable control scheme with those basic "control primitives".

For example, Super Mario Bros had really simple controls: two directional buttons, one button to accelerate and one button to jump, but even that would be tricky to replicate in a touchscreen (Gamasutra has an article on touchscreen controls, where the author gives a nice step by step description on progressively more complex controls [8]).

However, in a turn based game, most of the controls are menu based, so that won't be as much of a problem.

[1]Like Applause, for example, `http://www.applause.com/mobile-app-testing`

### 2.3.4 Environmental constraints

Mobile games are played mostly while commuting and at home, in bed [5] and with an average playsession duration of 6.8 minutes [3]. Those two factor combined lead to two considerations:

- Games, even if with a multiplayer component, should allow for offline play. This is because if someone is playing outdoors, or while traveling, may not have a good internet connection, and/or may need to stop their gameplay midway, so if the game is purely online it is losing on a big percentage of potential players.
- Games should allow short playsessions. This means that a game "match" (either a level of a puzzle game, or a battle of a tactical TB game) should be short, so that the player can play one entire match in a short period.
- Games should also be careful with the usage of sound, as many people play mobile games while traveling, so they may not have the sound enabled. This means that sound should be used as a support and not as a fundamental part of the game.

## 3. Implementation

Since the objective of the project is to study multiple interface options, the gameplay was very simple. It consisted of the following characteristics

- 4 units with the following attributes: Hit points (HP), Range (Ran), Attack (Atk) and Speed (Spd);
- One terrain type: where one point in Spd equates to one square per turn;
- Simple combat: whenever one unit is in range of another one and attacks, the attacked unit loses HP equal to the attacking unit Atk;
- One game mode: the first player to destroy all the opponent's troops wins.

The main work was done in regards to the developed interfaces. Three different options were created:

- Alternative 1 (Control mode);
- Alternative 2 (Basic range mode);
- Alternative 3 (Auto attack mode).

### 3.1. Control mode

This interface had the following characteristics: upon clicking on a friendly unit, 3 sets of icons appeared. **Green squares** represented the spaces where that unit could move to, **gray squares** represented places where the unit could move but would be in attack range of an enemy and **red targets** represented the max range at which the unit could attack (max range in this case means ”range that the unit can hit if it moves to it’s fullest capability).

Red targets would also appear if the selected unit could attack an enemy unit from its current position (figure 1).

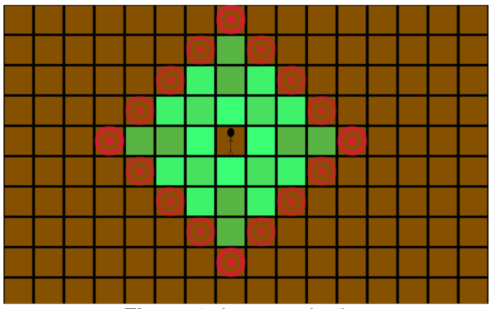


**Figure 1:** Input method 1.

To attack, the players had to move into attack range, then select the same unit again and target the enemy unit they wanted to attack.

This tried to emulate as much as possible the interface commonly used in games.

### 3.2. Basic range mode

This interface had the following characteristics: upon clicking on a friendly unit, 3 sets of icons appeared. **Green squares** represented the spaces where that unit could move to, **gray squares** depicted places where the unit could move but would be in attack range of an enemy and **red targets** appeared on the spots that the unit could attack **from the current position**.

Red targets would also appear if the selected unit could attack an enemy unit from its current position (figure 2).

To attack, the players had to move into range, then select the same unit again and target the enemy unit they wanted to attack.

While it requires as many clicks as the first approach, it is here to see if it’s easier to understand the units’ range with this presentation or the ”max range” one.

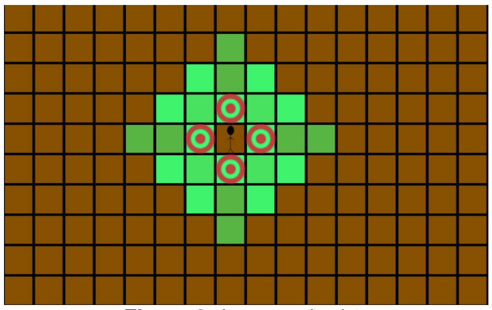


**Figure 2:** Input method 2.

### 3.3. Auto attack mode

This interface had the following characteristics: upon clicking on a friendly unit, 3 sets of icons appeared. **Green squares** represented the spaces where that unit could move to, **gray squares** depicted places where the unit could move but would be in attack range of an enemy and **red targets** appeared over enemy units that the selected unit could attack either in its current position or in any of its possible moves.

This means that if the selected unit currently isn’t in attack range of an enemy unit, but can move somewhere where it gets in attack range, a red target appears on that enemy. Then, by clicking on that target, the selected unit automatically moves into attack range and attacks (in the figure 3 we can see that the airplane unit is currently not in range of the infantry unit, but a target still appeared on that unit).

Alternatively, the players could also move the unit to a spot where the unit is in attack range of an enemy and manually target an enemy, just like in the previous case.

This tried to make it so experienced players (that already know the units and their attribute (att)s) could easily issue the desired commands. So, instead of:

1. Selecting a unit;
2. Moving it into the desired range;
3. Selecting the same unit;
4. Attacking an enemy.

Players could just:

1. Select an unit;
2. Attack an enemy.

This, however, could pose a couple of problems: since this mode it doesn’t present the attack range as clearly, it could be difficult to understand all of the units. Also, it could give a sense of unpredictability, as the players don’t know the path the unit will take in order to perform the attack.

This is the reverse of the previous one. Instead of making it easier for newer players, it tried to make the game faster for experienced players.

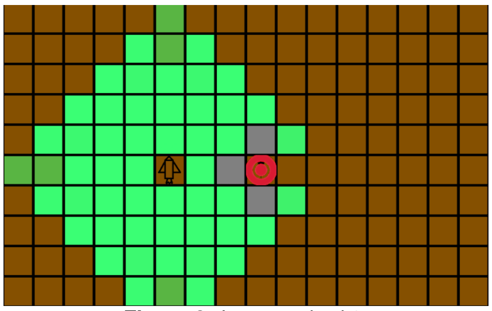


**Figure 3:** Input method 3.

## 4. Results & discussion

With both the gameplay and all the required interfaces complete, user tests had to be performed. The objective was for the tests to be conducted with a minimum of 20 subjects in order to ensure some minimum consistency, but there was no upper bound in the number of subjects.

Each test was conducted with a different interface order, to prevent some result skewing due to one interface being played before/after another one.

### 4.1. Setup

The tests had a simple procedure: since the game didn't have any AI, it was played by 2 test subjects at a time in a versus match, played in 3 rounds. Each round was played with a different interface option, being that the order of the interfaces changed from one test to another. However, the ordering wasn't random nor tried to cover all the different possibilities. This is because covering all the possibilities would make it so very few people played one given option, which in turn would lead to a sample size not big enough to be conclusive.

So, three orders were chosen:

- Order A: Alternative 1, 2, 3
- Order B: Alternative 3, 1, 2
- Order C: Alternative 2, 3, 1

After playing a session, each subject received a questionnaire that was focused on 3 aspects:

- The habits of each subject;
- The playability of the game itself;
- The control modes and interface: this was the main focus of the project and so was the Section that was more detailed. It had questions about each of the interface options and also asked the subjects to order them by preference. Each question will be analyzed in more detail in Section 4.2.

The test was conducted in Taguspark, during a game related event (MOntra de JOgos (MOJO), which could lead to some skewing of the results, as 96% of the population had some prior experience with videogames) and counted with the presence of 32 participants with 10 people playing each of the interface orders.

### 4.2. Results

In this section, we will try to analyze the user responses and try to figure out multiple things: **which** play mode was the preferred one, **why** it was the preferred one and also if the other modes had some features that could be taken and used to upgrade the best mode.

#### Preferred mode

The first thing to analyze is what the preferred mode was, as it will help when deciding which features to look at as good examples to follow. For that, a simple graph depicting how the users organized the different modes is present in figure 4. The graph shows the number of users that placed each mode in 1st, 2nd and 3rd place.

In it, we can see that the difference between modes is fairly well defined, with 23 users (76% of the whole test base) placing the basic range mode in 1st place, 16 users (53%) placing the auto attack mode in 2nd place and 20 users (67%) placing the control method in 3rd place. This shows pretty clearly that most users preferred the basic range mode.

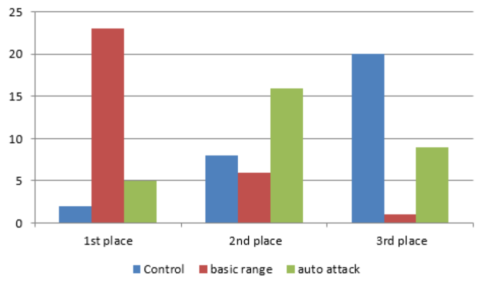


**Figure 4:** Control methods preference

#### Breakind down each mode

To understand why, we need to analyze each mode individually. In order to do so, the users where asked 4 questions for each of the modes (where the users could answer on a scale of 1 to 5, where 1 was didn't understand at all and 5 was understood perfectly):

- Comprehension: the overall understanding of the mode;
- Quickness: how fast the users could execute their turns;
- Reach: how easy it was to understand the range of each unit;

- Vulnerable: how easy to see were the positions where a unit would be vulnerable to enemy attacks.

We performed a Friedman's test to check if the answers to the questions are statistically different between the different modes, which had a positive result with $p < 0.01$.

Then, to understand exactly where the differences occur, we perform a Wilcoxon's signed-rank test between every pair of modes.

Following are the results obtained by said test:

- The control mode is significantly different from the basic range mode with $p < 0.05$ for every question;
- The control mode is significantly different from the auto attack mode with $p < 0.05$ for all the questions, except for the vulnerability question, with $p = 0.08$;
- The basic range mode is significantly different from the auto attack mode only for the reach question with $p < 0.05$ and is not significantly different for all the other questions ($p = 0.426$ for the comprehension question, $p = 0.083$ for the quickness and $p = 0.091$ for the vulnerable). In this case we consider that the quickness isn't statistically different because while the $p = 0.08$ (just as in the control mode - auto attack mode comparison) in this case the median value changes less than in the previous case.

From this analysis, we can conclude that, while some differences were still present between the basic range and auto attack mode (particularly concerning the units reach) the main differences were between the control method and the other two.

Now, to understand how big were the differences, we can look at a graph showing how many users gave which score to each question (figure 5) and a box-plot chart that shows the average value of each question (figure 6).

The fact that the control mode was the worst one is once again present, and now we can see that it was the worst one in every individual category, as well as the worst one overall, so from now on all the tests will be conducted on the other two modes.

Regarding the other two modes, even though the global evaluation was fairly distinct (23/6 users placed the control mode in 1st/2nd place, while 5/16 users placed the auto attack mode in the same positions) that difference isn't as evident when comparing the separate categories.

Firstly, as we analyzed previously, most of the answers regarding this two modes aren't statistically different, so any conclusions drawn from this information is prone to be dismissed with extra testing. However, there are some relevant factors that can still be observed.

For example, even though the answers are pretty similar, we can see that, in average, the basic range mode has slightly better scores overall than the auto attack mode. This very slight difference could be the reason why users placed the auto attack mode in 1st place a higher number of times. One of the possible explanations could be that both

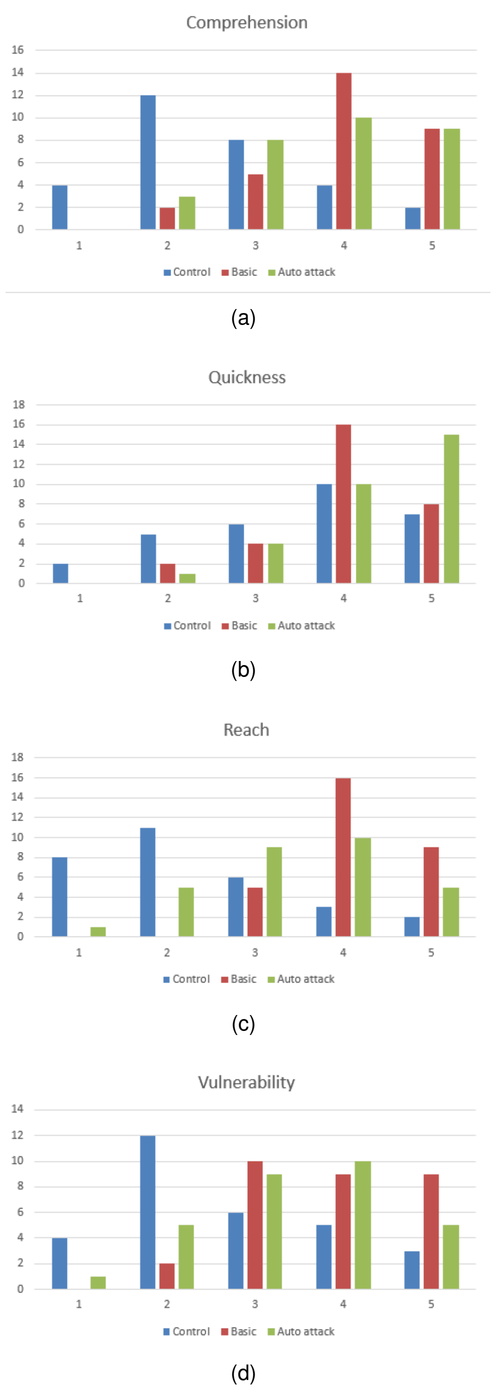
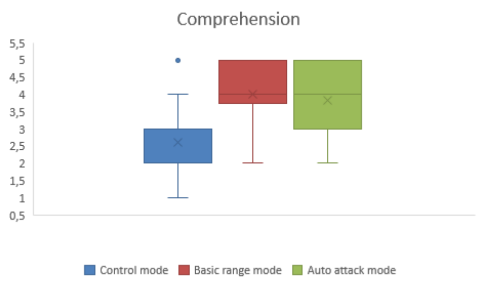
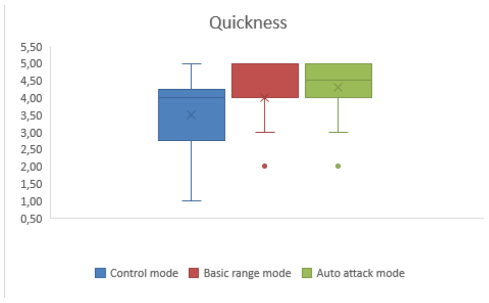


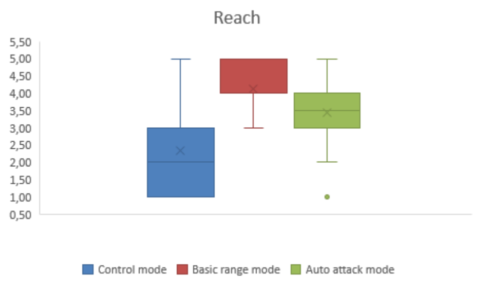
**Figure 5:** Bar graph showing the user distribution amongst different play modes.



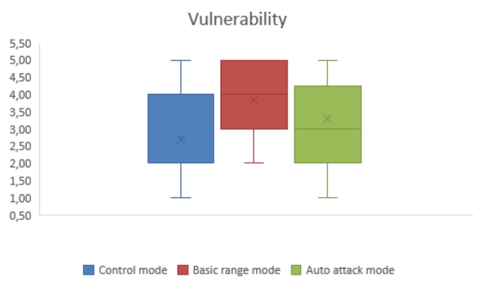
(a)



(b)



(c)



(d)

**Figure 6:** Box-plots depicting the evaluation users gave to the different play modes.

of the options support the same concept, but the execution in one of them was perceived as superior to the other mode.

The reach question, however, is substantially different (evident both by the quartiles as well as the test conducted previously) for the worse. This, coupled with the fact that the quickness question has virtually the same answers, can indicate that the tradeoff between clarity and speed was not worthwhile.

Still, this conclusion cannot be drawn with certainty, as the objective of the quickness mode was to give **experienced** players an advantage. The problem with that is threefold:

- The number of users that had experience with tactical games was negligible: only 16% (5 out of 30) of the users had prior experience with tactical games;
- Short playsessions: comprised of 20 minutes and one playthrough of each mode, didn't give enough time to the users to get used to the game (which in theory should be the point at which the auto attack advantages became more apparent, as that was how it was designed);
- The order of the different modes: due to how the tests were designed, some users **began** with this mode, which could lead to very different responses depending on the order the modes were played. The different orders were used to mitigate this difference, but analyzing each order could provide some helpful insight.

## 5. Conclusions

This project tried to do a study on what the best user interface/user experience (UI/UX) is for mobile devices, being that most current games on the genre are either pc ports or don't have the best user interaction. Having a better understanding of the subject is important because it would lead to better mobile games, a market commonly criticized for being composed mostly of time waster games without any real depth. However, this depth is often coupled with more user options and operations, leading to an increasing amount of information required on screen at a given time.

In order to do that, the approach taken was the development of three different options, each with a different objective, for users to try, later performing user tests, where each user would play the different options. The options were:

- Control mode (mode 1): trying to emulate how tactical games present information, with the attack range laid out after the movement range;
- Basic range mode (mode 2): displays the attack range in regards to the unit, not the maximum movement range. This tried to check if the users thought the information was clearer when presented like this or the usual way;
- Auto attack mode (mode 3): makes the units able to attack automatically if they can move in attack range of an enemy and hit it in the same turn. This tried to give experienced

users (ones who understood the game properly) a way to speed up their turns, requiring only 1 click to attack instead of 2 (very relevant in mobile, preventing misclicks).

Since every user played the 3 game modes instead of playing only one, and to prevent the results being skewed by the order in which users played the modes, three orders were selected and each users played one of those. The user tests had several different objectives:

1. Figure out if one alternative was clearly better than the others: this would give a good first idea of what features to look at as good practice;
2. Understand what made the alternatives good or bad: this is a more thorough approach than the first one, looking at several different parameters instead of just the overall rating of each mode;
3. Check if previous experience with tactical games influences how the users react to the different modes.

The first objective was somewhat achieved, with the control mode being the clear worst one. However, while the basic range mode was the preferred one by the majority of users, the difference between it and the auto attack mode, regarding the chosen metrics was small enough to warrant further analysis.

This analysis crossed over with the second objective, as the best way to distinguish between the alternatives is look at what makes them good and/or bad. All the alternatives, apart from being rated in terms of global preference, were also divided in categories: overall **comprehension**, **quickness** of the turns, understanding of the **reach** of units and understanding of **vulnerable** positions (spaces where a unit would be in range of attack from an enemy troop).

An early analysis of the different parameters showed no meaningful difference between the basic range and auto attack modes, except in the reach category, where the basic range was clearly better than the auto attack one. However, this conclusion cannot be drawn just from this fact, because the auto attack method was meant to give experienced users (experienced both in the game, with a good understanding of how the units worked and what their ranges were, as well as experienced with tactical games in general) an advantage.

These conclusions, however, don't mean that the auto attack mode is useless or harmful to the experience, rather meaning that it serves a different purpose than the basic range mode, and with some improvements can become even more powerful, discussed in section 5.1.

Regarding the third objective, there weren't enough experienced users to be able to perform statistically relevant tests, as we leave that conclusion to further testing.

### 5.1. Future work

The work developed in this project, even though serving as a decent ground for other projects to be built upon, still is in a very early stage, and could use some changes and improvements. Some of these, even if looking only small quality of life improvements, could mean major changes in the way users perceive the game (evidently subject to further testing).

One addition that should be done to the game as a whole is to add the option for users to click on enemy units and seeing their range (the way it would be presented depends on how the developed decides to implement range in the first place, but the option should exist nonetheless). This was suggested by some users but another, more important sign it is a significant feature is the fact that while playing, many users after moving their units, were attacked by their opponent, later saying they didn't have conscience that that was a vulnerable position.

Regarding the auto attack mode, one change that could be made (even though it wasn't suggested and there wasn't evidence during the tests that it impacted the experience negatively) is allowing the users to select where they want their units to move when attacking a target, unlike how it is currently done, where a position is automatically selected. This would defeat the purpose of this mode altogether, because it would then require the same number of clicks to perform the same actions, it is only mentioned purely from a theoretical point of view, as further testing could reveal some interesting conclusions.

Another change that could be done, and was even proposed by some users, would be to merge the basic range mode and the auto attack mode. This way, the range would be presented just like in the basic range mode, allowing for manual control of the attacks, while simultaneously giving the option to automatically attack any unit in movement+attack range.

### References

[1] Android's core app quality `http://developer.android.com/distribute/essentials/quality/core.html`, accessed at October 17, 2016.

[2] App store guidelines `https://developer.apple.com/app-store/`

review/guidelines/#terms-conditions, accessed at October 17, 2016.

[3] Average mobile app session length as of august 2014, by category (in minutes) http://www.statista.com/statistics/202485/average-ipad-app-session-length-by-app-categories/, accessed at October 17, 2016.

[4] Fire emblem villages system http://fireemblem.wikia.com/wiki/Village, accessed at October 17, 2016.

[5] Home is where the mobile game is: 96% play games in the house, 53% in bed http://techcrunch.com/2012/03/22/home-is-where-the-mobile-game-is-96-play-games-in-the-house-53-in-bed/, accessed at October 17, 2016.

[6] Turns, rounds and time-keeping systems in games http://en.wikipedia.org/wiki/Turns,_rounds_and_time-keeping_systems_in_games#Turn-based, accessed at October 17, 2016.

[7] Xcom morale system http://www.ufopaedia.org/index.php?title=Agents_Stats_%28Apocalypse%29#Morale.2FBravery, accessed at October 17, 2016.

[8] T. Rogers. Let's talk about touching: Making great touchscreen controls http://www.gamasutra.com/view/feature/187126/lets_talk_about_touching_making_.php, accessed at October 17, 2016.

[9] C. Stern. 12 ways to improve turn-based rpg combat systems http://sinisterdesign.net/12-ways-to-improve-turn-based-rpg-combat-systems/, accessed at October 17, 2016.

[10] C. Stern. 6 more ways to improve turn-based rpg combat systems http://sinisterdesign.net/6-more-ways-to-improve-turn-based-rpg-combat-systems/, accessed at October 17, 2016.