DA3 Group 5 - Handin 6

Jonas Madsen, 201708090   Frederik Nielsen, 201705351

December 7, 2018

No changes.

Game Description

Our games is based around controlling a ball using the tilt feature on the phone. This ball is to be tilted through a maze and it cannot touch the walls. Touching the walls is a game over for that level and forces a restart. Winning requires you to cross a line at the bottom of the maze and it allows you to progress to the next level. The higher you go in levels, the higher the difficulty. Obstacles are introduced that require you to interact with them or change up the way you play the game.

Application Goal

Our goal with the application is a rather simple one. We intend for our target users to have fun playing our game and hopefully also a relaxing experience. Of course, it may be a different experience in reality as some use games not to relax but to strengthen their cognitive abilities. In the end we want our users to enjoy the application we have made.

Benefits to the target users

Our application is intended for 10 year-old's to 20 year-old's. We envision them using it in their down-time, so when they're commuting, watching television or something akin. There are several benefits to these users, but they may be different from person to person. The younger part of the user group will see an increase in their visuospatial skills, which is dept perception, spatial navigation and the ability to estimate movement. This connection has been researched and published by Frontiers [1]. Games also makes the brain release dopamine, this is related to accomplishing something, for an example winning a level in the game. Dopamine makes you feel good and this is a positive for all our users, as is stated by an article from Psychology Today [2].

Usability trade-offs

Time to learn is the top priority based on the fact that our interface and design are not the main elements in the application. Our main focus is on the game and our users should be able to get into the game as fast as possible. Rate of errors by users is the second priority and it is connected by the argument used in time to learn. Our own experience is that applications that have a tendency to cause errors for users are quickly scrapped and a better one is found. Speed of performance is the third priority and closely followed by subjective satisfaction. Research by the Nielsen Norman Group shows a close connection between the two and concludes that, if people have an easier time at using the interface, they'll like it more [3]. Retention over time is the last priority. An article by Techcrunch show that users spend most of their time on 5 main apps and that they use these apps regularly [4]. These applications are decided by the other trade-offs more than the retention over time, so that's why it's our last priority. This is based on the former articles and own experience.

Scenarios

Scenario 1 - User wants to change his settings

John Doe is a student studying for an IT related degree. He is currently playing the game, but he is unsatisfied with the settings and wants to turn off the sound. He moves his finger to press the pause button. He clicks on settings and is interrupted by a phone call from his Mother. A few minutes pass and he finishes his call. He is now on his phone frontpage and searches for the application icon to reopen it. The app is continued where he left off and he ticks off the mute setting. He then clicks the back button and the resume playing button. John is now playing again with no sound.

Scenario 2 - User wants to play his first game while watching television

John Smith is a 14 year old watching TV. John opens the game to play while the ads are playing on the TV. He locates the big play button and clicks it. The app shows a tutorial but he skips it. He starts learning the basics, but the ads comes to an end. John decides to keep playing and finishes his first level. A message appears with a congratulation and he feels rewarded. He moves to click exit and continues watching television. New ads appear shortly and he wishes to continue with the app. He reopens it and upon clicking play, he can continue where he left off. - According to Accenture, 74 percentage of their consumers from age 14-17 use a Smartphone when watching TV [5].

Target Users

Our users will be using our application when they're "on the fly" e.g on the bus, waiting for someone, taking a break at school. Other times it will be used when they're multitasking at home or have downtime between dinner or TV. This means that in many cases they will only have limited time and the user should be able to exit at anytime, which should be taken into account. This is based upon statistics that shows where teens play mobile games [6].

Specific characteristics

A young audience tend to be less acceptable towards large amounts of text. This is for some because of poor reading abilities, but the main deal is that many teenagers prefer easy and effortless interaction with the interface when wanting to be entertained. The younger audience are more confident using advanced features and are not afraid to investigate unknown entities. This is due to most of them being expert users and using mobile features on a daily basis. This is stated by an article on ready4s [7]. A very young audience prefers flashy screens and pretty colours to keep their attention on the screen, but this is not something that an older audience wants as they have no problem keeping attention if they feel captivated. This is based from personal experience and talking with others. Another characteristic according to personal experience is that they tend to multitask and this means that switching between applications will be a common thing for our audience. It is also more common to use the mobile with only one hand compared to two hands. This is connected to multitasking and the users might switch off if they cannot use the application in their preferred way. Research done by a group from Michigan University [8] show that 5th-graders, 8th-graders and people from College prefer the video game genres of Diversion and Competition.

Impact on design

It is important that the interface the users firstly encounter, does not overwhelm the user with large chunks of text. This impacts the design towards a more graphical and nonverbal interface. That the user is able to make sense of by just looking at it. This could mean replacing most of the text with icons where they implicit know the meaning. Cogwheel for settings, speaker for sound and so on. We picked out our target users around 10-20 and designing the interface for any lower age group would mean that we lose the older part of the users. The more kid-friendly design is likely to scare them away, so we're going to be using a nice middle ground and keep the interface somewhere between. Most of our users are likely to be multitasking. This means that the users memory load is going to be limited, because they are unlikely to remember what they were doing, the last time they entered the application. This leads to a design that allows fast for pause/unpause, storing earlier progression, and one that is focused upon one-handed usage. On are more game design level. We have taken into account that the preferred genres are of diversion and competition. This is seen by diversifying our levels by introducing new obstacles and challenges. Each level is equipped with timers that encourages competition and connect with the benefits for our users.

Mobile Features

Feature 1 - Gyroscope/Tilt

The gyroscope will be our main feature for the game and to be more precise, we will be using it for tilt. Our users as stated above are often using applications on the go and by basing the game mechanics on tilt, they can play it with one hand, leaving the other hand free to use. They are expert users and to become one of the five regularly used apps, the application needs to catch their attention. This feature will allow for increasingly challenging gameplay as it can focus on both balancing skills and puzzle-solving skills.

Feature 2 - Touch interactions

The touch interactions will be essential to the interface, as it is used to navigate the different activities the application contains. It will also be used as a game mechanic where the user can click the screen to perform different actions in the game. This will be a function towards expert users for a more challenging gameplay and therefore will be introduced in later stages of the game.

Feature 3 - Network connectivity

This feature will be using network connectivity to show a global leaderboard. For this to work we will be using an API from Google, which makes it possible to keep this data online at all times. The leaderboard will compare data for how long a given user takes to complete a level. This will further challenge users to try harder and keep users engaged even when they have completed everything.

Task Analysis

We have divided our application in three high-level task actions. These are to play the game, to change the setting and to look up the leaderboards.

To perform the first high-level task, to play the game, the user has to perform the medium-level task that is choose an unlocked level in the level menu. This can further be broken down into the atomic actions, the user clicks the play button and clicks on a level that is unlocked.
Two other medium tasks are winning the level and losing the level. Winning a level splits up into the atomic actions to click on next button or click on quit button and thereafter click play and click a level. Losing the level splits into the atomic actions either to click restart or click quit.
The user decides to pause, which can be broken down to the atomic actions, where they click the pause button on top right corner or closing the application.

Next one is changing the settings. Changing settings can be broken down to these two medium level actions of either changing settings inside the game or changing settings outside the game. Changing it inside the game comes down to clicking the pause button and then clicking the settings button. You can setup medium-level actions to be changing the sound, changing the music and so on, but in the settings menu they are represented with only buttons. These tasks can therefore be considered atomic actions because they only requires the user to click on a button or using a slider.
Changing the settings outside the game can be done by clicking on the settings button from main menu. Considering the act of changing sound it will also be an atomic action which can be seen from the main menu. Here it is a matter of clicking the sound shortcut to turn it on/off.

Last one is looking up the leaderboard. It can be split up in twp different medium-level actions. They can look up the best time on a level or they can look up own/friends best time. This can be further broken down into the atomic actions, where they click the leaderboard icon and click the level of which they want to see the leaderboard, then they use a slider so it says either "all" or "social".

Paper mock-ups

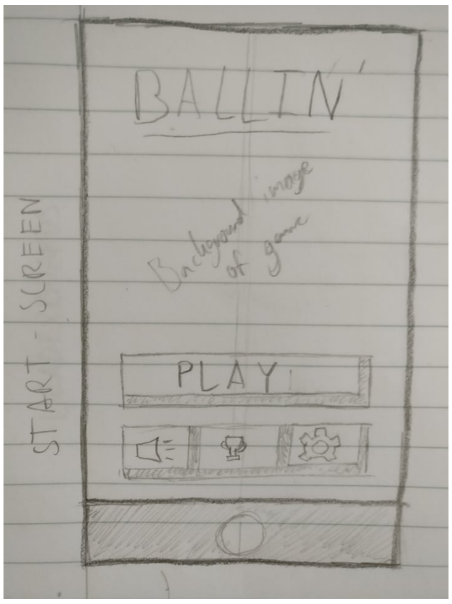


Figure 1: Main Menu

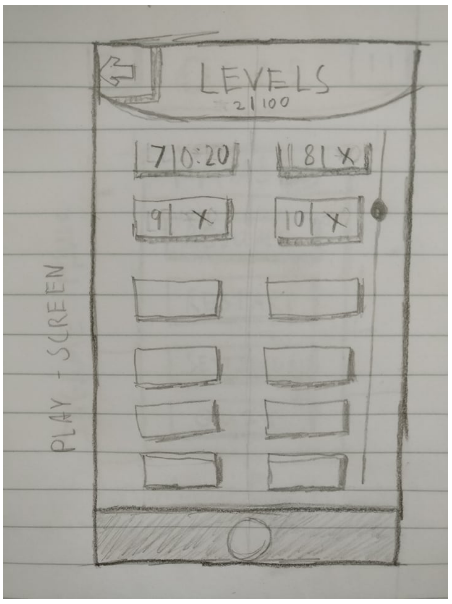


Figure 2: Level Menu

Our first figure 1 shows the main menu of our application. This is the first thing you see upon opening our app. A play button is displayed in the somewhat middle and is the biggest part of the interface, so that users can't miss click it. The three buttons that are under the play button starting from left, a shortcut to turn sound on or off, a shortcut to leaderboards and a shortcut to settings. A video related to the game will play in the background. Figure 2 is the level menu and the next screen that arrives upon clicking "PLAY" on the main menu. This menu has a shortcut in the top left to return to the previous menu. At the top is also displayed the currently completed levels out of total levels. Each level features its own button that shows completed in what time or that it hasn't been completed. This menu is scrollable, it starts from the top and scrolls down, this is akin to webpages or normal reading.

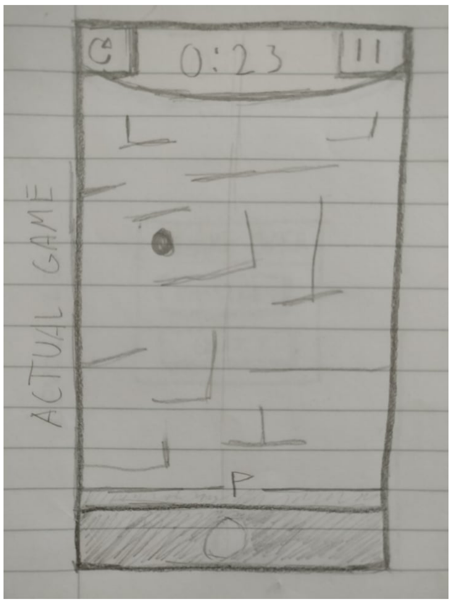


Figure 3: Ingame

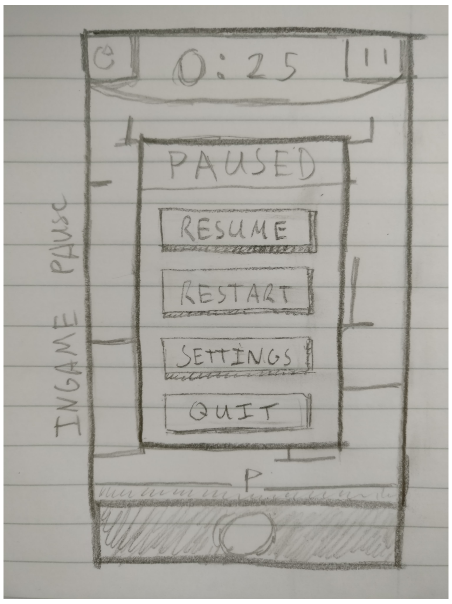


Figure 4: Ingame Pause

Figure 3 displays actual gameplay of the game. This appears when users have selected a level from the level menu in figure 2. On this screen we have a shortcut to restart the game for the more competitive users, this is shown in the top-left corner. They are placed high and small to avoid miss-clicks and not take too much space of the important game play. In the middle there's a counter/timer that shows you the current time spent on this level. In the other corner on the right side, we have the shortcut to pause the game. This button leads to figure 4. Resume button and restart button does as they imply. They are the same size currently, but this can cause some miss clicks, further testing will reveal this. Settings leads to figure 5. Quit leads to figure 1, the main menu.

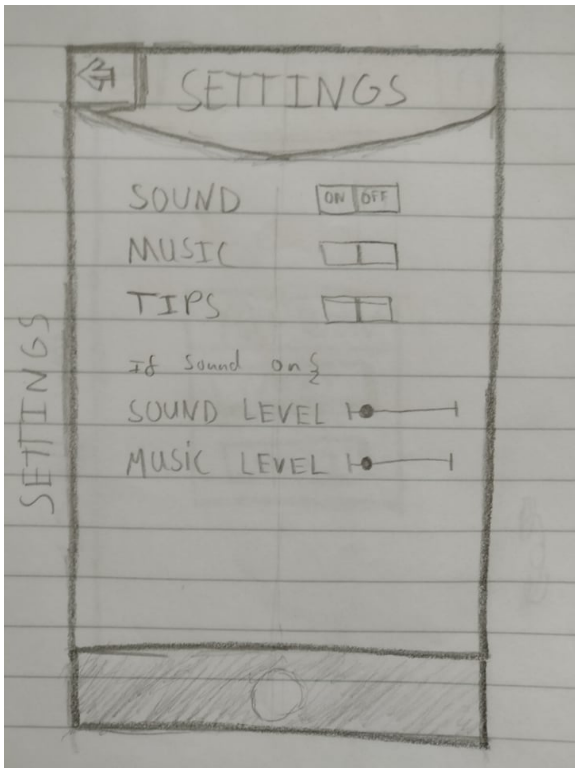


Figure 5: Settings

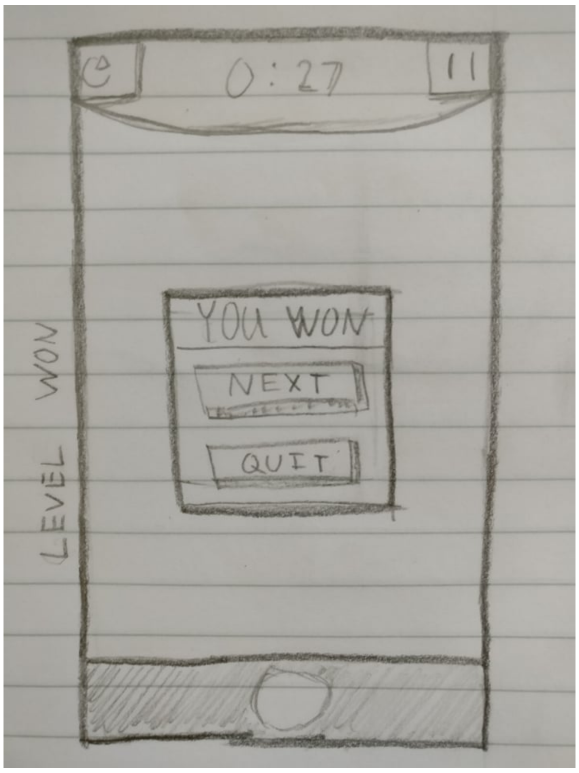


Figure 6: Victory

Figure 5 is the settings menu and it was reached through figure 4 as explained above or figure 1 and cogwheel shortcut. It is badly drawn but the button are tapped and it switches from on to off and the other way. When Sound and Music are on a new set of sliders will appear at the bottom. These sliders adjust the volume of sound and music. This is consistent through other applications and should avoid any mistakes. The shortcut in top left returns you to the previous menu, which would be figure 4 or figure 1. The next screen is figure 6 and this appears when the ball crossed the finish line, also known as victory/winning the level. Users can here use "NEXT" to get quickly to the next level or "QUIT" to figure 1, the main menu.

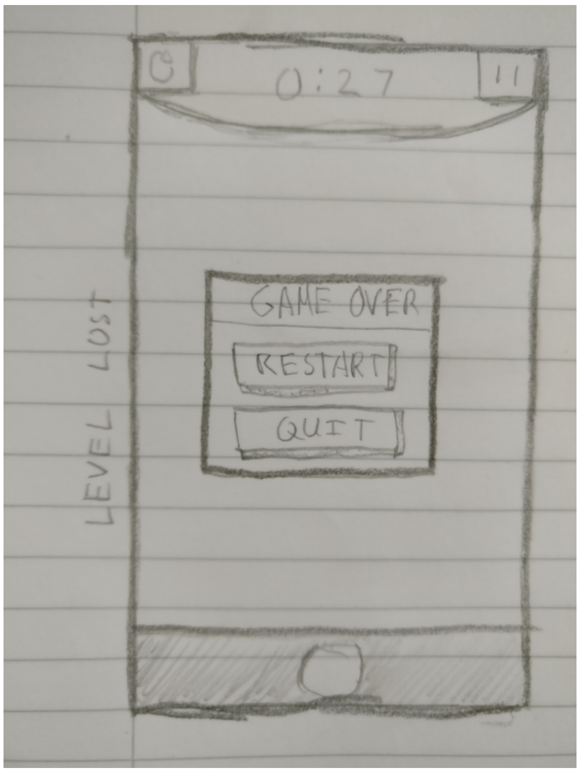


Figure 7: Defeat

The figure above, which is figure 7, shows the defeat/game over screen that appears when you touch a wall or fail the level in the figure 3. It is possible to restart the level by clicking "RESTART" or to "QUIT" to figure 1, the main menu. This is consistent with the figure 6.

There is also the Leaderboard screen which you get to by clicking the trophy shortcut on figure 1. This leaderboard is provided by Google's API, so it is already predesigned and made, which is why we haven't included it here.

Design rationale

Visibility is used at figure 1 which shows the main menu, which has a button to switch on/off sounds. This is a commonly used functionality and by making it visible for the users they will easily be able to use it. The "PLAY" button in figure 1 is also bigger than the other buttons, which also elaborates that it is the main function in the main menu. We don't really have any hidden away features for the expert users, so everything is highly visible. The buttons have a raised 3D effect, which implies that they are press-able. This gives **affordance**. In a way, you could say that including a tutorial of game mechanics is also affordance, but that leans more towards game design. The interface will also be adding **constraints**. Every time a "QUIT" button appears it will be coloured red, for example in figure 6 and figure 4. This will let users know that this action is dangerous. In figure 2 a logical constrain is used when users have to scroll down for more levels. The users can see that the numbers goes higher the more you go down, which gives the user the ability to deduce that scrolling down gives higher levels.

The application adheres to **strive for consistency**. This is done by having similar layout throughout the application. Figure 5 and figure 2 both have the same back button which both does the same and takes the user to the front menu. This button is also included in the leaderboards menu provided by the API, but it differs a little by design. Buttons are also done consistently throughout the application. In figure 5 it can also be seen that the three turn on/off buttons are similar and also when adjusting sound or music it works the same way. The interface uses **dialogs to yield closure**, which can be seen on Figure 6 and Figure 7. These are the screens shown when the game comes to a closure, either by winning or losing. When the user gets the one of these dialogs, they are done and a new action has to be taken to continue. **Permit easy reversal of actions**, the user can enter menus to explore, as they are given the option to quickly return to the last visited menu. This is shown by the back button on figure 5 and figure 2. Figure 5 also shows the tips setting which is added to seek **universal usability**. This setting will give explanations for the novice user, but the expert users can still choose to turn it off. There are also added a shortcut to restart the current game instantly, as seen in top-left on figure 3. This is geared towards the more competitive/experienced users, but the more novice users can instead pause take a moment to consider and then restart as figure 4 shows.

Reflection on Design Changes

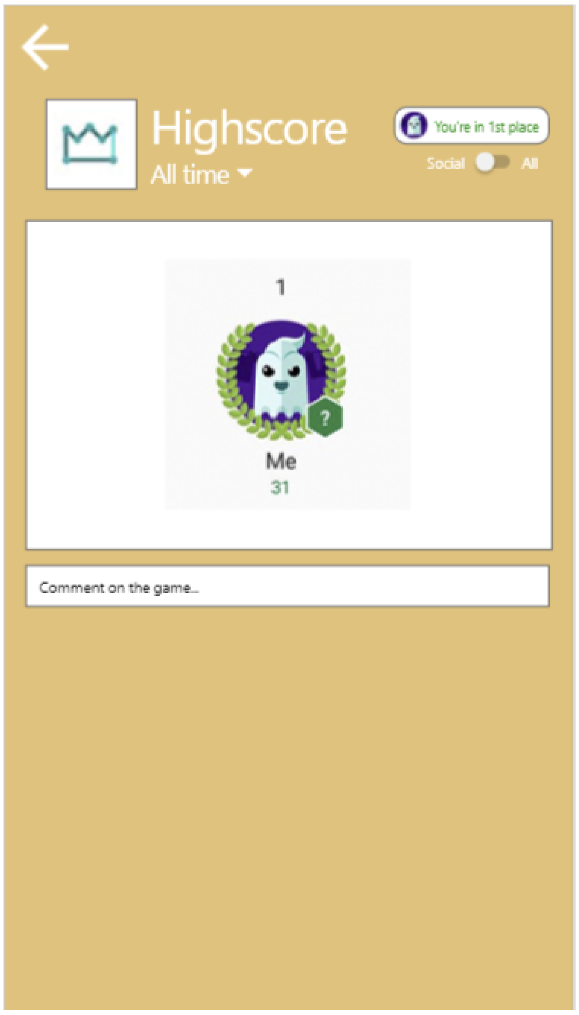


Figure 8: Leaderboard Social

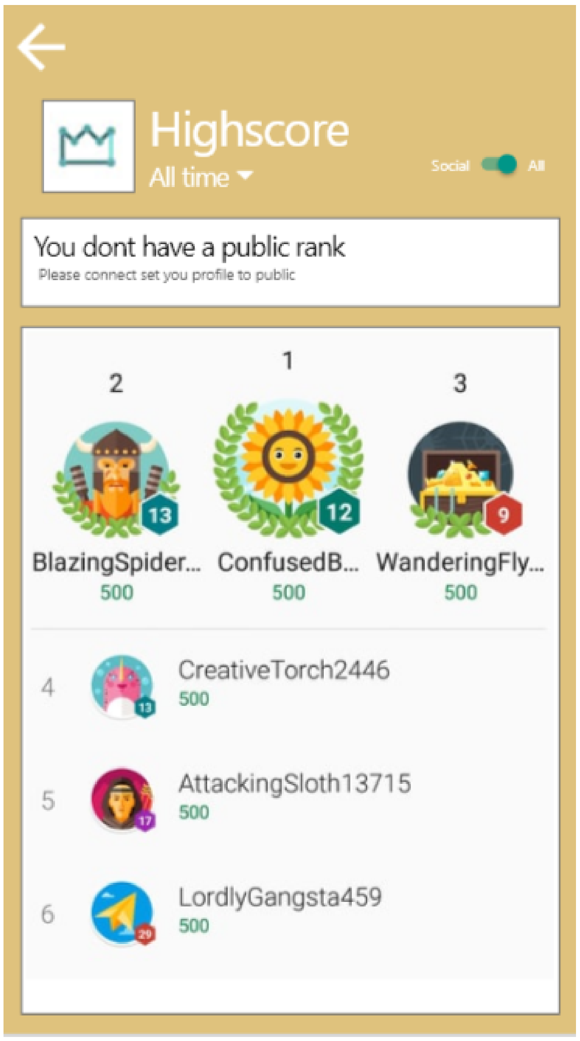


Figure 9: Leaderboard All

The leaderboard is provided by Google API and to stay consistent with other applications, we stayed true to their design. Figure 8 shows the social option, which shows your point score among your friends. Figure 9 shows the all option, which is the leaderboard over everyone playing the game. This is just a prototype, so the scores and names were borrowed from a different game. Google is favouring flat design, so we had to turn down some affordability to gain consistency.

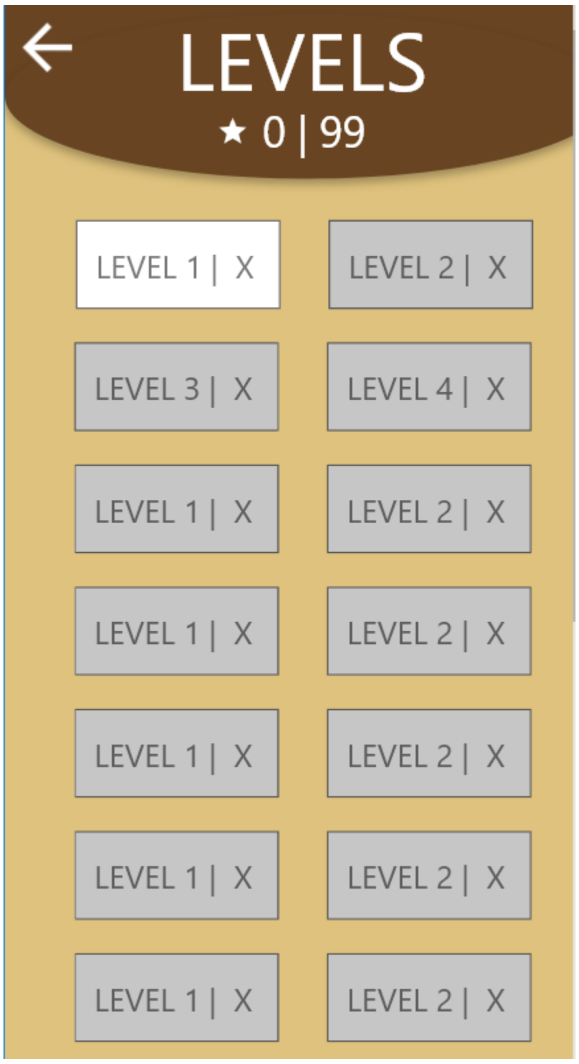


Figure 10: Levels

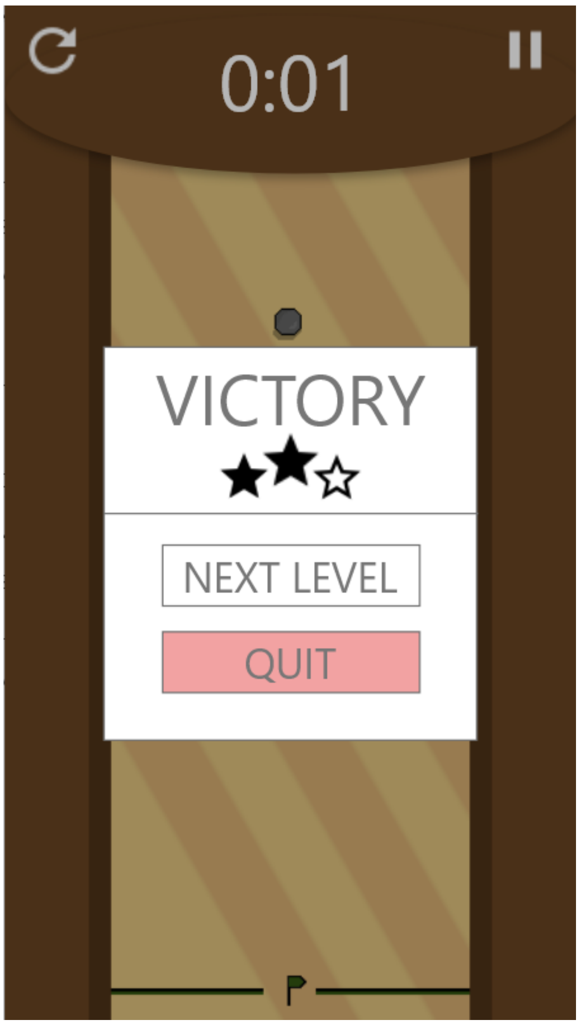


Figure 11: Victory

Figure 10 shows our new version of the selection of Levels menu. The changes from the paper mock-up are that locked levels are now greyed out. The return button in top left has moved to flat design, per the same reason out for the leaderboard, consistency through other apps. On a game design level we have moved over to a star based reward system instead of time. The amount of stars earned are now shown under Levels. This change has also been introduced to our Victory and Defeat screens, as shown by figure 11. Here the player has earned two stars for his performance in this level. The "QUIT" button has also been coloured red to show that it is a destructive action.

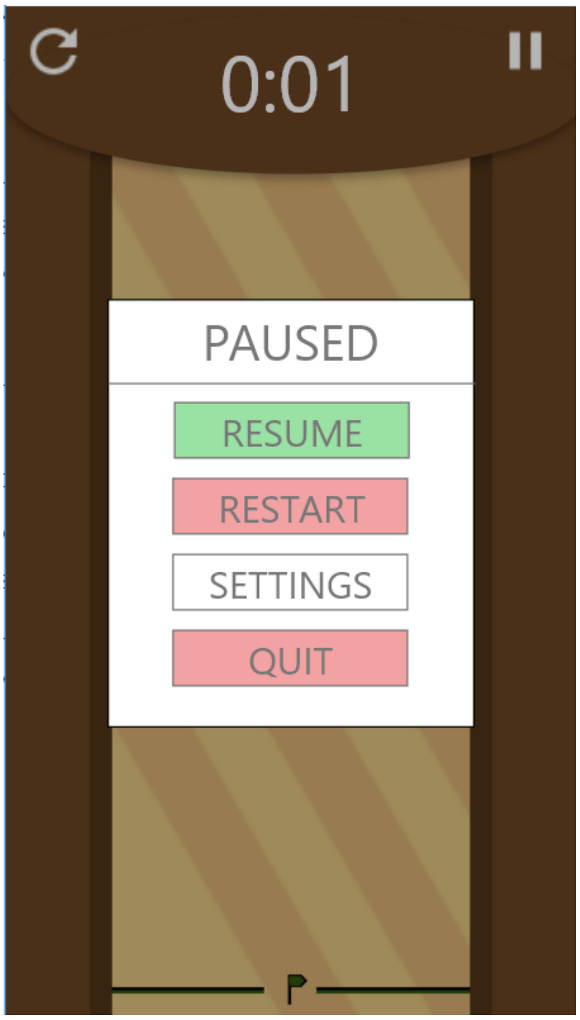


Figure 12: Ingame Pause

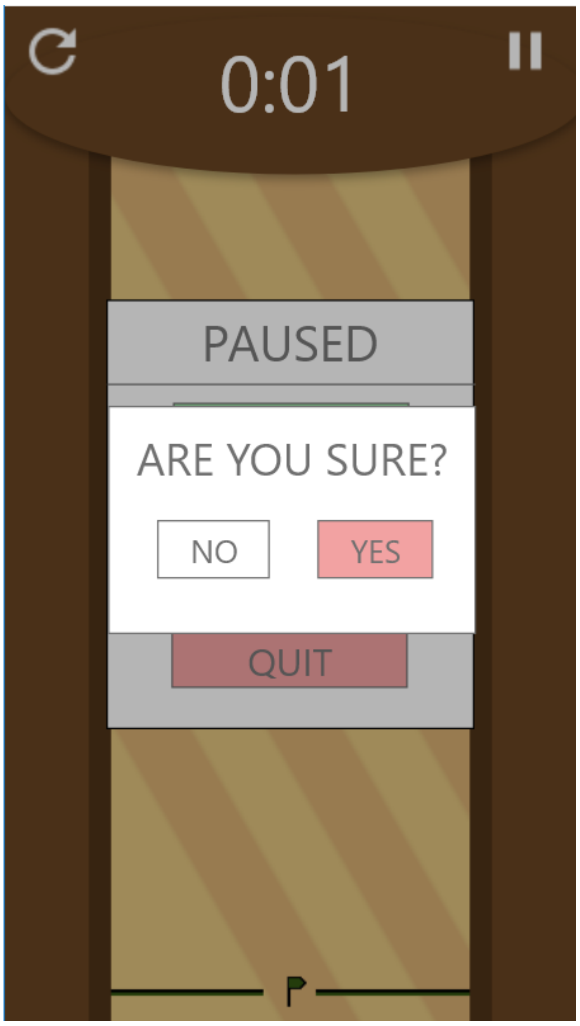


Figure 13: Ingame Pause Restart-Confirmation

Figure 12 shows the ingame pause menu. Major difference here is that we have added colour to indicate destructive and beneficial actions, with red and green accordingly. The restart button now leads to the figure 13, a confirmation menu. This was added to avoid miss clicks on "RESUME" and "RESTART" as there's a huge difference on the two. The confirmation menu makes sure that the user meant to press "RESTART" and follows the consistency of the right button allowing you to continue your action and the left dismisses. This does add an additional button press to restart, but users should mainly use the shortcut restart in the op left of ingame.

Copy of Expert Reviews

Heuristic Evaluation - Group 5

Rasmus Elias Thisted, 201708242

October 2018

Shneiderman's 8 Golden Rules

Consistency

The app was generally very consistent, with a consistent color scheme of recurring brown colors and common Android logos. The only place I could find that the app was not consistent with this theme, was in the "Highscore" view, where the title of "Highscore" was white instead of brown, and not a centered header like in the "Settings" view, but it still looks fine. I also liked that the game was consistent with other Google apps in its ability to comment on games in the Highscore view.

Universal Usability

The app is very easy to use with simple common logos, and therefore I think anyone can use the app. The app hits a nice balance with it's design and logos (like the ones in the Highscore view, where you have a little cartoon character on top of your score) of neither being too childish or too mature, and thereby hitting the sweet spot of a broad target audience, as was the intention. I had no negative concerns about universal usability.

Informative feedback

I liked that the app shows you how many stars you earned after a game had ended. A nice feature would also be to have a "New record!" statement be there, when a previous record was broken, if this was not already the intention when actually developing the app.

Dialogs to Yield Closure

The app is very easy to navigate, as the amount of views are kept to a minimum. It is very easy to see how far in your progress with the levels you are, and how many stars you have in total out of all possible stars in the Levels view. I had no negative concerns about this.

Error Prevention

I could not imagine a lot of things going wrong when using the app. The quit button is red and clearly emphasized, so anyone will know that it will result in an unchangeable action. However, maybe a pop-up saying something like "Are you sure you want to quit?" could be useful to prevent errors.

Easy Reversal of Action

The app doesn't really have a lot of actions that could need to be reversed. You can easily restart a level with a single click, which I liked, and easily both pause and resume the game. The only thing I could think of that a user might want to reverse, would be to be able to reset all their current progress, so that all their stars and levels would be reset, if they wish the play the game anew, or maybe a parent wants to lend their phone to their child to play the game. It's not a crucial feature, but maybe could be a nice option in the "Settings" menu.

Keeping Users in Control

The app lets you change sound and music separately, which I think is a useful feature. It also lets you change the "Highscore" time scope, if you want to see the weekly scores, instead of "All time" scores and the ability to change the view of highscores between "Social" and "All", which I liked. One thing that I maybe think could be done a little easier and more clearly is to go back to the "Levels" view from a game in progress. It is not so obvious that you have to pause the game and then click "Quit" in order to do so. However, it is not crucial, as I think most users would find out eventually, as there are only 2 buttons to click. But maybe adding a "Change level" button on the "DEFEAT"/"VICTORY" pop-up would be nice, as I could imagine it's often after either a defeat or a victory that a user might want to change level.

Reducing Short-Term Memory Load

The only real things that I think the app needs to keep track of is high scores and the best personal score of each level, which are shown in the Levels and Highscore views. One thing that could be nice however, would be to also have the "Previous Best" information shown inside a level, so a user can quickly see what to beat, and doesn't have to remember it.

Other issues

There were also some minor design choices that grabbed my attention. Firstly, I think it looked a little off to have the "Restart" button in the pause menu be red, when the "Quit" button was also red. I think it took a bit of the emphasis away from the red "Quit" button, and I usually only associate red button with quit buttons. However, "Restart" is also a critical action, so to prevent errors, you could include a pop-up, which says something like "Are you sure you want to restart?", when "Restart" is clicked. Secondly, in the Highscore view, it says "Please connect set you profile to public", which I assume is a mistake/typo. Other than that, the design is really good looking.

Heuristic Evaluation of Ballin'

Computer Science, University of Aarhus
8200 Århus N, Denmark
Anne Dorte Rafn Spangsberg, 201406920, au518510

October 14, 2018

First of all, it was generally a fine user experience and a nice looking app.

Strive for consistency

Regarding consistency the app seemed really fine, for instance the pop up boxes was consistent in width, fond and placement of options. The options which where coloured (quit for instance), had the same colour every time. There was a nice consistent colour scheme.

There was also a repeating layout, with a brown half circle in the top, which I thought was nice. This was however not present in the high score page.

The use of icons where not totally consistent. For instance there was a restart icon in the top left corner of the game page, but in the pop up it was written. There is a sound icon on the start page, but in the settings menu it is only written. A pause icon is used but not a play icon.

I think an increased use of icons could be a strength, especially when your target users are people down to the age of 10. This could maybe also be obtained by an increased use of coloured options, maybe the option of going to the next level (when you win) could have a special colour, that indicated that this is what you want to do.

Seek universal usability.

The game is easy to start for a new user, and it does not take many clicks for an expert to get the game started either.

To save a single interaction, one could consider making the play button start the highest available level as a default, and then have 'levels' as an extra button on the start page, so that the user can choose a different level.

Offer informative feedback.

The app has many strengths regarding giving feedback. For instance when you turn of the sound in the start page or in the settings, or when you win or lose a game.

Design dialogs to yield closure.

It might be a strength of the app, if there the first time it is turned on, is show how to play the game, but this does not need to appear every time a game is started.

The app might also benefit from a notice about how the point system works.

Prevent errors.

The red quit button works very fine do to preventing closing a game by mistake.

The restart button however is placed where there would often be a going back button, so a user might press that by mistake.

Further errors could maybe be prevented by the use of pop ups, that asks you, whether you are sure you want to restart or quit the game.

Permit easy reversal of actions.

Generally it is easy to enter or exit a game, which is nice. However if the user wants to play a specific level, but by mistake enters the wrong one, there is no direct option for going back. The user has to exit the started game and go into a new to try to choose the right level. A ordinary back button could be nice here.

Keep users in control.

This is very fine. The user decides what level he/she want to play (among the ones that are available), and are able to turn of the sound.

Reduce short term memory load

Generally it worked pretty fine. There wasn't much to remember.

Maybe if a user is looking for a specific level, it could be hard to find, because the user needs to remember it solely from the level number, and there is no direct way to go back to the level choosing site from the game site. One could consider making a back button or maybe naming the levels in some clever way.

Other issues or questions

When you win a game, there is a quit button that are red, but the next level button is not green or any other inviting colour. I think it could be a strength of the app, to use colour, to tell the user which action is desirable.

If the player is to be able to upload a high score, he/she will probably need to be logged in? A login/logout option would then permit multiple or changing players. But it might help to keep users in control, that they don't have to log in, before they want to upload a high score.

I think there is a small scrolling error in the levels page, the header scrolls away but the brown half circle does not.

In settings, when you turn of the sound, the scale for sound disappears. I think it could be a strength that it didn't disappear entirely, maybe just get greyed out. Personally I like when I can set the sound volume, when a app is silenced, so I can turn the sound on with only a little volume.

DA3_G05

1. Strive for Consistency

The app is consistent with actions, such as going back to the previous view and buttons in the "title" area of the app (top of the screen).

However, the buttons are not consistent within the app, the play button is different from the leaderboard/settings/etc. buttons just below it, while the buttons in the pause menu follow a 3rd style. The solution is to be consistent with button design and colors across menus/screens.

2. Seek universal Usability

There is only one way to do things, and the app offers no shortcuts for advanced users. This can be improved by adding shortcuts/different ways to do the same things for advanced users.

The game tells the user how to interact with the game and play it (assuming similar guidance will be used in the actual working game).

3. Offer informative feedback

Dialogs provide good feedback, when pressing restart from the pause menu the app will show a dialog asking the user to confirm or cancel this action. Furthermore, the app also provides feedback for the user when the loose or complete a level.

4. Design dialogs to yield closure

The one dialog used in the app (the confirm restart dialog) provides good closure, by clearly explaining what the dialog is about and what the individual actions do. Though maybe it could be improved with some extra text, telling the user that this action is not reversible?

5. Prevent Errors

The app prevents errors by showing dangerous actions and red, and asking for confirmations when the user attempts to perform a dangerous action and clearly indicates when an action is dangerous.

6. Permit easy reversal of actions

The app permits easy "reversal", in the sense that it prompts the user for dangerous actions, asking them to confirm the action, allowing the to "reverse" the action if they invoked it by mistake.

7. Keep users in control

Overall the user are kept well in control, fx. on the main menu where all actions are easily accessible and indicates that they are indeed actions that can be performed.

However, some actions are hidden away by menus (the pause menu), that could instead be shown to the user directly next to the icons for the pause menu itself. This would better keep the user in control by allowing them to access the action without going through the menu.

8. Reduce short-term memory load

There is no direct information the user needs to remember between views or actions, but the app provides no indication of where the user is. This could be fixed by fx. showing what the current level the user is playing is, or breadcrumbs/path to the current view/screen the user is at.

Heuristic Evaluation of Group 5

Mathias Bendtsen
October 2018

Strive for consistency

The app is generally consistent with colour scheme and with the mapping of buttons of similar actions. the games colour is in general, brown and white, menus is white. While stop playing is made red, and continue playing is green as the 2 outliers.

Another part where the app is not completely consistent with its design is in the highscore window & game. since in the “settings” and “levels” windows both have headers where the game and highscore doesn't have.
On the other hand in a lot of online gaming apps the highscore window is taken from google highscores, which it also looks like here. But in the game a title/header with which level you are at would be better designed compared to only the timer.

Seek universal usability

The generally large buttons helps to click more easily and in such makes it more usable for a larger group of people. Furthermore the icons makes it rather easy to understand what they do, as they follow a rather standard design, Eg. the small speaker as a mute button.

Informative Feedback

After completing a level, the next function which brings you to the next level, yields close to no feedback on screen transition. that added with no titles in the top makes it harder to see if you have gone to the next level.

Design dialogs to yield closure

The only dialog in this app is after you have pressed the quit button, which is in the game window. This dialog is rather simple and the only way it could be easier to understand would be to say “are you sure you want to quit”. which might be a to large amount of text for such a simple dialog. therefor not much to say here.

Prevent errors

Since the game looks to have a ranking on how you complete the levels ill have to consider errors in games to. which have been solved by the quick reset button.
Since there is only 5 changes in total, there arent alot of room for error in the settings part, and 3 of them is toggles which also help preventing them.

Permit easy reversal of actions

As said in “prevent errors” the quick reset button helps in case of errors in game. furthermore the pause button to either change setting or leave the level also makes it fast to go back to the level selection.

Keep users in control

The large overview over the different levels, makes it easy for the user to chose their desired level and in general the app doesn't really do anything without input from the user.

Reduce short-term memory load

The few amount of screens good makes it rather easy to learn while also quick to jump back into. might be able to increase with “completed” level marks. depends on the limitations of going to next level in the game. so this is more a design on the game rather than the interface itself.

As said in the consistency part, the missing header for levels makes it rather hard to know which level you are playing at the moment, so if you have jumped out, then you might have some problems remembering which level you are trying to complete.

Assignment 3b: Heuristic Evaluation- Jonas og Frederik

Strive for consistency:

- **Strengths:** There is good consistency between the layouts on the levels and setting page (the brown round circle thing at the top). The functionality and language is all consistent which is good.
- **Weaknesses:** The leaderboard page isn't consistent with the above mentioned design (if it looks stupid just ignore this).

Seek Universal usability:

- **Strengths:** Good colors makes the screen easy to scan. There isn't much text which makes it good for children/adults who have a hard time reading, so it fits well with your target users.
- **Weaknesses:** Even though the app layout and buttons are very intuitive, the first time the app is being used it might lack an explanation for the buttons, e.g. the leaderboard button.

Offer informative feedback:

- **Strengths:** The sliders offer very good feedback on the action you just took. Same as the sound-button in the main menu. The pause is also very good at offering feedback.
- **Weaknesses:** The restart button doesn't tell you that you have restarted the game(this would probably be clear when the game is implemented and you can see it begins at the top again, so maybe just ignore this.)
    - There are no dialogbox when you try to choose a level which isn't unlocked, or no feel of you having pressed the button.

Design dialogs to yield closure:

- **Strengths:** it is good that win gives you a menu to select whether you would like to go to a new level or quit - the same goes for the defeat popup. Same with the play button, that it takes you to a level menu and no just the game directly.
- **Weaknesses:**

Prevent errors:

- **Strengths:** When you have pressed pause in the game, and then press restart it is made sure through a dialogbox that you really want to restart.
- **Weaknesses:** If you press quit or settings, it doesn't make this check, even though it probably would have the same impact on the game.

Permit easy reversal of actions:

- **Strengths:** There's always a return button, or a way to quit the page you are in. It is very easy to get back to the start-menu.
- **weaknesses**

Keep users in control:

- **Strengths:** Great that it is possible to change the soundlevels. All the buttons does what you would expect.
- **Weaknesses:** There aren’t a lot of possibility to show that users have gained experience in using your app.

Reduce short-term memory load:

- **Strengths:** There is nothing to remember from screen to screen.
- **weaknesses**

Expert Review Analysis

Strive for consistency

In general the expert reviewers liked that the overall theme was consistent, with a brownish colour scheme and the brown round circle in the top of most activities. Actions like the back button was also kept consistent which some reviewers commented on. This was also our intention, so the layout was consistent.

However the buttons was not all consistent with the style they had and this is seen on figure 10, where level buttons are flat buttons and not 3d buttons like in the main menu, this could mean that the users might have difficulty learning what is a button. This issue is taken into account by making all rectangle buttons 3D to be consistent. This change can be seen at Figure 15, where you see 3D buttons.

Some expert reviewers addressed that the leaderboard activity is not consistent with the other designs. This is true but adding a header like in the other activities will mean that consistency across other game applications is lost, which can mean more time to learn and users might also like Google's layout more, so impact on layout might not get better for all users. therefore it is not changed.

It was suggested that icons could be consistently used instead of textfields. For example the restart button was an icon in the ingame activity but was written in the pause menu. Changing the textfield to an icon would mean that the user that is a beginner finds these icons hard to understand and it is more of a expert user thing. That is also why we use the icons mostly as shortcuts for expert users. Therefore it is kept the same to keep icons as a fast way to do something and textfields for places where the user needs to be sure what this functionality does.

Adding more coloured options was suggested as a strength to our target users. Adding a colour to the next level button (when you win) could help indicating that this button is what they should choose. This has been done, which can be seen on Figure 16, where the next level now is a gold-ish button.

Seek Universal usability

The expert reviewers is positive about the layout with common icons and big buttons, which helps hitting a more broad target audience. This was a deliberate decision, where we kept our young target users in mind when making the layout.

The balance between expert users and new users is the only change suggested, where functionality like making the user enter the highest level by clicking the play button or adding new functionalities for expert users is suggested. This is not done yet, because it is currently unknown which features expert users might want to have added and adding another button to save one click is just not worth it in our opinion, when mostly users would want to go to the level menu anyways. We want to reduce clicks required to perform actions, but also an healthy balance between cluttering the interface and reducing these clicks.

Informative feedback

Some positive comments on informative feedback is that the restart button gives a confirmation dialog and when you lose or win a level you get a dialog providing feedback that this is achieved. This was intentionally done to keep the user in control after every transition in the game.

Things like the mute/unmute sound icon in the main menu that gives feedback by switching the icon and stars gives feedback on how well you did. This was positively commented by the expert reviewers and only that a ”new record!” statement could be added was suggested. This suggesting is not developed into our game, because it might get underwhelming seeing this feature every time you complete a level. Nonetheless this might be reworked into a later version of the application.

Design dialogs to yield closure

Making more dialogs to show how to play or how point system work is suggested. This is certainly a possibility and something we will add later, where aspects of the application is difficult to understand, but at this stage there has not been any issues with learnability, because there is only simple features in the prototype. At a later stage these hints/tips will be implemented to help newcomers.

The already used restart dialog was given positive feedback because of good closure and simple layout. Our intention was to make sure this buttons feature is not used by mistake, because this is an irreversible action where other actions can easily be reversed.

Prevent errors

The expert reviewers were positive about the fact that the colours make the user more aware of which buttons are dangerous and this prevents errors by the user. Also the expert reviewers empathize that there is not a lot of stuff that could go wrong. This was also the intention, because the use of colour can greatly improve our target users perception of the application.

Although one commented that the restart button was placed, where he expected a back button. This is to some extend an issue but in our opinion the dialog box and the red colour makes sure the user only uses it when needed. This is why we kept our normal pause menu. Attempting to go back and clicking the restart button instead is no harmful action and the user can then adapt without making errors.

Permit easy reversal of actions

The expert reviewers agreed that it was easy to reverse actions wherever you might be in the application. The idea was that the pause menu with resume and quit option will help reversability. Users can always use a button to reverse their actions, which in most cases can be done in one click.

It was suggested that we added a back button to level menu when in game. This is a great idea but in our opinion it has the cost of using up space without an actual purpose for now. This might be a feature in the actual application if it comes up as a problem for users.

Keep users in control

It was commented that in general the application keeps the user in control by only doing something when there is some kind of input from the user. This is further improved by having settings to change the users experience to what they want. This has a positive effect on our target users because they will not get frustrated by some anomaly that can not be changed, when using the application.

One expert reviewer had the idea to give more user control by making more options visible when in game, instead of having them when paused. In some cases this might be a good idea, but with our application, there has to be minimal space used towards anything else but the game, when playing. Else the target users will not be able to play the game. Another option could be to make a option to hide/unhide these options, but we do not see any reason to keep options available when the game is running, where the user is focused on the game.

Reduce short-term memory load

Overall the expert reviewers had no issue with short-term memory load, which was because there was no information that needed to be stored from each activity. This also follows that we wanted a more youth friendly application, with less text.

There has been a change to the level menu concerning the star system, because expert reviewers suggested that the stars gained on a level should be showed at the level menu. This was also our intention

but what not visible in the last stage of the prototype. Now at Figure 15 it has been added to show what it looks like when a user has completed a level.
There was a problem with remembering which level you have entered, which one of the expert reviewers pointed out. This had impact that users might need to quit game and enter again just to see which level they played. This is fixed at Figure 14, where the prototype now has a current level text in the header, when playing the game.

Other Issues

Sound level disappears when turning it off, which an expert reviewer gave the idea to change into just grey when off instead. This change can be seen at Figure 17, where sound is off but the slider is still visible.

Summary of and Reflection on Design Changes After Expert Reviews

Summary

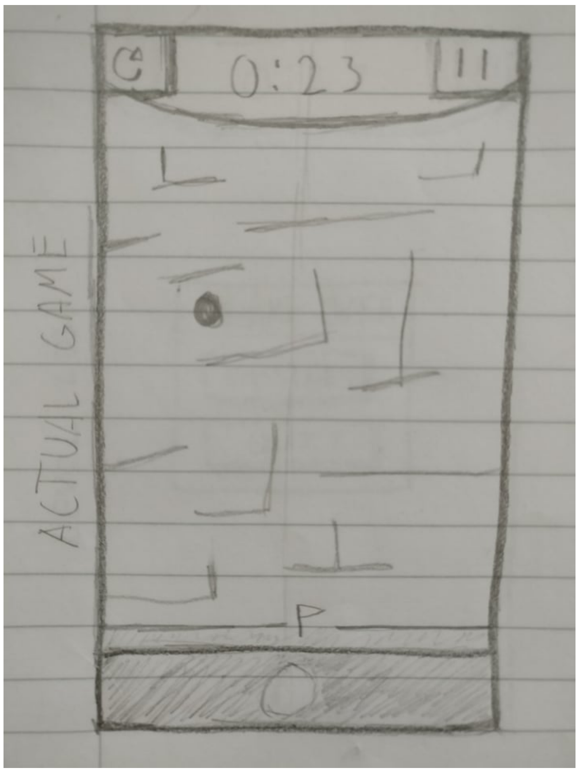


Figure 3: Before - Ingame

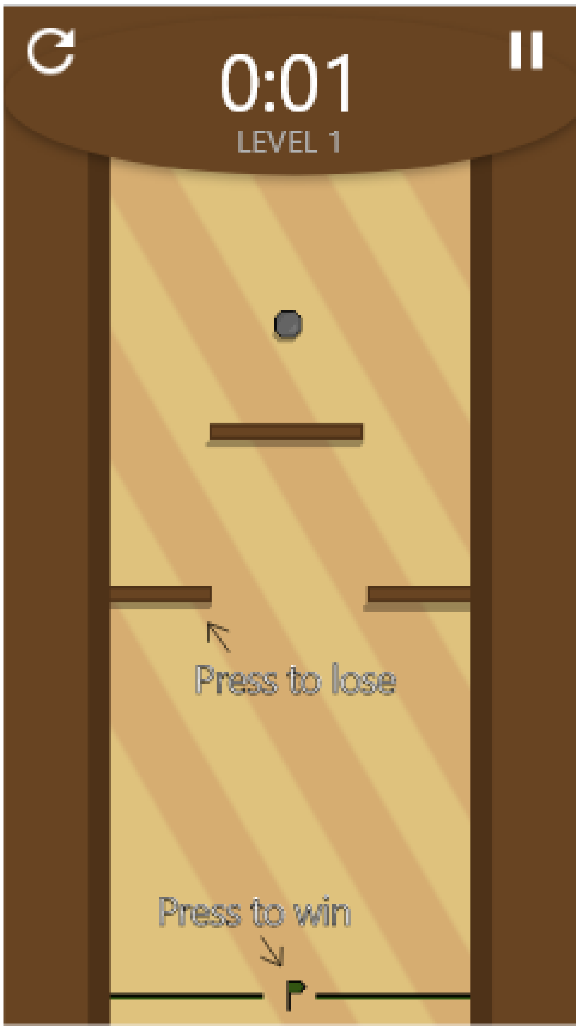


Figure 14: After - Ingame

Figure 3 shows the before image of the ingame layout. This layout was not satisfying according to short-term memory, so to help our users we changed up the top header. It now includes a current level display. As seen under the timer in Figure 14.

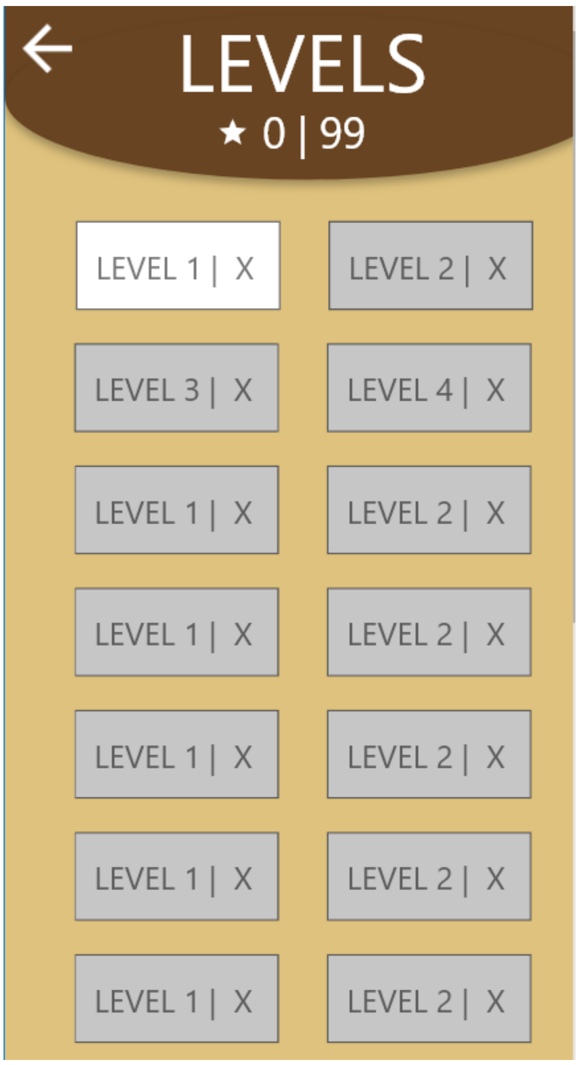


Figure 10: Before - Levels

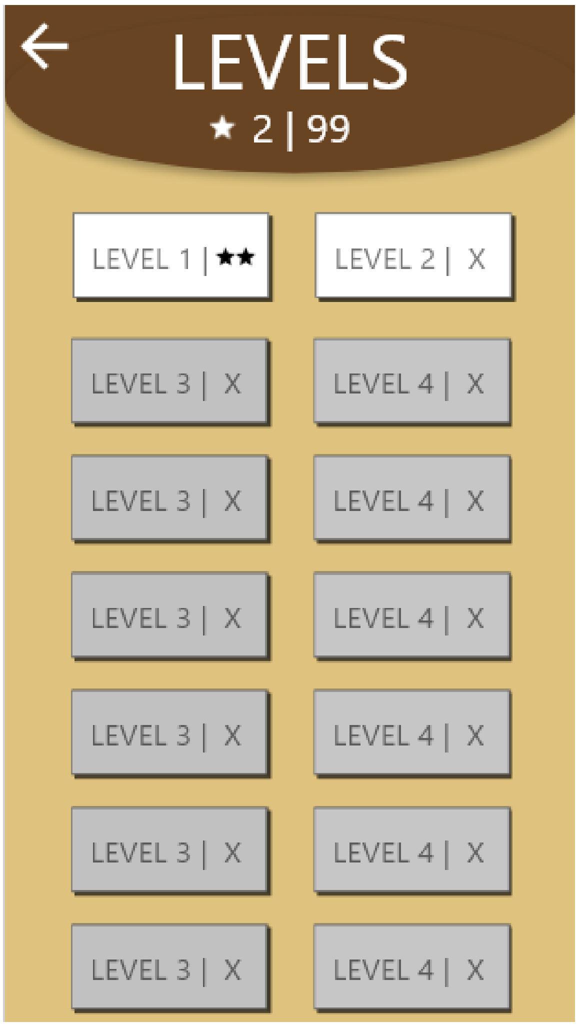


Figure 15: After - Levels

Figure 10 is the before image of levels. The buttons were not consistent with our other buttons, so to we added the 3D effect to the buttons, as seen on Figure 15. To improve the prototype itself, we also added a completed level to the screen, to showcase what it would look like.

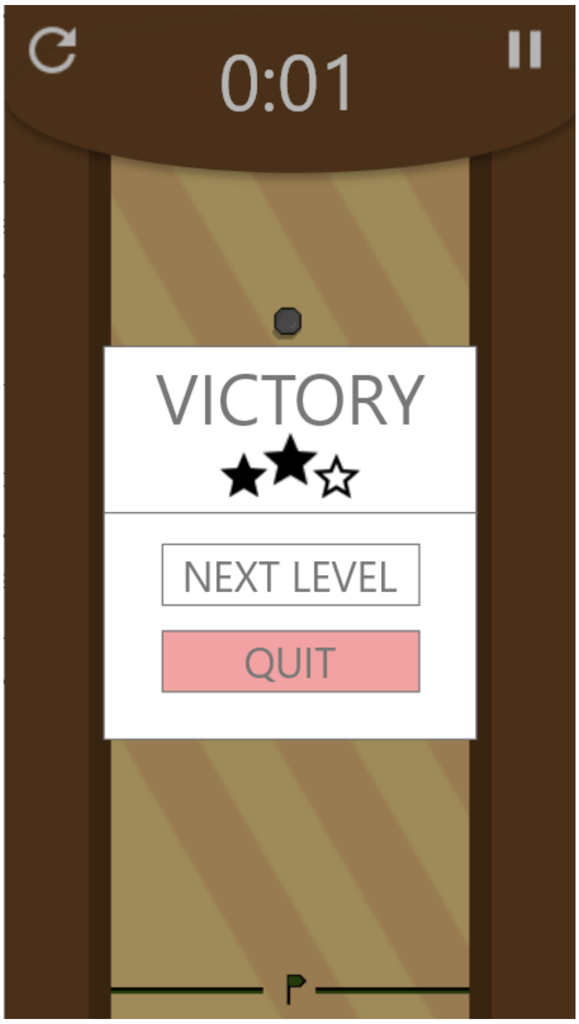


Figure 11: Before - Victory

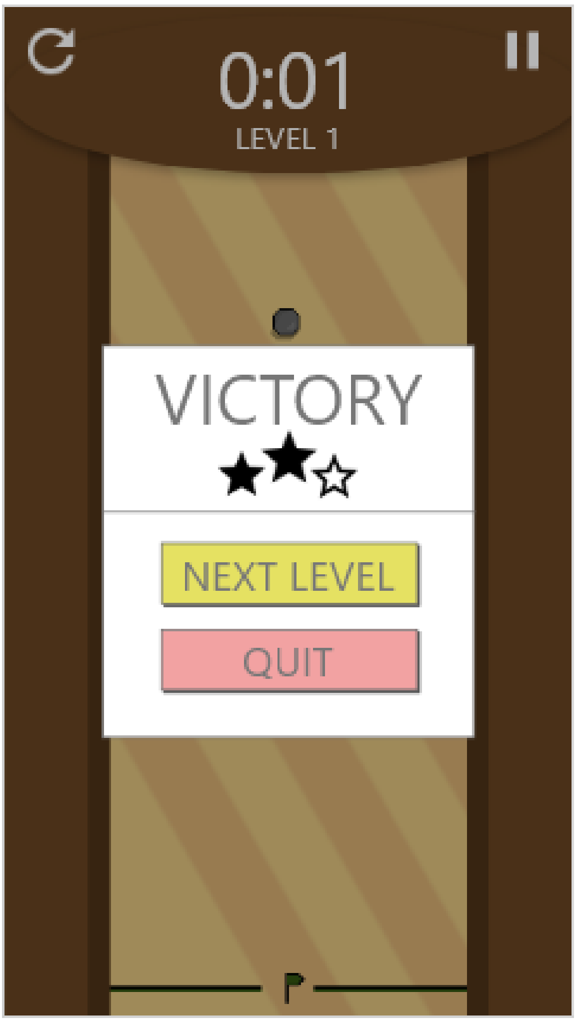


Figure 16: After - Victory

A similar issue appeared here with the inconsistent buttons as seen on Figure 11, they are flat. This was fixed by adding a 3D effect, shown on Figure 16. Another change was to make the ”NEXT LEVEL” button feel more rewarding it was recoloured to gold.

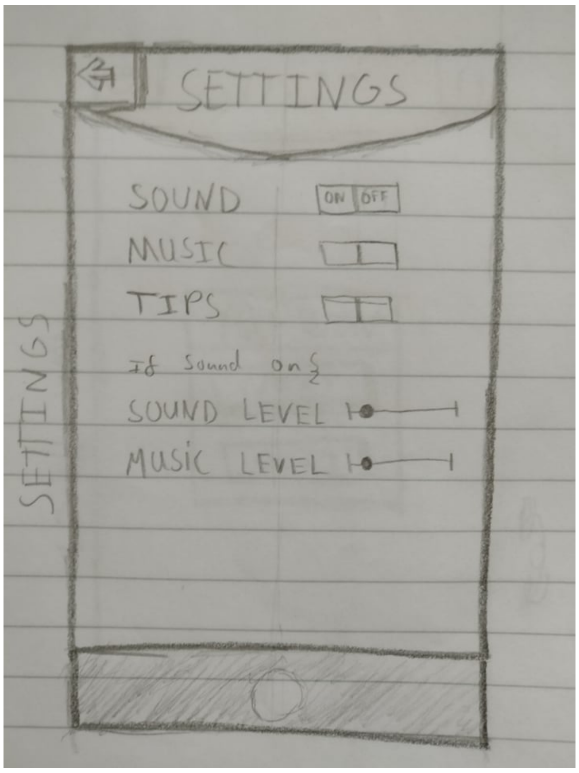


Figure 5: Before - Settings

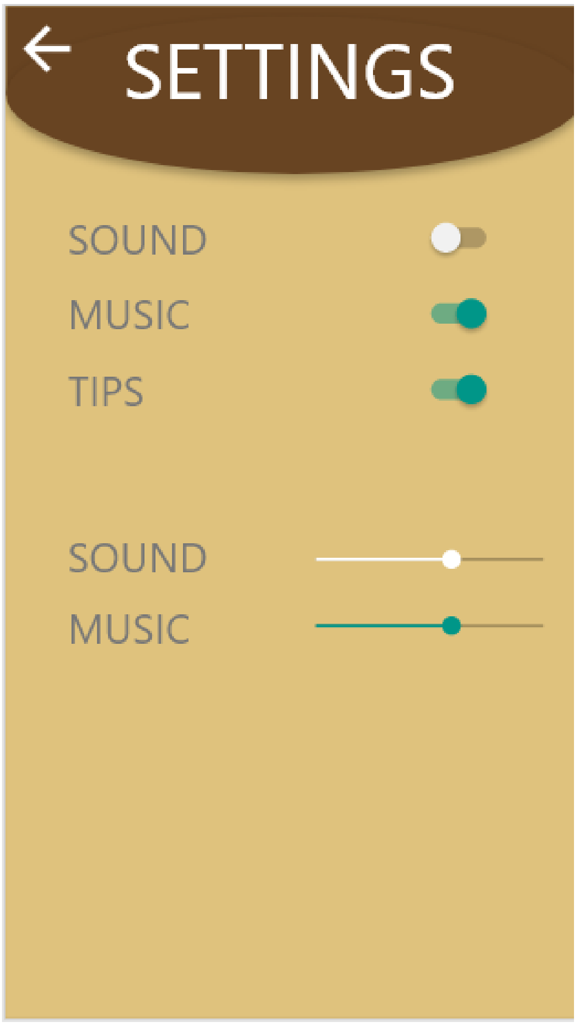


Figure 17: After - Settings

Figure 5 of our before settings doesn't really show it, but the "SOUND LEVEL" would disappear upon turning off sound. This was unclear what it meant, so we instead 'greyed' it out when turned off. As seen on Figure 17.

Reflection

Most of the former design worked well. Our problems stemmed from minor inconsistencies or things we missed. We made no major changes except adding a 3D effect to all our rectangle buttons, but that was more of an oversight. We contemplated making major changes according to the reviews we got, but our current design is more consistent with other gaming related applications. Consistency tends to be more important than useability, as the users get to know the application.

The prototype is missing some clear elements that helps to clarify things. Our hints/tips are not implemented, but they are intended to really help the user out on their first playthrough. A key missing part is also the hint showing what times gives what scores when playing. These are all weaknesses specific to the prototype, as they'll be implemented in the finished application. Design wise we have reached a fine middle point between a childish application and something that caters to an older demographic. Switching between icons and text from the main menu to the pause screen. A now consistent design between all buttons and also improving short-term memory by adding what level the user is playing.

Software Architecture

High-level architecture

The MainActivity is the activity opened when the app launches. This activity's UI is the main menu. From here the activity can open settingsActivity, which sets UI to settings menu. From MainActivity the LevelActivity, which sets UI to level menu and from here you open the GameActivity, which sets the UI to the Game class and therefore launches the game.
The Game class uses ConstraintLayout to set the static layout of the game. Overriding the onDraw method the dynamic layout of Game class is set.

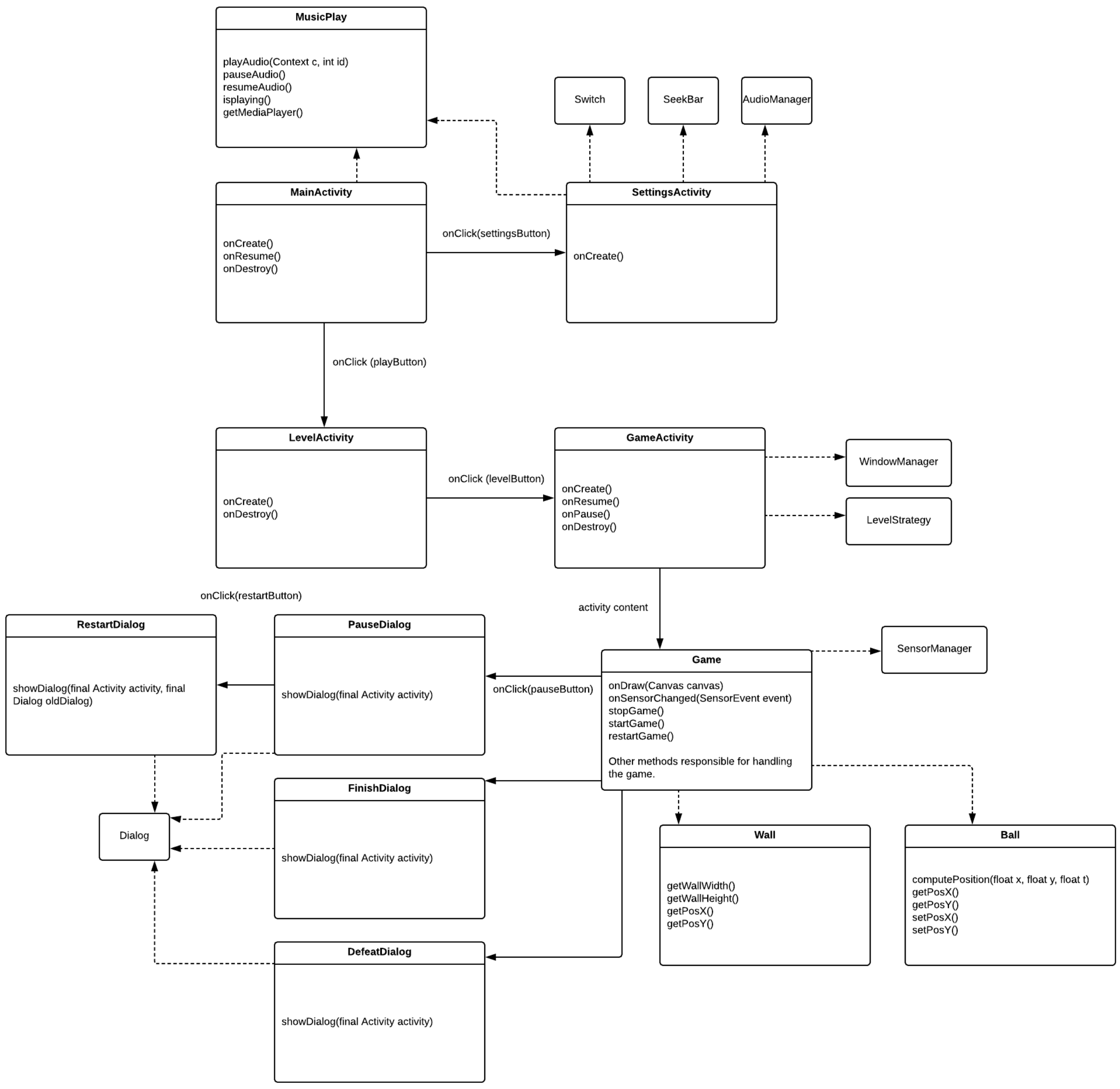
The game loop works by calling startGame() when GameActivity calls onResume() and this registers sensor listener and starts time until stopGame() is called. stopGame() is called when GameActivity calls onPause() and this unregisters sensor listener and stops the time until game is started again.
onSensorChanged() is used as the main method to apply changes to the ball, which stores accelerometer data and uses it to recalculate the position of the ball. The onDraw() method then redraws the ball, every time a new change has been applied to the ball, using the collected data.

Every UI is implemented using a ConstraintLayout. This is a ViewGroup, where it is possible to add views which is arranged to fit the screen as you wish.

MediaPlayer is used to display the background music across the application. It is used in conjuction with AudioManager, which controls audio on the android phone. That way the volume level of the music being played can be controlled.

The SensorManager is to collect accelerometer data given by the phone when rotating it. This is done by making sure the sensor type is an accelerometer and that the screen is not rotated, when recording new data.

The WindowManager is used to keep the screen on at all time, when playing the game. Also for all activities it makes sure to keep the screen on fullscreen (which hides the status bar).



The diagram illustrates relations between each class and their core methods and some are shown because they are not overridden and just uses the super method. The FinishDialog and DefeatDialog is not triggered by onClick but instead it is triggered when the game is either won or lost.
The activity content for MainActivity, LevelActivity and SettingsActivity is using XML layout, which is stored in three different XML files and here the UI elements are declared. This is not the case for GameActivity as it uses Game class as the activity content, which can be done because Game class extends ConstraintLayout.

Motivation for software design choices

Our game draws the graphics using a canvas, which could be done in many different ways. There is a lot of Game Engines like, LibGDX, OpenGL and AndEngine which can handle graphics for you. Using these Game Engines has benefits by giving a smoother game and more advanced features are possible. If our game was more advanced it would be beneficial to use these engines, but in that case it would be time consuming to learn a Game Engine and thereafter be able to use it. This is not the case when using the canvas, because of its simplicity and for that reason we build the game upon the canvas. We choose to delegate the Game out from the GameActivity because this allows us to further delegate the games features and keep same abstraction levels in same class. Currently some game handling features could be further delegated, which will be the case when a feature becomes too large in the Game class. Keeping GameActivity seperate from the game also gives the freedom to restart, start and stop game, while the activity is still running.

Another decision was made on the MediaPlayer or generally how to play our background music. The problem was that it had to last across the activities and be able to be turned off at the main menu and at the settings activity. One solution was making use of Services to turn a MediaPlayer into one. This would give good performance and the music would be played across activities, but it would make it harder to control the MediaPlayer inside the service and the music would persist even after closing the app. Harder to code as well. The other solution which we chose was making a MediaPlayer class and creating it at the MainActivity. To keep it consitently running through all activities we just to not finish our MainActivity, this can be costly on performance/memory depending on what is done. It makes it easy to control the MediaPlayer inside the class as you can call its methods from anywhere. This is likely to be changed at a later point when we introduce sounds to the game.

The Dialog API is a convention way of introducing a small window that prompts the user to do something before they can proceed. Which aligns with our design choice to have dialogs to yield closure. This also allows for custom layout at therefore every dialog was made with this API. When finished or defeated we use dialogs because this allows us to keep the GameActivity running and restart game when defeated. In the case of finish it could also be another activity, but to keep our design close to the prototype we made it a dialog.

Project log

Division of Labour

Our application could be split up in two main parts:
- Interface
- Gameplay

With some parts being a bit of a mix between the two, for an example, the stars you are rewarded for completing a level and Levels menu transitioning into Game. We naturally split it up, so Jonas is doing most of the interface and design, while Frederik is doing the gameplay parts. The Interface consist of activities, dialogs and their associated UMLs. This also includes the MediaPlayer and AudioManager components. The Gameplay revolves around the accelerometer and its android API SensorManager. It also makes use of WindowManager to incorporate a interface onto the game.

We plan to add these features at a later date and in the following order:
- Timer for levels
- Star reward system
- Replace placeholder graphics and finish design.
- Tips
- Sounds
- Leaderboards

Activity log

Friday 19th Oktober
Jonas (total: 1.5 hours)
  - Added barebone Main menu and Levels

Sunday 11th November
Jonas (total: 2 hours)
  - Main menu and Levels functional (design missing)
  - Added barebone Settings

Monday 12th November

Jonas (total: 0.5 hours)
- Minor refactoring

Frederik (total: 1 hours)
- Added barebone Game and GameActivity

Wednesday 14th November

Jonas (total: 2 hours)
- Background music implemented/controlled by settings and main menu shortcut
- Button added to Levels for Game testing

Frederik (total: 3 hours)
- Added Ball to Game
- Ball can now be moved with accelerometer

Thursday 15th November

Jonas (total: 3 hours)
- Buttons controlling music are now synchronized
- Changed app theme, looks more in line with our prototype
- General design changes
- Locked orientation to vertical

Frederik (total: 3 hours)
- Ball can't move out of screen
- Added finishline to Game
- Finish dialog added
- Ball movement changed
- Refactored Game

Friday 16th November

Jonas (total: 1 hours)
- Design changes on Main menu
- Added buttons and layout to finish dialog
- Implemented pause and restart dialog /restart mechanic not implemented

Frederik (total: 3 hours)
- Added wall to Game that triggers dialog when touched
- Level selection is now passed to GameActivity
- Bug fix for emulator

Saturday 17th November

Jonas (total: 2 hours)
- Implemented pause and defeat dialog /restart mechanic not implemented
- Graphical changes to Main, Levels and Settings
- Restart button in pause dialog now prompts a confirmation dialog
- Implemented a temporary restart to dialogs

Frederik (total: 3 hours)
- Finish line is now a ImageView
- Game restart is now implemented
- Movement is now paused on pause dialog
- Minor changes to Game and Ball

Sunday 18th November

Jonas (total: 1.5 hours)
- Refactored all XML files. They now correctly get values from values

Frederik (total: 0.5 hours)
- Game can now restart properly

Final Software Architecture

High-level architecture

The MainActivity is the activity opened when the app launches. This activity's UI is the main menu. From here the activity can open settingsActivity, which sets UI to settings menu. From MainActivity to the LevelActivity, which sets UI to level menu and from here you open the GameActivity, which sets the UI to the Game class and therefore launches the game.
The Game class uses ConstraintLayout to set the static layout of the game. The dynamic layout ofor the game is set by overriding the onDraw method.
The game loop works by calling newGame() in onCreate() and thereafter stopGame() is called when GameActivity calls onPause() and this unregisters sensor listener and stops the time until game is started again. startGame() is called when GameActivity calls onResume() and this registers sensor listener and starts time until stopGame() is called.
onSensorChanged() is used as the main method to apply changes to the ball, which stores accelerometer data and uses it to recalculate the position of the ball. The onDraw() method then redraws the ball, every time a new change has been applied to the ball, using the collected data.

Every UI is implemented using a ConstraintLayout. This is a ViewGroup, where it is possible to add views which is arranged to fit the screen as you wish.

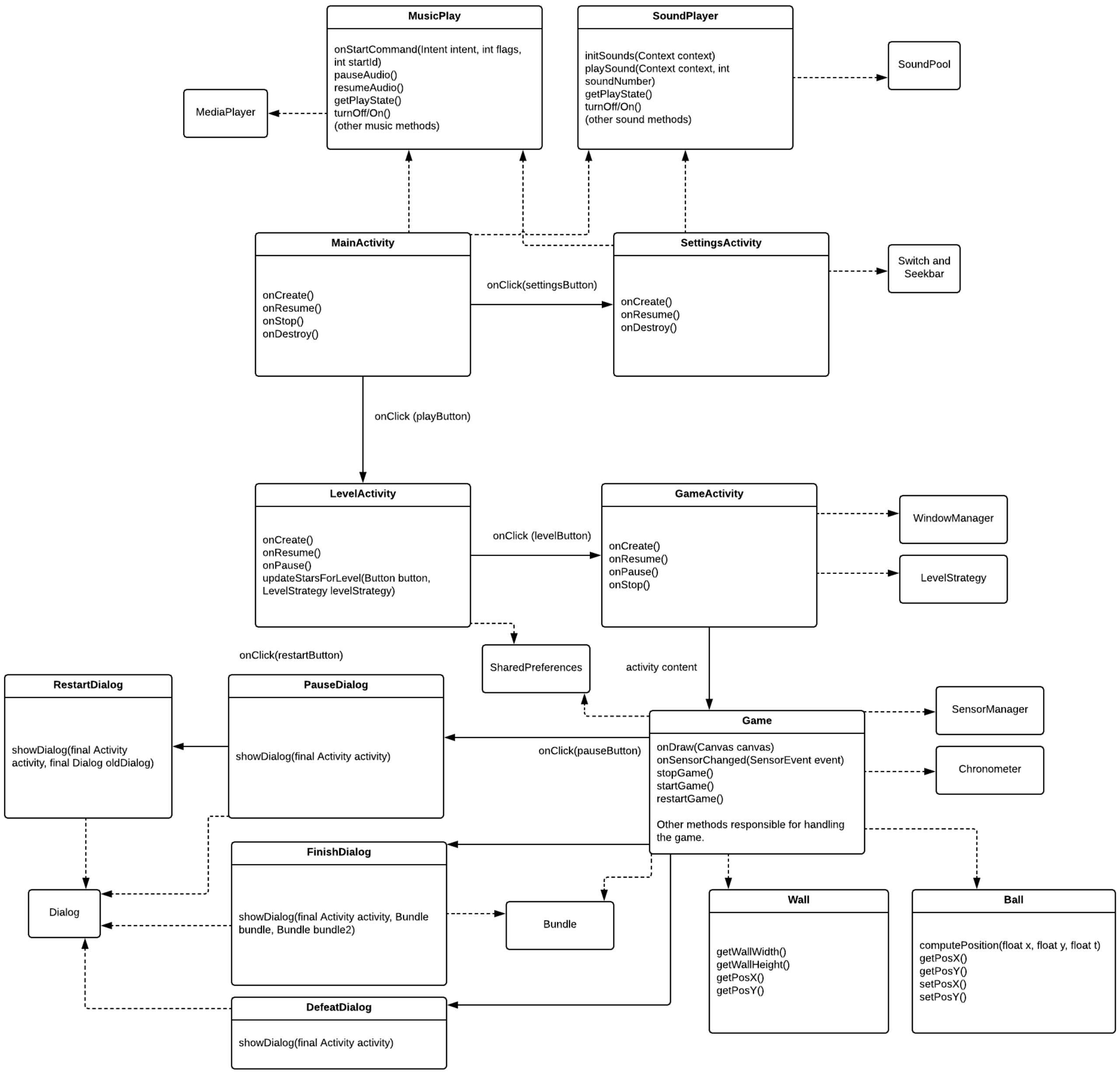
MediaPlayer is used to keep the background music playing across the application. SoundPool is used for playing small sound effects stored in a HashMap. All the sound effects are prepared at MainActivity. Volume control is done by increasing/decreasing the volume of the MediaPlayer or SoundPool itself.

The SensorManager is to collect accelerometer data given by the phone when rotating it. This is done by making sure the sensor type is an accelerometer and that the screen is not rotated, when recording new data.

The WindowManager is used to keep the screen on at all time, when playing the game. Also for all activities it makes sure to keep the screen on fullscreen (which hides the status bar).

The leaderboard would be implemented using mainly two Google Play service APIs. Which can be found in com.google.android.gms.games and com.google.android.gms.games.leaderboards
The way it works is that you make the user sign in/out of google account and using this account to open a leaderboard made in the google play console. This is done through Firebase, which is a part of Google play services. However because the app is unpublished, it has come to our attention that it is unavaliable to use without giving special permissions and releasing as alpha version. For this reason and credentials issues we did not implement this feature.

Bundle is used to parse the stars earned to the FinishDialog, which then shows a dialog matching the given value. To parse and save the stars into the LevelActivity we instead use Shared Preferences because the data should be kept even when the user destroys the activity.



Comparing this diagram to the last one. We added SoundPlayer class and with this comes SoundPool. The Game class now uses Chronometer, SharedPreferences and Bundles, where the last two helps communication across different classes. Several classes has been updated with new methods to support our new implementations.

Motivation for software design choices

Our game draws the graphics using a canvas, which could be done in many different ways. There is a lot of Game Engines like, LibGDX, OpenGL and AndEngine which can handle graphics for you. Using these Game Engines has benefits by giving a smoother game and more advanced features are possible. If our game was more advanced it would be beneficial to use these engines, but in that case it would be time consuming to learn a Game Engine and thereafter be able to use it. This is not the case when using the canvas because of its simplicity, and for that reason we build the game upon the canvas. We choose to delegate the Game out from the GameActivity because this allows us to further delegate the games features and keep same abstraction levels in same class. Currently some game handling features could be further delegated, which will be the case when a feature becomes too large in the Game class. Keeping GameActivity seperate from the game also gives the freedom to restart, start and stop game, while the activity is still running.

Another decision was made on the MediaPlayer or generally how to play our background music. The problem was that it had to last across the activities and be able to be turned off at the main menu and at the settings activity. One solution was making use of Services to turn a MediaPlayer into one. This would give good performance and the music would be played across activities, but it would make it harder to control the MediaPlayer inside the service and the music would persist even after closing the app. Harder to code as well. The other solution was making a MediaPlayer class and creating it at the MainActivity. To keep it consistently running through all activities we just have to not finish our MainActivity, this can be costly on performance/memory depending on what is done. It makes it easy to control the MediaPlayer inside the class as you can call its methods from anywhere. We chose the service so that the MediePlayer can be prepared asynchronous to everything else and won't hang our UI thread.

MediaPlayer could have been used for playing sound effects as well. That way we could initizalise a new sound effect and play it, but it would require a new class and every time a sound effect was to be used, it would have to re-initizalised. Not very suitable for small sound clips. SoundPool can initizalise all the songs at once, which is a problem if there is a lot of long songs, but that is not the case with sound effects. A call can then be made to call each sound effect when it is needed. Both can be done asynchronous, but the MediaPlayer would be slightly harder to code. That is why we chose SoundPool.

To keep a running timer, we use Chronometer class, which was mainly because of its simplicity to implement. It extends a TextView, which means we could set the timers layout firstly and then add the Chronometer functionality on top. Other options could be using Handler, but we stick with Chronometer because it can in few lines handle our task.

The levels are each an implementation of a LevelStrategy interface. In our case we do not autogenerate levels, which means we need a class for each level. For now this is fine, because the amount of levels are limited. If later we intend to make level generation this can be done also by implementing the LevelStrategy interface, but adding additional methods which handles generating a level given a parameter(int currentLevel).

To save stars earned locally we use shared preferences, this class already comes with strong consistency guarantees, but at the cost of speed. In our case there is only small data added to this internal storage, which means the slower speed compared to building a database is not noticed. In our case loss cannot be tolerated and that is why we use this storage across our application. It would also be viable to build a database using SQLite but as mentioned consistency need to be taken into account.

The Dialog API is a convention way of introducing a small window that prompts the user to do something before they can proceed. Which aligns with our design choice to have dialogs to yield closure. This also allows for custom layout at therefore every dialog was made with this API. When finished or defeated we use dialogs because this allows us to keep the GameActivity running and restart game when defeated. In the case of finish it could also be another activity, but to keep our design close to the prototype we made it a dialog.

Backlog and known bugs

Leaderboards isn't implemented, as we lacked time to properly implement it. We have an idea on how to implement as described in the "High-level architecture" chapter above.

More levels and levels generation was planned at a later stage, but it suffered the same fate as leaderboards. Hardcoded levels was put in as placeholders to showcase the gameplay.

The MusicPlayer is not always released properly cause we call it in onDestroy. This should have been refactored to be done in onPause, but that caused other issues and again lack of time to solve them.

Final Project Log

Division of Labour

Our application could be split up in two main parts:
- Interface
- Gameplay

With some parts being a bit of a mix between the two, for an example, the stars you are rewarded for completing a level and Levels menu transitioning into Game. We naturally split it up, so Jonas is doing most of the interface and design, while Frederik is doing the gameplay parts. The Interface consist of activities, dialogs and their associated UMLs. This also includes the MediaPlayer and SoundPool components. The Gameplay revolves around the accelerometer and its android API SensorManager. It also makes use of WindowManager to incorporate a interface onto the game.

Activity log

Friday 19th Oktober
Jonas (total: 1.5 hours)
- Added barebone Main menu and Levels

Sunday 11th November
Jonas (total: 2 hours)
- Main menu and Levels functional (design missing)
- Added barebone Settings

Monday 12th November
Jonas (total: 0.5 hours)
- Minor refactoring

Frederik (total: 1 hours)
- Added barebone Game and GameActivity

Wednesday 14th November
Jonas (total: 2 hours)
- Background music implemented/controlled by settings and main menu shortcut
- Button added to Levels for Game testing

Frederik (total: 3 hours)
- Added Ball to Game
- Ball can now be moved with accelerometer

Thursday 15th November
Jonas (total: 3 hours)
- Buttons controlling music are now synchronized
- Changed app theme, looks more in line with our prototype
- General design changes
- Locked orientation to vertical

Frederik (total: 3 hours)
- Ball can't move out of screen
- Added finishline to Game
- Finish dialog added
- Ball movement changed
- Refactored Game

Friday 16th November
Jonas (total: 1 hours)
- Design changes on Main menu
- Added buttons and layout to finish dialog
- Implemented pause and restart dialog /restart mechanic not implemented

Frederik (total: 3 hours)
- Added wall to Game that triggers dialog when touched

- Level selection is now passed to GameActivity
- Bug fix for emulator

Saturday 17th November
Jonas (total: 2 hours)
- Implemented pause and defeat dialog /restart mechanic not implemented
- Graphical changes to Main, Levels and Settings
- Restart button in pause dialog now prompts a confirmation dialog
- Implemented a temporary restart to dialogs

Frederik (total: 3 hours)
- Finish line is now a ImageView
- Game restart is now implemented
- Movement is now paused on pause dialog
- Minor changes to Game and Ball

Sunday 18th November
Jonas (total: 1.5 hours)
- Refactored all XML files. They now correctly get values from values

Frederik (total: 0.5 hours)
- Game can now restart properly

Saturday 1st December
Jonas (total: 2 hours)
- Refactored MusicPlay to be a service
- Implemented SoundPool and added sound effects to buttons

Sunday 2th December
Jonas (total: 2 hours)
- MusicPlay is now played async
- Last sound effects implemented
- Sound and music volume can be controlled independently
- Barebone GameDialog added

Monday 3th December
Jonas (total: 1 hours)
- Removed outline on dialogs
- Ball and finishline graphics implemented

Frederik (total: 4 hours)
- Implemented timer to Game
- Stars can now be earned based on time in Game
- Game pauses on GameDialog
- Game shows the current level being played

Tuesday 4th December
Jonas (total: 3 hours)
- Stars are now shown graphically in levels. Hardcoded currently
- Timer refactored to use seconds in Game
- Fixed graphical errors with dialogs and stars
- Levels now have stars earned independently. No longer hardcoded

Frederik (total: 4 hours)
- Added more levels to Game
- Walls are now an arraylist in Game
- Each level keep track of the times required for stars
- GameDialog is now passed times from Levels
- Fixed bug with DefeatDialog
- Fixed bug with resetting timer in Game

Wednesday 5th December
Jonas (total: 1 hours)
- Added launcher icon
- Changed the star and ball sprite
- GameDialog layout finished

Frederik (total: 0.5 hours)
- FinishDialog's Next Level takes you to the next level
- fixed small bugs.

Thursday 6th December
Jonas (total: 1 hours)
- Refactored XML files to use more styles
- Added missing comments
- Bug fix with the placement of buttons

Frederik (total: 0.5 hours)
- Added missing comments

References

[1] Raquel Viejo-Sobera Marc Palaus Elena M. Marron and Diego Redolar-Ripoll. *Neural Basis of Video Gaming: A Systematic Review.* URL: `https://www.frontiersin.org/articles/10.3389/fnhum.2017.00248/full`. (accessed: 09.09.2018).

[2] David J Linden. *Video Games Can Activate the Brain's Pleasure Circuits.* URL: `https://www.psychologytoday.com/us/blog/the-compass-pleasure/201110/video-games-can-activate-the-brains-pleasure-circuits-0`. (accessed: 09.09.2018).

[3] Jakob Nielsen. *User Satisfaction vs. Performance Metrics.* URL: `https://www.nngroup.com/articles/satisfaction-vs-performance-metrics/`. (accessed: 09.09.2018).

[4] Sarah Perez. *Consumers Spend 85% Of Time On Smartphones In Apps, But Only 5 Apps See Heavy Use.* URL: `https://techcrunch.com/2015/06/22/consumers-spend-85-of-time-on-smartphones-in-apps-but-only-5-apps-see-heavy-use/`. (accessed: 09.09.2018).

[5] Gavin Mann. *Digital Video and the Connected Consumer.* URL: `https://www.accenture.com/t20160512T013702Z_w_/us-en/_acnmedia/Accenture/Conversion-Assets/DotCom/Documents/Global/PDF/Dualpub_15/Accenture-Digital-Video-Connected-Consumer.pdf`. (accessed: 09.09.2018).

[6] Heather Nofziger. *Share, Stream, and Show - How today's teens interact with mobile games.* URL: `https://www.gamesindustry.biz/articles/2018-01-25-share-stream-and-show-how-todays-teens-interact-with-mobile-games`. (accessed: 09.09.2018).

[7] Mariusz Sojka. *Adjust your app to your users needs: UI design for children, Millenials and older people.* URL: `https://www.ready4s.com/blog/adjust-your-app-to-your-users-needs-ui-design-for-children-millennials-and-older-people/`. (accessed: 09.09.2018).

[8] Kenneth A. Lachlan Bradley S. Greenberg John L. Sherry and Amanda J. Holmstrom. *Orientations to Video Games Among Gender and Age Groups.* URL: `https://www.researchgate.net/publication/247740438_Orientations_to_Video_Games_Among_Gender_and_Age_Groups`. (accessed: 22.09.2018).