

University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Master's thesis

Mobile application focused on practicing cognitive functions

Místo této strany bude
zadání práce.

Declaration

I hereby declare that this master's thesis is completely my own work and that I used only the cited sources.

Pilsen, 10th August 2020

Petr Lukašík

Acknowledgement

I want to thank the supervisor, Ing. Ladislav Pešička for expert advice during the creation of diploma thesis. Moreover, I thank my whole family for the support throughout the final year, and at last, I want to thank highly appreciated testers Bc. Filip Jani, Ing. Jan Šedivý and Bc. Jan Palcút, for their endurance on time spent on the app testing.

Abstract

This diploma thesis describes the development process and the final product of a mobile application used to practice cognitive skills. The aim is to make the suitability of such a health-related application known to a broader range of people. Existing applications focused on cognitive skills training are first analyzed. From this, it is deduced the result of deficiencies that this application eliminates. Cognitive abilities are summarized in five main types, according to various available literature publications on this topic. Based on the types of cognitive functions, training games are modeled. Subsequently, a system consisting of a mobile application and a server for collecting statistics from games played is designed. This system is implemented by combining system design and modeled training games. The resulting mobile application is appropriately tested, and lastly, possible further extensions are proposed.

Abstrakt

Tato diplomová práce popisuje proces vývoje a finální produkt mobilní aplikace sloužící pro procvičování kognitivních schopností. Cílem je informovat širší okruh lidí o vhodnosti takové aplikace související se zdravím. Existující aplikace zaměřené na trénink kognitivních schopností jsou nejprve analyzovány. Z toho je vyvozen výsledek nedostatků, které tato aplikace odstraňuje. Kognitivní schopnosti jsou shrnuty do pěti hlavních typů, dle mnoha dostupných literárních publikací na toto téma. Na základě typů kognitivních funkcí jsou vymodelovány tréninkové hry. Následně je navržen systém skládající se z mobilní aplikace a serveru pro shromažďování statistik z hraných her. Tento systém je implementován zkombinováním návrhu systému a vymodelovaných tréninkových her. Výsledná mobilní aplikace je řádně otestována a nakonec jsou navržena možná další rozšíření.

Contents

1	Introduction	1
1.1	Mobile devices and health apps	1
1.2	General view on cognitive training apps	3
1.3	Issues with cognitive training apps	4
2	Cognitive functions	7
2.1	Memory	7
2.1.1	Short-term memory	8
2.1.2	Long-term memory	8
2.2	Attention	9
2.2.1	Focused attention	9
2.2.2	Sustained attention	9
2.2.3	Selective attention	9
2.2.4	Alternating attention	10
2.2.5	Divided attention	10
2.3	Visual-Spatial processing	10
2.4	Language and speech skills	12
2.5	Executive functions	13
3	Analysis of selected applications	14
3.1	Lumosity	14
3.2	CogniFit	15
3.3	Peak	15
3.4	Elevate	16
3.5	NeuroNation	16
3.6	Mensa Brain Training	17
3.7	Analysis summary	17
4	Games functionality theory and scenarios design	20
4.1	Memory	20
4.1.1	Box lookup	20
4.1.2	Dangerous path	22
4.2	Attention	23
4.2.1	Flying frog	23
4.2.2	Two cars	24
4.3	Visual-Spatial	25

4.3.1	Follow the owl	25
4.3.2	Hiding mosquito	26
4.4	Language and speech	27
4.4.1	Color confusion	27
4.5	Math	28
4.5.1	Bubbles	28
4.5.2	Estimation	29
4.6	Games for future expansion	31
4.6.1	Language and Speech - Words	31
4.6.2	Attention - Flanker test	32
5	System design	34
5.1	Operating systems	34
5.2	Server and database model	35
5.2.1	Database	37
5.2.2	Database model	38
5.3	Programming language and framework	40
6	Implementation	42
6.1	Project structure	43
6.2	Firestore services	44
6.2.1	Firestore with LibGDX	46
6.2.2	Users data gathering	47
6.3	Assets	49
6.3.1	Game assets	49
6.3.2	Localization	51
6.3.3	App textures	52
6.4	App screens	54
6.5	Games	56
6.5.1	Box lookup	57
6.5.2	Dangerous path	58
6.5.3	Flying frog	58
6.5.4	Two cars	59
6.5.5	Follow the owl	59
6.5.6	Hiding mosquito	60
6.5.7	Color confusion	60
6.5.8	Bubbles	61
6.5.9	Estimation	61

7	Testing of implemented app	63
7.1	Mini-scenarios	63
7.1.1	App launch	63
7.1.2	User authentication	64
7.1.3	User registration	67
7.1.4	User settings	69
7.1.5	Games information	71
7.1.6	Games activities	74
7.1.7	User statistics	78
7.2	Continuous testing	80
7.3	Cracking up the problems	81
7.3.1	Minimal API level	82
7.3.2	Fonts	82
7.3.3	Emulator	82
7.3.4	Device texture blackouts	83
7.3.5	Framebuffer	84
7.3.6	Display cutout alias Notch	84
7.3.7	Elderly people game test reports	84
8	Future improvements	86
8.1	Publish app	86
8.2	Email confirmation and forgotten password	86
8.3	Custom solution to LibGDX Firebase API	87
8.4	iOS launch	87
8.5	Web application	88
8.6	Data sharing and extended data gathering	88
9	Summary	89
	List of abbreviations	90
	Bibliography	91
	Appendices	97
A	List of all testers	97
B	Games documentation	98
C	Contents of attached DVD	108

1 Introduction

With developing modern society comes more possibilities for clinical advice or even improvements through technology. Notably, the rise of health-related apps on the mobile platform stores by the last decade created a stir that has not gone unnoticed by many scientists. This project is based on the initiative of the author's interest in developing mobile entertainment apps and connecting it with his studies' medical specialization. The purpose is to explore the possibilities of training and improving cognitive functions through existing apps. By familiarization with existing cognitive training apps, it intends to develop proper functionality and design game scenarios. Afterward, building and implementing a system composed of a prototype of own cognitive training mobile app, a server, that will gather statistics about games from the mentioned app, and a presentation layer, where users can see their achieved progress and statistics.

1.1 Mobile devices and health apps

Since this app is from the beginning focused on mobile devices, this section will cover a basic overview of how mobile devices are playing a higher role in our casual life. Altogether mobile health-related apps are on the rise and are helping with more and more human health issues.

To give a general overview, the internet use and the ownership of mobile devices, for this research needs, especially smartphones, are on the rise mentioned in the report of Pew Research Center from 2018. The report claims that smartphones are essential tools in modern society, especially in poorer countries, without a fixed internet connection. In 2013-2014 around 25% of emerging or developing economies reported using a smartphone, and by 2017 this percentage has risen to 42%. For advanced economies, this number stalls around the same rate of 72% in the years 2013-2017.[1]

The development of mobile health apps has increased with people adapting to the use of smartphones and wearable sensors for medical or just self-care reasons. The central vision of the rise can be seen in the number of released apps in major app stores. In 2014 the number of published health-related apps was over 100,000 collectively on Android and iOS leading platforms [2]. However, since then, the number to 2017 has surpassed

over 350,000, which is more than three times higher per three years [3]. Of course, with more health apps available for use in a clinical environment, the doctors can have problems suggesting to the patients the quality ones.

Health-related apps are handy for many people and even for many scientists, thanks to the gathered data. This way, a lot of workout and fitness apps, which helps, tracks, and recommends training plans to users, gained tremendous popularity. Maybe even because of that, the opinion of people changed on health-related apps, and they perceive it as a technology that can support a healthy lifestyle or even can save lives. So with the current situation in the world, newly developed apps for tracking users and providing information about COVID-19 are widely downloaded and used. Those help governments to track problematic locations and preventing another spread by predicting infected users, thus moving them into quarantine. This information is based on the statistics from April 2020, which shows the top ten most downloaded health-related apps from Google Play Store, where Brazil app "Coronavírus - SUS" has taken second place with 2.17 million downloads from health-related apps in March 2020 [4]. Even though it was created only for Brazil, it is a great achievement, and supposedly other nations also have similar projects. For example, the Czech Republic has created not long ago "eRouška" for the same COVID-19 purpose or app "Záchranka" for simple contact with emergency medical service.

Naturally, the health app invented by this thesis should mainly focus on training cognitive functions for elderly adults, yet they could have potential problems with the smartphone and internet technology. These problems have been addressed in the past. Nevertheless, nowadays the older adults are mostly used to smartphones and internet use. Defined by the Pew Research Center demographics gathered from the United States for the year 2019, 52% of older adults of age above 65, owns a smartphone [5]. The same surveyed age group stated previously has also experienced a high rise in internet usage from 14% in the year 2000 to 73% in the year 2019 [6].

With this data, it is possibly safe to say that elderly users are not having many problems with mobile technology and are keen to learn with it. However, attempts to use it may be accompanied by frustration. This has been noted in Nikky L. Hill's research which supported that, even when older adults are open to learning, benefits of using mobile apps, need proper, adequate targeted training, and device interaction optimized with older adults in mind. For example, lower the possibilities of interactions, the option to enlarge the font or access to simple tutorials in their native language. [7]

With a focus on health information, older adults are firstly trusting health providers as the best source of information because of knowledge,

personal connection, and trust altogether with information gathered from family or friends. On the imaginary second place lies the internet to access sought health information, yet many older adults expressed concerns about the quality of found online information.[8]

1.2 General view on cognitive training apps

For many scientists, it is very complicated to embrace computer-based cognitive-training software, also known and called "Brain apps," which are applications suitable to improve cognition functions. Looking through their perspective, "aggressive" advertising of brain apps claims to reduce or reverse cognitive decline is not supported by enough evidence, more precisely by scientific background. Additional claims are cognitive functions that can preferably be strengthened by doing physical exercise, which is already scientifically proven or that studies conducted by the researchers with financial interests in the products are not enough.[9]

Not long after the previously mentioned statement has been announced, a pack of experts in the field of neural plasticity has responded with an open letter, where they expressed their disagreement with the statement as consensus from the scientific community. Strong disapproval is with the claim that little to no evidence is present for brain exercising apps, which resulted in showing published papers with evidence on the benefits of this conflicting topic. Despite the fact that agreements have been done with a lot of the article claims in the response. Undoubtedly, the quality of such brain training programs needs to be divided into appropriate and inappropriate. If one application is developed with a team of programmers cooperating with physicists, doctors, scientists, and other experts with in-depth knowledge of neuroplasticity, it will give much more contribution to society and science than applications developed by gamers, who have almost none brain functionality knowledge.[10]

The difference between a casual "brain game" and an effective cognitive training program is, therefore, noteworthy. Health and Medicine Division of the National Academies published in 2015 five criteria by which should be cognitive training programs evaluated to see the effectivity of those trainings [11]. The transcript of the criteria is as follows:

1. Has the product demonstrated transfer of training to other laboratory tasks that measure the same cognitive construct as the training task?
2. Has the product demonstrated transfer of training to relevant real-world tasks?

3. Has the product performance been evaluated using an active control group whose members have the same expectations of cognitive benefits as do members of the experimental group?
4. How long are the trained skills retained?
5. Have the purported benefits of the training product been replicated by research groups other than those selling the product

The perception of cognitive training by the general public is perceived in the right way based on the research from this year. Participants were asked questions regarding reasons for using cognitive training programs, motivations to continue using them, and if the perception of application utility was changed by presumed cognitive results or in time. The study showed that a significant group of participants that previously used cognitive training programs had endorsed better results against the previously non-using participants. Furthermore, individuals with self-reported psychological or neurological disorders were much more likely to use cognitive training programs than other members of the general population.[12]

1.3 Issues with cognitive training apps

Vergani et al. wrote about general issues in cognitive training apps that not many of them have background studies with a focus on improving cognitive performance. Effectivity of those apps could not be determined with certainty. Altogether, homogeneity about specific cognitive function training is not generally defined, and many apps are diverging with the consistency of training for specific cognitive abilities (e.g., executive functions may include attention function). Connected with inconsistency in apps, is even inconsistency in studies about cognitive domains that are differently labeled, and thus it is difficult to prove which cognitive ability is improved with cognitive training app.[13]

Based on the discussion section in the J. Zhao's article, he pointed out some of the main aspects of successful applications that had a high retention rate. For weight loss apps, participants achieved more significant improvement in physical and behavioral outcome measures when apps supported expert-designed, tailored content to users' preferences and interests, and weekly supportive text messages. In cases with exceptional personalized feedback on progress provided by weekly telephone consultations with experts were having much higher success of retention than non-personalized

generic text suggestions. Young adults also responded with evaluating real-time feedback, tailored information, and expert consultations as the most useful and made them prolong using the app with higher standards. Furthermore, apps with better designed and more straightforward interface has shown that participants are more likely to be longer using this application over others.[2]

Another feedback is by a tested group of older adults with age 60+ on Attention Training Application (ATA). Each individual had a 2-hour session with a member of the study team to use their developed app. Participants were mostly negatively affected (distraction and frustration) by app feedback on correct or incorrect responses. Although feedback on correct responses was considered as a motivator for some, the app was not always responding to users' touch, yet this information could be misleading as it has many other factors as programming faults or detecting user power of pressure on the device. In summary, participants were also less incentive to use the app more often, because the app was uninteresting, did not have specific goal or theme.[7]

Many other types of research support using handheld devices for cognitive training, but does not mention arisen issues or discomfort while using them for training. These apps specifically focused on a broad range of disorders suppressing examined groups of patients, are diverting from ADHD, multiple sclerosis, depression to strokes, or cancers concludes that these apps are mostly beneficial for related groups. [2] [13] [14] [15] [16] [17]

Even though this section does not offer a large proportion of studies with users feedbacks, combined common sense with a summarization of previous paragraphs, a few key aspects to keep in mind when developing a cognitive training app are following:

- Games should be recognized by single cognitive ability and not by cognitive function group (e.g. executive function -> working memory)
- App should be graphically interesting and easy to use to keep the interest of users for regular use.
- Feedback on success is motivational, but with it should be approached in moderation (do not show positive outcome after every successful action)
- Content should be tailored to users' preferences and interests
- Bad responsivity of application can degrade user interest in the continuing use of the application

On the other side, even when this application is going to be mainly developed for smartphones, with nowadays possible technologies, the choice of development will also take in consideration of using cross-platform possibilities, which will be further discussed in the section 5.3.

2 Cognitive functions

These functions relate to the primary mental abilities that allow us to carry out any task. It forms how the person understands the world and perception of his actions on the surroundings. Mentioned processes we do in our daily life are mechanisms of how we learn, remember, plan our actions, problem-solve, or make a decision. While a healthy younger person does not even notice using these abilities, they weaken with increasing age or by some injury, when a piece of our peculiar self can be lost. It can hinder our daily functioning, and we can feel being less useful and uncertain.

Given by many scientific works, it is known that training cognitive skills can improve and maintain seniors' cognitive competence, as mentioned by MD-PhD Croisile in 2006 [18]. These trainings are even translated to a computer or mobile training as one of the beneficial aids, which in current modern times older adults use with ease and not disgust.

As scientists never fully agreed on given groups of cognitive abilities, there exist many different approaches for basic function definitions. This paper will separate cognitive functions into five main groups as memory, attention, visual-spatial processing, language, speech skills, and executive functions described by Klucká in 2009 [19].

2.1 Memory

Memory is a mental function that helps us remember all kinds of information, experiences, and perceptions. The process allows us to receive, store, and recover information from the past. It is essential in creating and developing our personality. Other than that, well-working memory is influencing a lot of other critical cognitive functions, e.g., executive functions.

In the past, scientists thought that for different parts of information, there was only one type of memory. Nowadays, the opinion prevails, that memory is divided into more categories, which was first described in 1968 by Atkinson and Shiffrin (see picture 2.1) [20]. Sensory memory presents information that we experience in the time window of a few hundred milliseconds. For this type of memory, the perfect example is visible persistence. In terms of the length of storage memory, we distinguish between the other two main segments.

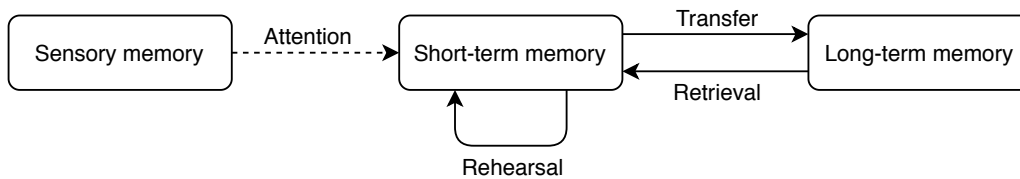


Figure 2.1: Atkinson-Shiffrin model (also known as the Multi-Store Model)

2.1.1 Short-term memory

Short-term memory retains information for a short period of time (seconds). For example, remembering verbally received phone numbers until they have been written down on the paper. Short-term memory has limited capacity in the number of objects an average human can hold, which is being called a "magical number" with a range of 7 ± 2 , as noted by Miller, G. A. in 1956 [21]. Every person can find out for himself the number by rewriting just seen, e.g., a shopping list on another paper. This number is higher when the person is logically connecting any items, which then becomes as one.

2.1.2 Long-term memory

Long-term memory contains memories that are being kept for an extended period of time (minutes-years). From short-term memory, only some significant or repeated information is transferred to long-term memory. The transfer process is being positively influenced by how it is understood and connected to the previously acquired experiences. Long-term memory is not a single entity, but it is composed of several different components [22].

Procedural memory

This memory is considered as an implicit memory that holds up profoundly embedded skills that we do automatically and are no longer aware of them. It refers to learning and storing information through repetition and practice, such as knowing how to ride a bike, playing the piano, or cook some recipe. It takes knowledge from procedural previously-stored memory and forms new memories without an effort. An excellent example is when lyrics to a popular song playing on the radio is stuck in your head.

Declarative memory

Also called an explicit memory since it consists of information that is explicitly stored and retrieved. It is further sub-divided into two other subtypes

semantic and episodic memory [23].

If we remember general knowledge or universally valid facts, we speak about **semantic memory** that is commonly used in the process of learning new information from school, e.g., knowing grammar rules or remembering capital cities in the EU.

The second type is **episodic memory**, which is when we try to recall personal memories, e.g., dentist appointment, grandma's unique cooking, or memories on friends. The memorization context is vital for this type of memory.

2.2 Attention

Attention is the ability to choose and concentrate focus in a specific direction. This function goes hand in hand with memory, and it is crucial and essential in our daily life. While a huge amount of stimuli is continuously flooding us, attention protects us by selecting more important ones in the given moment. There are five main types of attention based on the clinical model based on the work of Sohlberg and Mateer [24] we discuss below.

2.2.1 Focused attention

It is described as a basic reaction that refers to focus on any stimuli. Whether it is focusing on a sound, smell, some visual aspect, or just feel, everyone should possess this ability to focus on these stimuli. The one who does not have this ability is having severe brain problems that will not allow them to engage this type of attention.

2.2.2 Sustained attention

Also associated with concentration and vigilance, allows us to focus on one specific task for an endless amount of time. In the school sphere, the example would be applied in the lecture where a student needs to keep focus without getting distracted. It can be challenging to maintain attention for a significant amount of time, and after tens of seconds, the focus may begin to lapse. However, the key aspect is the ability to re-focus on the task.

2.2.3 Selective attention

Throughout the day, we can taste, smell, hear, touch, and see many stimuli. However, we cannot focus on every stimulus equally to deal with the

problems simultaneously. That is why we are reacting only to some of them by selecting them not to get overwhelmed. Thanks to selectivity, we can block certain features out of the environment to focus, for example, on our conversation with a friend.

2.2.4 Alternating attention

Used modern word for this type of attention is multitasking, whereas the switchability trait gives us the ability to switch from one stimulus to the other. We take in mind that these two tasks have different cognitive requirements that use different areas of the brain. Using this ability is done in daily life regularly. An example from a student of applied sciences is learning to program in a new language, where he first reads some documentation (learning) and then writing the newly learned code (doing). Another possible example could be doing two unrelated tasks like when talking to a friend, and then swiftly reacting to nearby starting fire by putting it out. Mentioned by the Kulišťák in 2003, this ability is getting worse with increasing age [25].

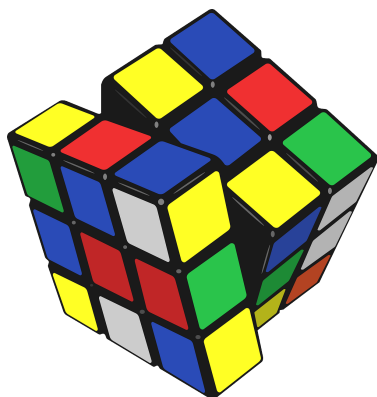
2.2.5 Divided attention

The ability to do two or more things at once is divided attention, and it allows us to split attention. An example of dividing attention is when a person is driving a car and simultaneously listening to the news on the radio or speaking with a passenger. It is humanly impossible to concentrate on two different tasks concurrently because the brain can process only one task at a time. So, in reality, the brain is continuously alternating the attention between tasks. The distribution of attention is even simpler when every one of them performed tasks is more automatized, so the person does not need to be entirely focused on it. It tends to be believed that the increasing age of the person decreases this multitasking ability.

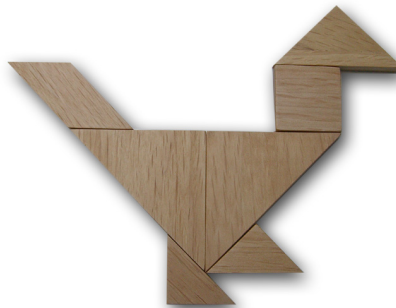
2.3 Visual-Spatial processing

Visual-Spatial skills help the person to gather information from the world around him to find his orientation in space, perceive objects around him, and organize that information into a visual scene. From this visual scene, the individual can grasp spatial relationships between objects and visualize objects that are not physically present. Basic well-known and popular games

Rubik's cube and tangram that are used to train our visual-spatial function are shown in image 2.2.



(a) Rubik's cube [26]



(b) Tangram [27]

Figure 2.2: Popular games to train visual-spatial skills

Manipulation with two-dimensional and three-dimensional objects on a surface or in space shows essential information about right hemisphere functionality. Difficulties can be seen for older adults with an example of finding the closest soccer ball near the football goal. While a simple task to do, this relies on complex cognitive processes. Recognizing shapes, viewing perspective, the number of soccer balls, determining their position, orientation, size, or even moving direction and then estimating the distance between them and football goal makes this decision particularly difficult for our brain. Hijazi M.M. mentions that visual sense plays a vital role in sports. She underlies this idea in an article focused on fencing [28], where she claims an athlete can have excellent visual acuity, yet that does not mean he can determine his place in space, how quickly opponent moves or in which direction opponent moves. Other problems can also be seen while orienting own body in space on unknown or less known places e.g., shopping mall.

One aspect that should also be covered by this topic is imagination and fantasizing. This aspect is a cognitive task that helps to perceive something that is not there, e.g., objects, sounds, or smells. This is widely used by chess players, where they imagine possible most useful moves ahead with the opponent's moves, which he can not even touch in reality.

As mentioned in Klucká's book [19], disruption of visual-spatial skills also leads to deterioration of motor skills. This may result in apraxia, which means a loss of previously acquired complex moves. Dressing apraxia is the inability to dress by coordinating and organizing individual movements. The problem is in the integration of those tasks into one.

2.4 Language and speech skills

Language is the main instrument of communication and characteristic of humans. We use it in reading, writing, and speaking to understand others and express ourselves. Speech is a specific language skill to create sound from very complex coordination of lip, tongue, inner mouth, and vocal cords movements. With this ability, we try to create meaning by building sentences using appropriate words together according to the grammar rules applicable.

We distinguish between two basic categories while using language: production and understanding. While speaking, the sentences are produced by transforming the thought into words. After that, they are then put together to form a sentence. Finally, the sound is created that then transfers the thought to the listener. With understanding its the reverse procedure whereby hearing sounds begins the conversion into words with meaning, connecting them into sentences and finally getting the statement of the message that the speaker wanted to tell [29]. Retrieval of words can be described by Cohort model (picture 2.3) proposed by William Marslen-Wilson [30], which is saying that person is mapping words letter by letter in realtime, received by the visual or audio input, to a hearer's lexicon of known words. With more segments, the hearer is narrowing down the options until only one left that still matches the output.

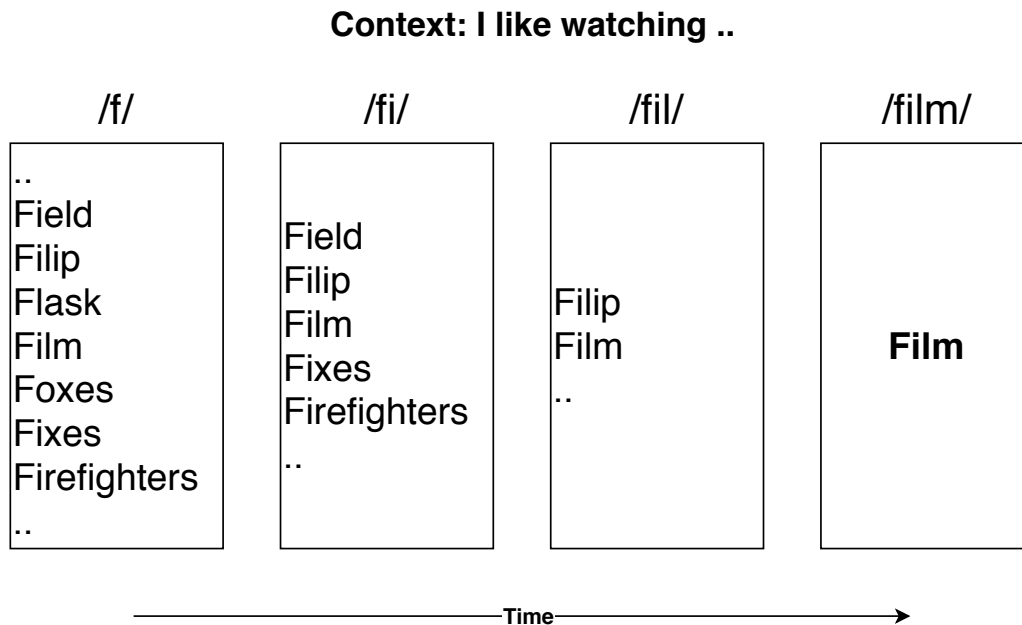


Figure 2.3: Cohort model word recognition

One of the aspects of language cognitive function is writing and reading.

While writing, it is common to follow grammar rules and spelling and pay attention to avoid repetitions by using synonyms [31]. Also, it depends on which age group or audience the text is written for. When reading, it is necessary to memorize pieces of information from previous sentences to help to understand the next sentence or whole paragraph. Redundant pieces of information are mostly immediately erased from memory, and we keep only important ones in our memory, so the whole text makes sense to us.

2.5 Executive functions

Executive functions are a set of essential skills necessary for life. This includes functions to plan activities, solve problems, do more activities at once, logical reasoning, and coordinate our verbal reasoning [32]. For example, planning an activity helps us to determine an action plan or split a task into smaller tasks with priorities.

Based on the widespread consensus, executive functions are parted into three basic sections: [33] [34]:

1. **Updating**

Closely connected with working memory are operations that hold, update, and mentally work with information. Said differently defending information from distraction and having it in immediately "ready to use" state for work.

2. **Shifting**

Commonly referred to as task-switching, which assists in switching between activities, do multiple tasks at once or by the named executive function is shifting between mental sets.

3. **Inhibition**

Allowing one's ability to make a decision not based on the basic external stimuli and instincts but rather overriding those emotions, attention, or behavior and do what is more appropriate. This is, when trained, what sets us apart from unintelligent creatures, which react impulsively to stimulus, and slightly gives us the "power" of deciding by choice.

Closely related to executive functions and thinking is one another notion called creativity, which falls under working memory. It is the ability to create new thoughts and ideas that improves our quality of life by producing something new, original, unexpected, and at the same time, beneficial.

3 Analysis of selected applications

There are many applications of similar types of cognitive training. Most of them are available for free, but much of the content is blocked. These parts are unlocked by paying a premium account or monthly payment. That is true for almost all popular applications with the number of downloads above 1 million. For the rest of the apps, there are often ads that can be removed by paying a smaller sum of units of euros. Below are the results of an analysis of selected popular apps available in the Google Play and App Store.

3.1 Lumosity

According to statistics, Lumosity is the most popular application for brain training. It has been available for download since 2007 and has since been registered with over 85 million users from 182 countries. The application is freely available for download on Desktop, Android and iOS (see image 3.1) and therefore has over 35 million downloads on both Android and iOS mobile devices. [35]

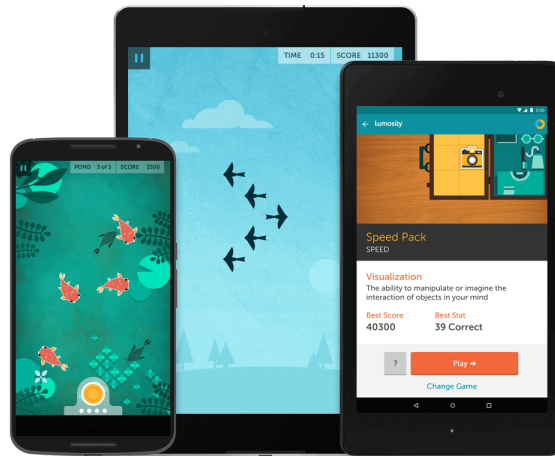


Figure 3.1: Detail on Lumosity mobile application [36]

Every day three different mini-games are offered for free training. Each of the available games tests a different part of the cognitive function, so two

games in one day do not repeat training for the same cognitive function. The user can access all games (50+) while viewing more statistics from the training progress by purchasing a premium account. The premium account is paid every month, and the price ranges between 5.6-13.6 EUR.

3.2 CogniFit

CogniFit is a healthcare company founded in 1999 focused on the assessment and improvement of cognitive health. Accurate data on the number of users or statistics are not available, but the number of users using the programs offered is in the millions. The company also offers estimates for cognitive abilities or training programs for target groups. [37]

As with Lumosity, this mobile app offers three different games every day. In the case of playing multiple games or repeats on a given day, it requires a monthly payment for a premium account, which is priced at between 4.2-22.99 EUR. There is also an option to purchase a targeted program evaluation for 19.95 EUR (e.g., dyslexia test, Parkinson's disease test).

3.3 Peak

Peak application is based only on iOS and Android devices. Detailed statistics or app information is not publicly available, so only Google Play downloads over 10 million can be compared. Visualization of app contents can be seen on image 3.2.

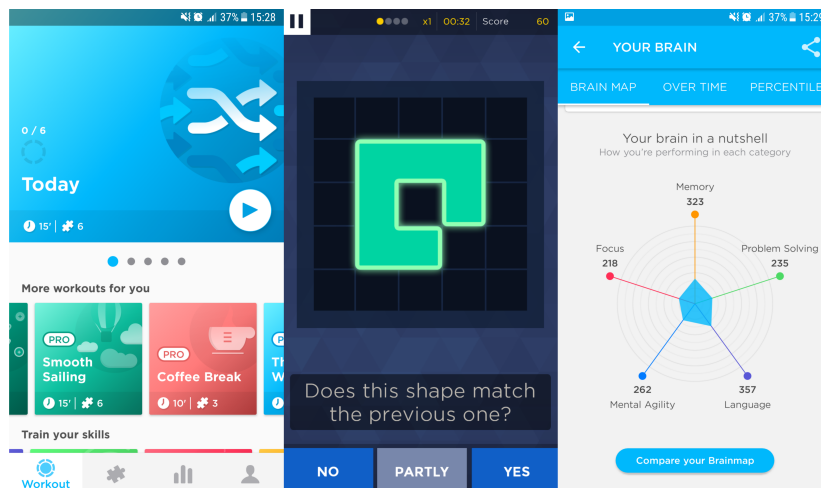


Figure 3.2: Insight into Peak application

Unlike other apps, it offers free 4-6 mini-games exactly 24 hours after the last game last day. As with the previous ones, purchasing a premium account allows users to unlock different goals, customized workouts, advanced statistics, access to all games, and the ability to compare with other users. Price per month ranges from 1-2 EUR. However, there is the possibility of a one-time purchase for a lifetime for 22.99 EUR. Currently, there is only one advanced specialized training focused on the memory test.

3.4 Elevate

Elevate application is designed mainly for improving communication and analytical skills. It was launched in 2014 and has since been downloaded by more than 25 million users on Google Play and the App Store. Available on mobile devices only. [38]

Most mini-games are based on reading and writing only in English. Three free games are available every day to unlock all 35+ games. Customization options, detailed training reports, and study materials to improve cognitive abilities are required to pay 42.55 EUR per year for a "PRO" account.

3.5 NeuroNation

More than 12 million users are using NeuroNation applications, making it one of Europe's leading providers of brain training applications. Established in 2011, when it introduced online courses directly on their website and then in 2014 launched the App Store and Google Play mobile app. [39]



Figure 3.3: Neuronation application

NeuroNation used a different approach, namely the possibility of playing eight free games available. However, they encourage the user to play a "session" of 4 recommended games focused on one cognitive aspect. An individual difference is the evaluation of user statistics, which only takes place after eight sessions. Compared to other games, they let users choose their games for their most appropriate training and do not leave them random games for one day. Of course, a premium account is offered, which costs 3-18.18 EUR for one month. With a premium account, all games (27) are unlocked, with a total of 250 levels and a personalized training experience based on the user's characteristics. Their focus on cleaner graphical design is portrayed in the picture 3.3.

3.6 Mensa Brain Training

The last application shortly named "Mensa" is one of frequently recommended brain training applications along with mentioned applications above. Its difference is that it is solely for iOS devices. The number of users is unknown as App Store, nor the official Mensa application website, does not provide any source of information about the number of downloads or active users. The launch started in 2014, and it is "developed by industry experts and accredited by the definitive high IQ society, Mensa" [40].

Unlike the other application, this one requires the user to create an account and log in before using it. In the first seven days, the account is in "free trial" mode, and after the trial runs out, the user needs to subscribe to use the application. The user has a daily option to check MBI (Mensa brain index) by playing three random free games. This index then counts towards statistics, and it is used to compare with the progress of other users. Another option is to daily play six free games, which are supposed only to train the user's brain. With subscription ranging from 0.65-1.93 EUR for one month, the user can access another two daily games and progress tracking mentioned above. Without the doubts, the application is containing some problems with not displaying special fonts or necessity subscribe to sending special offers and new releases to the user's email when first logging in.

3.7 Analysis summary

The basic details of the described applications above are summarized into the table 3.1. Sorting applications by popularity is made by choosing a suitable column, which is the number of downloads. The mentioned table is

already sorted by this column and thus can be perceived that from this order, less popular applications are more likely to produce advanced techniques to engage more people.

	Downloads	Availability	Notes
Lumosity	35M+	Web, Android, iOS	None
Elevate	25M+	Android, iOS	None
NeuroNation	12M+	Web, Android, iOS	None
Peak	10M+	Android, iOS	Advanced specialized memory test training (7.28 EUR)
CogniFit	~1M+	Web, Android, iOS	Targeted specialized training for testing various dysfunctions (19.95 EUR for each)
Mensa	Unknown	iOS	7 days free trial, after that needs a subscription to use

Table 3.1: Summary of basic information from the analysis

The second table 3.2 is focused on the pricing and feature aspect. Sorting by the number of downloads is staying the same for this table. From there can be noticed that every single element has some premium version that unlocks more advantages. Choosing from the bottom range of premium pricing where possible, it can be seen that Lumosity is the most expensive with 5.6 EUR, and Peak is the second cheapest with 1 EUR, yet both provide the same premium benefits. Possibly can be said that minimum payment differences are higher based on app popularity.

	Free features	Premium (Price / Month)	Premium Benefits
Lumosity	Daily 3 games training	5.6 - 13.6 EUR	Unlock All Games, Detailed Statistics, Unlock Achievements, Compare with Other Users
Elevate	Daily 3 games training	~3.51 EUR	Unlock All Games, Training Customization Options, Detailed Statistics and News, Study Materials
NeuroNation	Unlimited training with 8 Games	3 - 18.18 EUR	Unlock All Games, Choose Personalized Workout
Peak	Daily 4-6 games training	1 - 2 EUR (22.99 EUR forever)	Unlock All Games, Detailed Statistics, Unlock Achievements, Compare with other users
CogniFit	Daily 3 games training	4.2 - 22.99 EUR	Unlock All Games, Unlock Statistics
Mensa	Daily 6 games training, Once a day 3 games MBI check	0.65 - 1.93 EUR	Unlock 2 more daily games, Unlock statistics, Compare with other users

Table 3.2: Summary of functions and prices from the analysis

Nevertheless, not every mentioned application is focused on every cognitive skill, and some of them are oriented for a more narrow specialization.

Therefore it cannot be said with absolute certainty that one application is generally superior over the others. To see the differences between applications and their primary focus, look at the table 3.3.

	Lumosity	Elevate	NeuroNation	Peak	CogniFit	Mensa
Memory	✓	✗	✓	✓	✓	✓
Attention	✓	✗	✓	✓	✓	✓
Visual-Spatial processing	✓	✗	✗	✗	✓	✓
Language and speech skills	✓	✓	✗	✓	✗	✗
Executive functions	✓	✓	✓	✓	✓	✓

Table 3.3: Apps' focus on cognitive ability

By reviewing the apps, some interesting and important facts were found. Analyzed apps are top-rated within cognitive training apps, though they have mostly been developed by companies that consist of teams of specialists in the field. As a result, these apps often aim for some earning by offering premium accounts to the users, which gives users benefits over non-premium users. Another drawback is unsupported Czech language in any of these apps, which is not much of an issue for Czech younger adults, but it brings difficulties for Czech older adults to use these apps directed at them. Creating localization for each language, of course, is not worth it because of disadvantageous expenses, so companies do not much bother and make localization only for most spoken languages.

Notwithstanding, it is one of the future priorities for the author to give easier access to cognitive training app for older people of the same nationality as the author altogether with one or more widespread world languages. Also, the whole app is planned to be entirely free to any user without added fees. With older adults in mind, the size and readability of fonts is important yet very neglected topic. Even though another future priority will be implementing switchable font sizes for older adults, previously analyzed apps do not have this functionality, cause it presumably can make a considerable implementation problem for different device screen sizes and resolutions.

4 Games functionality theory and scenarios design

The leading part of the app is going to be games that are designed to be repeatably playable. To distinguish between a user's cognitive differences, separating them into groups of the same level of cognitive ability should be done. Therefore dynamic difficulty level is going to be implemented based on the correct or incorrect plays. Theoretically, the difficulty will range from 1 to 10, where one is the easiest, and ten is the hardest. Each user is going to start from the easiest difficulty dynamically progressing to his optimal difficulty. The difficulty with which a player ends one game is the difficulty with which the game will be started the next time. The games should be ended by a consistent limiting method like a number of played steps in-game or after time runs out. Unfortunately, a number of played steps in games can vary, and hence it is not entirely consistent for each game. So, for now, a time of sixty seconds has been decided as default game end criteria.

Individual game scenarios topics are based on cognitive functions covered in section 2, yet there is a difference in executive functions skills, which consist of an extensive range of functions that are hard to specify. Also, a lot of them are already partly included in the other cognitive skills. For that reason, the last group will be aimed in another direction, namely, math, where games will be focused on calculation with some parts of executive functions, e.g., planning activities, solving problems, or do more activities at once. Game scenarios by the groups are described below.

4.1 Memory

4.1.1 Box lookup

Functionality description

The player has to remember the position of certain objects, and after they are hidden, he has to act to show them afterward. For a better idea of the previous sentence you can see the design in the picture 4.1. Training is specified for short term memory function.

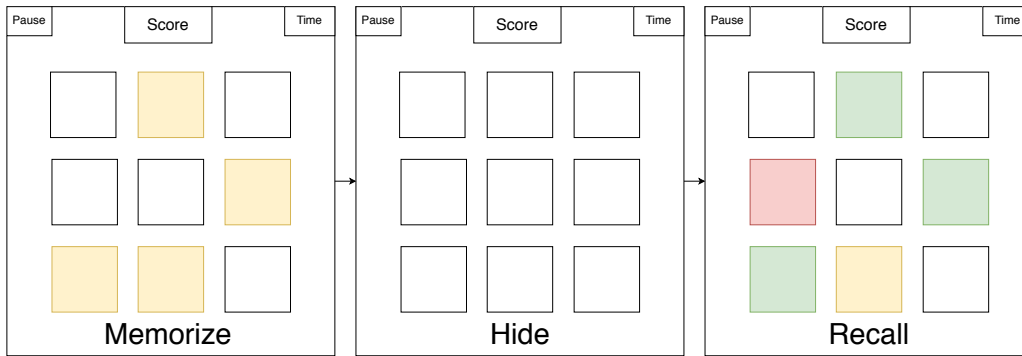


Figure 4.1: Sketch of Box lookup game design

Scenario

For one second, few fields of a different color, which the player has to remember, are going to be displayed on the square field defined by n^2 size. The minimal number of a square field is $n = 2$. After one second passes, the colored fields dissolve back into the normal field. After that player is allowed to touch the screen, and the objective is to tap all fields he remembered. If he fails to tap the correct field, the game restarts with less colored fields. The player has to complete as many square grids as possible in the given time.

Difficulty

The number of colored fields to remember is with each correctly completed grid incremented by one and decreased by one otherwise. Difficulty increases when the player has the number of colored fields higher than half of the size of the square field a decreases if the player has the number of colored fields lower than half of the size of the previous square field. Consequently, the difficulty is managed by the size of the square grid and the number of colored fields to remember.

Score

The score is going to be computed with every correct field tapped, current difficulty, and the current maximum number of colored fields.

4.1.2 Dangerous path

Functionality description

Similarly, like the previous memory game scenario, the player has to remember the position of objects, however, with the difference that he must distinguish them between good objects and other bad objects. After hiding, he has to create a path between good objects, while not touching the bad objects. Training is focused on memory, path planning, and visual perception.

Scenario

From a squared field defined by n^2 size with minimum $n = 3$, two good green fields, and a few, based on the difficulty, bad red fields will show. After one second, good and bad fields hide as normal ones, and the player is allowed to touch the screen. By dragging or touching pieces one by one, he has to "draw" path from the first good field to the second good field. While drawing, it is necessary to do not touch any bad fields, but if it happens, the game restarts and lowers the number of bad fields. In the given time, the player has to complete as many paths as possible.

Difficulty

The number of good fields does not change with difficulty. Only the number of bad fields increases or decreases by one based on correct or incorrect plays. With greater difficulty, good points are further away from each other. Difficulty increases when the number of bad fields plus two (for good fields) overflows half of the size of the square field and decreases when the same number is lower than half of the size of the previous square field. By summarizing difficulty changes the size of the square grid, the number of shown bad fields, and by prolonging the distance between good fields.

Score

The score is accumulated with the length of the correct path drawn, current difficulty, and the number of correctly dodged bad fields.

4.2 Attention

4.2.1 Flying frog

Functionality description

The idea for this game is based on the mobile game Flappy Bird [41], where the point of the game is to fly with a bird trying to score points by flying through obstacles. The fundamental concept of the game stays unchanged, yet there are alternations for the game to be training cognitive function. The player is facing two markers showing the right and wrong path. Based on the player's reaction, it is expected he moves to the right position before its too late. Game design proposal can be seen in the picture 4.2, where one of the pictures is view on previously created game by the author of the diploma thesis. This game should train the individual's reaction speed and selective attention.

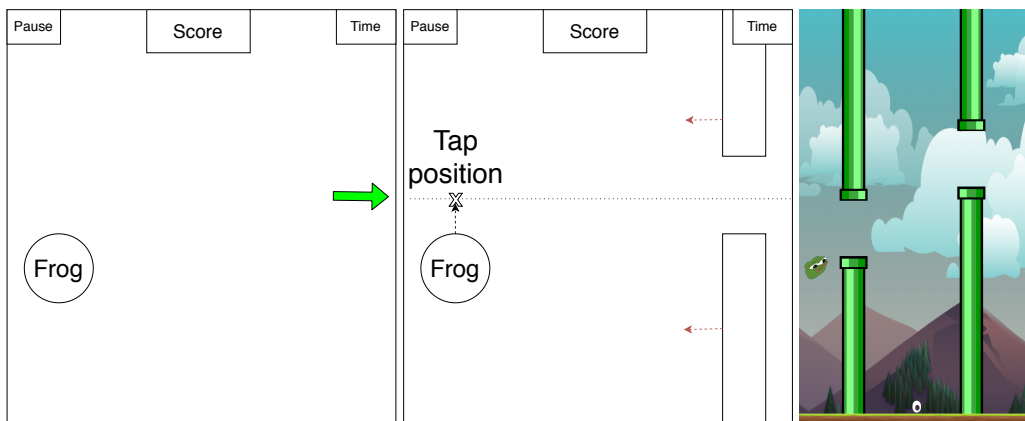


Figure 4.2: Sketch of Flying frog game design with one view of close to finished look

Scenario

On the left side of the screen, the player controls the character by tapping on the screen. The character only moves only on the Y-axis. During the gameplay, on the right side of the screen, green and red arrows will be shown, pointing to the right or wrong path. After a while, based on the difficulty, an obstacle appears moving towards the player. The player has to move out of the red arrow position or move to the green arrow position. When he fails to do so, obstacles are removed, and character respawned with a time to regenerate. After regeneration time passes, the process starts again.

Difficulty

Difficulty changes the time to react and the speed of the obstacles. With higher difficulty, the time to react is lower, and also the speed of the obstacles is faster. The character should at all times have the possibility to fly from the minimum bottom of the screen to the maximum top of the screen, thus modifying the speed of the character is necessary with modifying the difficulty. The difficulty is increased by one when the player goes thirty seconds without failing and decreases by one if the player fails to fly through the obstacle two times in a row.

Score

Scoring is added every time the player flies through an obstacle, adding current difficulty and survived time without fail.

4.2.2 Two cars

Functionality description

This training comes from HTML/mobile game Two Cars [42], which does great use of the game concept for cognitive training as this should train attention, multitasking, and speed processing of the player. By simultaneously moving two objects (cars), the player has to dodge bad objects coming in his way and collect good objects coming in his way.

Scenario

Two cars are positioned on the bottom of the screen, and the player can control both of them at the same time by dragging or tapping. Both cars have only two road lanes and can move only on the X-axis, while never colliding with each other or moving behind the centre of the screen. On the two roads, randomly bad square and good circle objects spawn periodically from the top of the screen, moving towards the bottom of the screen. Both of the cars have to dodge harmful square objects and pick up the good ones. Failing any criteria despawns the objects and resets the cars with time to regenerate. After the time passes, objects start to spawn again until end conditions are met.

Difficulty

The difficulty is increased by one when the player survived without failure for thirty seconds and decreases when he fails two times in a row under the

first five seconds of game resetting. Higher difficulty accelerates the speed of the objects coming in the way of cars and slows otherwise. The only thing that changes with difficulty is the speed of the objects.

Score

The score is added any time the car collects a good object adding the current difficulty level, survived time without fail, and the number of collected good points in a row.

4.3 Visual-Spatial

4.3.1 Follow the owl

Functionality description

This type of training game is based on the Corsi block-tapping test [43], which involved nine blocks while one person (experimenter) taps a sequence of blocks, and after that, the second person (participant) as to repeat the tapping in the same order. The sequence length is expanded every time participant does the correct sequence repeats. Corsi block-tapping test is fully shown in the picture 4.3, which actually presents the game itself. This test assesses the visual-spatial short-term working memory.

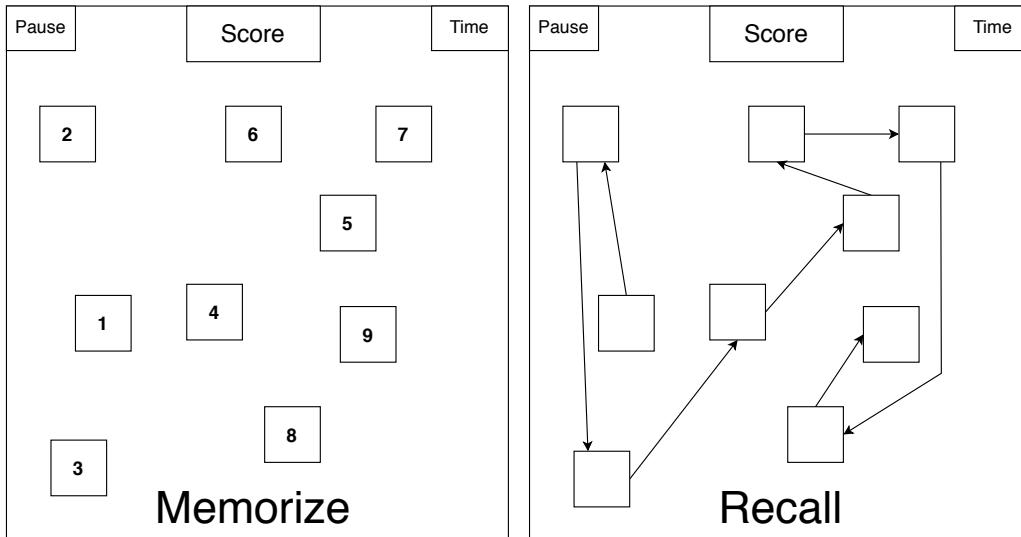


Figure 4.3: Sketch of the Corsi block-tapping test implemented as Follow the owl game design

Scenario

On the screen, ten randomly positioned boxes will be shown. After one second, an image of owl starts to gradually show in the boxes where the minimum number n of the times shown is $n = 3$. After the last one is shown, the screen notifies the user that he can start tapping. The player has to tap the boxes following the sequence previously presented by displaying owl. Clicking a wrong box stops the sequence and resets the current grid and starts to show owl again.

Difficulty

The difficulty arises when the number of correctly guessed sequences is equal to the current difficulty number and decreases when the same number is equal to minus two. The guessed sequence is lowered by one every time the player performs an incorrect sequence and increased by one every time he duplicates the sequence by tapping. When the difficulty is changed, the current guessed sequence is reset to zero. Difficulty changes the time of displayed owl time, and the number of times owl is shown n . Higher difficulty lowers the time displayed and increments the number of times shown.

Score

The score is added any time player taps the box in the sequence. Added are current difficulty, the number of correctly guessed sequences, and the position number of the box in the sequence.

4.3.2 Hiding mosquito

Functionality description

In the same way as the Follow the owl game, this training game is based on the Corsi block-tapping test [43]. The player has to remember the sequence of the directions and, in his mind, create an imaginary path based on the directions. After the presented sequence, he picks the box, where the displayed object ended, based on directions. This game trains visual-spatial processing.

Scenario

On the square grid of boxes size, n^2 with minimum $n = 3$ one box includes displayed mosquito. After one second, the mosquito disappears, and arrows

start to show one of the eight directions in the center of the screen. The player has to remember and imagine a path from the mosquito's starting box. After the sequence of directions end, the player is notified, and he has to tap the box where the mosquito secretly ended based on the directions. Clicking the wrong box resets the grid and mosquito starting position.

Difficulty

The difficulty changes the time of the displayed direction arrow and the size of the square grid of boxes. Higher difficulty means lower time to display arrow and greater size of a square grid. The difficulty is increased when the number of correct guesses is equal to the current difficulty number and decreases when the same number is equal to minus two. The guessed number is lowered by one every time the player taps incorrect end mosquito position and increases otherwise.

Score

For every correctly guessed final position of mosquito, the score is added. The score includes current difficulty, the number of guessed mosquito's end positions in a row, and the number of maximum existing directions.

4.4 Language and speech

4.4.1 Color confusion

Functionality description

The Stroop Color and Word Test is the basis for this game [44]. The test shows that naming the font color of the printed word is quicker when the word meaning and the font color are equal. The invented game idea is to show written color with random font color where the player has a few options to answer. The player has to answer the written color presented as the font color of the shown main element. Game designs suitable for this type of training are proposed in the picture 4.4. Trained cognitive abilities include language perception, selective attention, and processing speed.

Scenario

In the middle of the screen, cards stacked on top of each other are presented with the first card showing the written color in random font color. At the bottom of the screen, the player has n options based on the difficulty, but

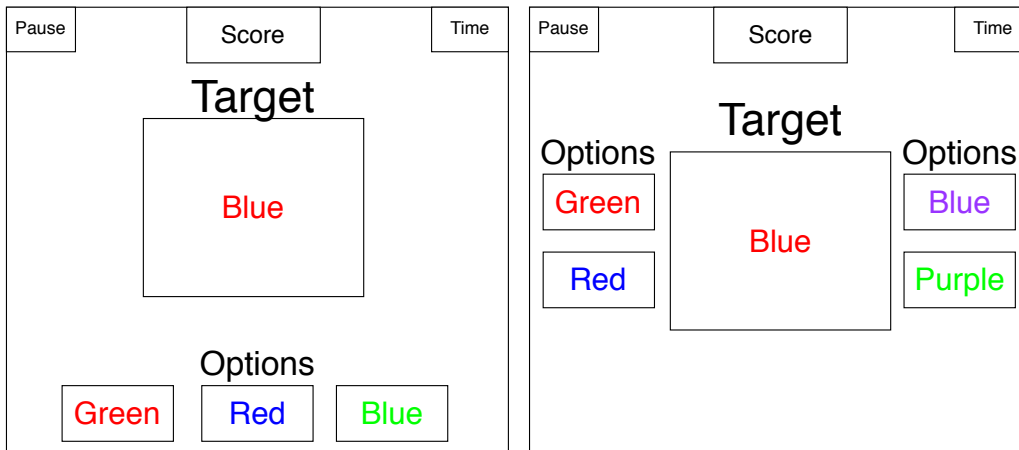


Figure 4.4: Sketch of 2 possible Color confusion game designs

the minimum is $n = 2$. The options are written colors, all also with a randomized unique font color from each option. The player taps written color in the options section that is identical to the font color presented in the middle card. If wrong, the card is thrown away, and the game continues.

Difficulty

Greater difficulty adds more color palette, which with the lowest difficulty, starts at two primary colors only (red and blue). Also, more options are presented with higher difficulty. An increase in difficulty appears after the player correctly guesses current difficulty times two colors in a row and decreases when two incorrect guesses appear in a row.

Score

The score is added every time player taps the correct option. The score adds current difficulty and the number of correct guesses in a row.

4.5 Math

4.5.1 Bubbles

Functionality description

With the use of present numbers, the player has to add numbers for the resulting displayed simple sum. As math is a summary of visual-spatial representations, spatial memory, working memory, selective attention, plan-

ning, and reasoning skills, these are all expected to improve as the training outcome.

Scenario

From the top of the screen, bubbles containing a number from 1 to 9 falls to the "floor". At the bottom of the screen, under the "floor", the number to calculate is displayed. The number to calculate is in the range from 10 to 30. The player taps the requisite bubbles with numbers to add until they are equal to the final sum. If the final sum of the numbers overflows the displayed result, a new number to calculate is displayed, and the game continues. In the game limit, the player should match as many results as possible.

Difficulty

The difficulty changes the speed of numbers falling from the top of the screen. Also, the number to calculate is going to be higher with upper difficulty. Difficulty increases when the number of calculated results equals the current difficulty times three. It decreases when the same number is equal to the previous difficulty minus two. The number of calculated results is increased by one for each correctly calculated result and decreased by one otherwise.

Score

The score is added every time the correct result is calculated. The score includes current difficulty, the number of correctly calculated results in a row, and the number of numbers used to calculate the result.

4.5.2 Estimation

Functionality description

The player is shown a collection of a few numbers. His task is to estimate the result of the sum of the numbers. Afterward, he chooses the position of estimation on the numerical axis. Vision of game look can be seen in the picture 4.5. The goal is to train the accuracy of the calculations with training all the cognitive functions included in the process.

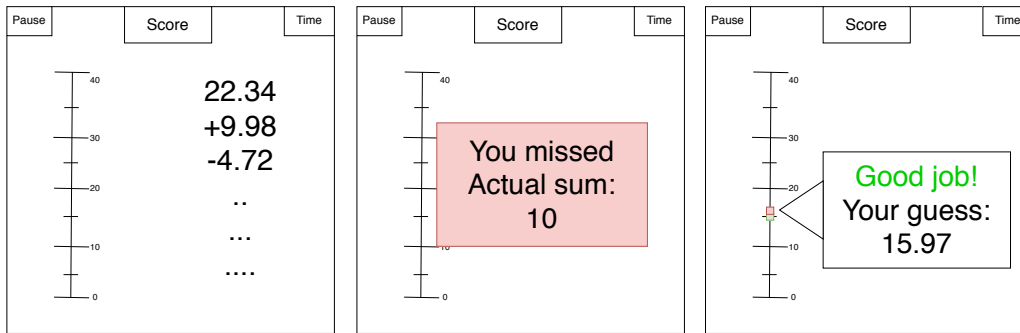


Figure 4.5: Sketch of Estimation game design

Scenario

The player is presented a few numbers, e.g., 1, 2, 3.5 on the left-center side of the screen. From top to bottom of the screen on the right side is the numerical axis from any number range, e.g., 0-10. The numerical axis has more highlighted numbers of interests, e.g., each even number. Mentally summing the numbers equals 6.5, thus on the displayed axis by dragging or tapping, the player chooses where the calculation belongs. The tips of estimation are acknowledged in 5% range around the dropped tip. If the user fails to tap in the estimated range correctly, the game continues.

Difficulty

The difficulty changes the number of numbers to sum and their decimal, altogether with a more zoomed-in numerical axis. Difficulty increases by one when the correct estimations in a row are equal to the current difficulty times two and decrease by one when there are two incorrect estimations in a row.

Score

The score is added every time estimation is hit in the range on the axis. Added to the score is the current difficulty, the number of correct guesses in a row, the number of numbers calculated, and how many of the presented numbers included decimals.

4.6 Games for future expansion

4.6.1 Language and Speech - Words

Functionality description

The player has to write as many words in his language with the number of letters equal strictly to the given length. This game trains language vocabulary and long term memory. Unfortunately, complications arise with not having access to large dictionaries in all languages that the game supports.

Scenario

The player is presented a m numbers with size n , based on the difficulty, with a minimum $n = 3$. He then has to write as many linguistically correct words as possible in the language defined by chosen game language in-game options starting on a random letter. Every word is entered by clicking the OK button. When the entered word is not present in the provided vocabulary, the game continues without stopping. This game is not suitable for completing in the default time limit as an end condition, and end condition is then met by completing all word length options to the upper limit given by current difficulty. The time limit is still present to provide failed end condition.

Difficulty

Higher difficulty adds more word length possibilities, m that has to be written at the same time. For example, when at the lowest difficulty, the player is expected to write only words of length three. With higher difficulty, the player must write words with length, three, four, and five, with self-explanatory n is diverse for each word length option. Difficulty increases if the game is completed with word length upper limit end condition and is decreased if the number of correctly entered words is less than half of the word length possibilities times upper limit.

Score

The player is rewarded with a bonus score for knowing three words in each word length option and rewarded less afterwards. Bonus reward should endorse the player to finish more word length options to gain a bonus score than to write many words just for one length word possibility. The score is added after every correct word and includes current difficulty, number

of words within the current word length option, and bonus score for three words in word length option if possible.

4.6.2 Attention - Flanker test

Functionality description

Based on the Eriksen flanker task [45], where the user needs to respond to the target stimuli while suppressing other unimportant responses. The target stimuli (e.g. middle element) is flanked by other stimuli, which can be compatible (Congruent), incompatible (Incongruent) or can be neutral (Neutral) to the target as seen in the picture 4.6. The reaction time to process information with the incongruent stimuli has been found as significantly greater than incongruent stimuli and is defined as the Flanker effect. The game should train the individual's reaction speed and selective attention.

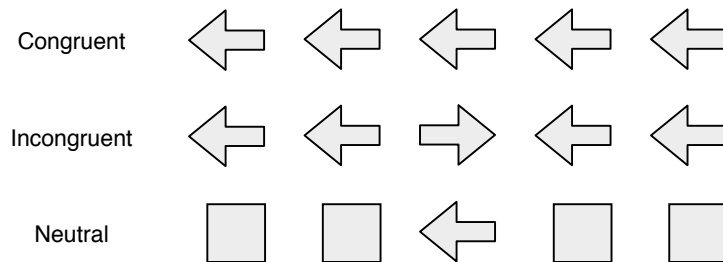


Figure 4.6: Eriksen flanker task visualisation

Scenario

The screen consists of one "anchor point" which is an inconsistent place near the target stimuli. This point is shown the whole game and the player should focus his sight at this point when nothing else is visible. The stimuli are visualised as some pointy/direction-showing object. As a reaction, the stimuli is firstly hidden to the player and shows after a while. The stimuli show in a row or around the anchor point. The target stimuli can be directed at four cardinal directions north, east, south and west. The player has to swipe with his finger towards the direction the target stimuli is pointing. After the interaction, correct or wrong, the stimuli are hidden and will show another combination after a while.

Difficulty

Lower difficulty shows static stimuli with higher space between them and also gives more time to regenerate after an interaction. Higher difficulty lowers

the spacing between stimuli and also after a certain point adds random colors to the stimuli to distract the player more while giving him lesser time for regeneration after his interactions. Difficulty increases with difficulty times two correct guesses in a row within a limited time and decreases when three wrong swipes have been performed.

Score

With every correct swipe in the direction shown by target stimuli, the score sums with new values of difficulty, the number of correct swipes in a row and the reaction time to swipe. Faster reaction time adds more points to the score.

5 System design

The whole system is going to be composed of a mobile app and a server to collect and present statistics of user games. To gather data from users, they need to be marked as an advanced user and connected to the internet. While there is a possibility of connection lapses, the app should not be dependent on an unending internet connection and will only try to synchronize with any data transfer action. Advanced user is going to be defined as a logged-in user in the application using the implemented social media login, or by custom email/password login. Altogether with necessary user information email and name, the user will only send predefined games data collected by the app to the server. As this app will not necessarily need the user to log in to accumulate data from the user, the model of a primary non-logged user will have free access to the app without sending any data yet allowing full access to app functionality.

The proposed architecture model can be seen in the picture 5.1. Besides, in the picture are already defined chosen programming language mobile app will be programmed in and server, that will be used. More on this in related theory section 5.3 for programming language and section 5.2 for server choice.

5.1 Operating systems

Based on the worldwide statistics, most of the market share own Android with 72.26%, and second is iOS with 27.03% to March 2020 [46]. A combined percentage of Android and iOS market share is enough to claim that other OS's are not going to be supported by default with this app due to the nature of specific developing difficulty for other OS. Primarily focus on finishing the working app on Android will be done, and after that, if the development allows cross-platform possibility, the app could also be released to other platforms, e.g., iOS, PC, HTML5.

Choosing the minimal Android API level for the app plays a significant role, which on Google Play Store determines on which user's devices can an app be installed. In the image 5.2 is shown the relative number for platform version information between Android's users. The data are coming from the official Google Android development team and "represent all of the active devices during a 7-day period ending on April 10, 2020" [47]. Altogether with other statistics showing Android version worldwide market share [48], this

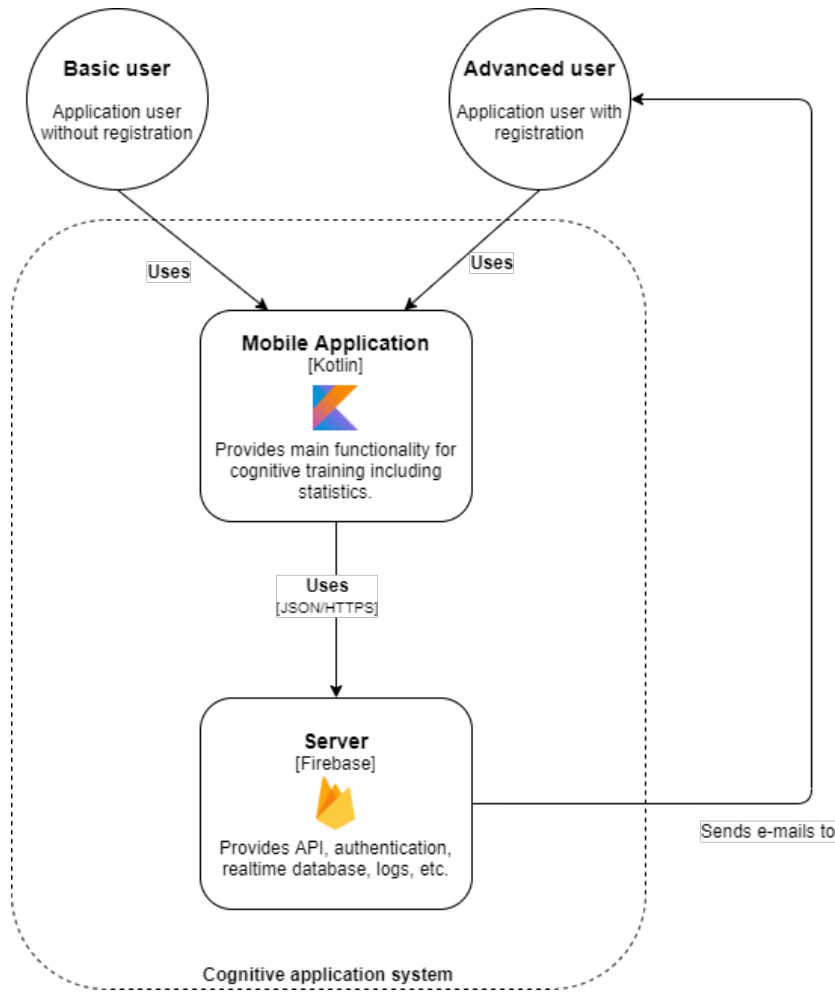


Figure 5.1: Prototype of system architecture

this app should support minimum API 19, also called Android KitKat 4.4, which is cumulatively supported by around 98.1% Android’s users. Trying to lower the minimum API to 15 can result in the necessity to deny using advanced programming functions that are not supported by the lowest API, and thus slowing down development time. Also, users with lower Android versions are not going to be a massive base of users for this app to consider them.

5.2 Server and database model

As one of the key aspects of this mobile app is collecting statistics from playable games, the server will be used for this purpose. The server should mainly allow to receive data from the users and send back requested ones

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

Figure 5.2: Android Studio’s population distribution over Android versions

for presenting the required information. This leads to narrowing down the importance of server services to act only as a database server. The first possibility, of course, is making an own server with a database, which would communicate with custom API. The server would contain only the most essential functions for the app’s primary purposes and could be scaled by the current needs. The problem is that developing a custom server would consume a high amount of author’s time and resources, which is making this option suboptimal to achieve in arranged deadline. The better and more preferable choice is then using an existing solution.

Based on the previous author’s experience and an enormous amount of offerable services, a Firebase platform will be used as a server. Firebase is mobile and web application, also known as BaaS (backend as a service), backed by Google since 2014, offering functions that free developers from writing own API or managing servers and allowing more focus on the users. Additionally, to this year, there are 20 products divided into three categories product development, product quality, and product growth that Firebase provides and many more pre-packaged solutions that developers can use. [49]

5.2.1 Database

Currently to May 2020, Firebase is ranked as the seventh most popular database based on the number of searches on Google Trends, and the popularity of the overall database grew the most in the last five years [50]. High demanding service for this project is going to be Firebase Realtime Database or Firebase Cloud Firestore, which are Firebase's implementations of cloud-hosted document-oriented databases using JSON format. Document stores do not need to have a uniform structure, thus allowing a more diverse approach of data collecting for different users. Records also can have nested values or can contain more values retrospectively arrays. This flexible approach is easier to work with without excessive knowledge of another programming language, and is prone to useful scalability, where adding more functions and games to this project, can unexpectedly change the growth of the application and need for mentioned scaling.

The pricing for the server and its functions differ into two plans free Spark plan and price scaling Blaze plan. As the app developed by this thesis is not going to gain any profit, a free Spark plan is planned to be used. The details of the Spark plan are fitting and acceptable for low-middle expectations, and if currently unexpected growth in popularity happens, switching to Blaze plan is relatively easy, and the price will scale based on usage of functions. To break down the crucial restrictions of a free plan, here is the list [51]:

- **Cloud Firestore**

- Stored data - 1GiB
- Document writes - 20,000/day
- Document deletes - 20,000/day
- Document reads - 50,000/day

- **Authentication services**

- Phone authentication - 10,000/month
- Other authentication services - free

- **Realtime Database**

- Simultaneous connections - 100
- GB stored - 1GB
- GB downloaded - 10GB/month
- single database per project only

- **Test Lab**

- Virtual Device Tests - 10 tests/day
- Physical Device Tests - 5 tests/day

The list shows only main features, yet there are other not mentioned features, most noticeable Analytics or Crashlytics, that are free and will be, with high interest, also used in development. The difference in restrictions of Cloud Firestore or Realtime Database is minimal, and for this project, Realtime Database has been chosen as database service.

Other considered options were MySQL or MongoDB, which were excluded from the choices because they lack other non-database services and the necessity to find hosting.

5.2.2 Database model

As mentioned in the section above, the database is being document-oriented. Creating a database model for a document-oriented database is probably more of a challenging task because it is not necessary to have table relations to everything. It is also essential to take into consideration which information is required to accumulate for future statistics presentation. Realizing the basic needs of the cognitive app defines two main collections, Users and Games, that further expand other mappings. The model with schema mappings can be seen in image 5.3.

Users

This collection contains information about users and their played games. Every user is described only by username and age. For each game, the user saves achieved scores, his difficulty level, and how many times he played the game. The model does not need to contain any contact attribute because saving data to the database is allowed only for registered users. Registered users use Firebase Authentication to log in, which identifies a user through trusted identity providers and allows access to necessary user information of those providers and so the email is firmly connected to the unique user id. Also, in early development, the app will not allow Phone, Game Center, Play Games, and other unusual sign-in methods.

- **user_id**: User unique identifier created by Firebase UUID generation method.
- **user_name**: User name or nickname.

- **game_name**: Game unique name identifier.
- **user_age**: Current user age, used only for analysis and calculation of statistics.
- **best_scores**: Descending array from best of top ten scores achieved in-game.
- **scores_in_week**: Hashmap of weekdays as key with top scores achieved that day as value. Each day is overwritten by the same day of the next week. Used for calculating user game progression.
- **current_difficulty**: Current level of game difficulty, in which user is. Used for calculating difficulty and cumulative user statistics.
- **times_played**: Number of times the user played a game. Used for calculating the popularity of games.

Games

This collection contains calculated scores data on all games. The scores are used for presenting statistical information to any user. Each game is defined by a unique game name that is identical to game names saved in **Users** collections. Data inside this collection are going to be procedurally calculated each day from games of all users. That way, every user will have access to the same statistical data for comparison, achievements, and own improvement correlated to global scores.

- **game_name**: Game unique name identifier.
- **the_best_score**: Maximum score in game achieved by any user, that is used for normalizing other values.
- **average_score**: Average score in game achieved by all users.
- **score_by_age**: Array of age ranges containing the **the_best_score** and **average_score** achieved in those groups.
- **score_by_difficulty**: Array of game difficulties containing the **the_best_score** and **average_score** achieved by the difficulty group.

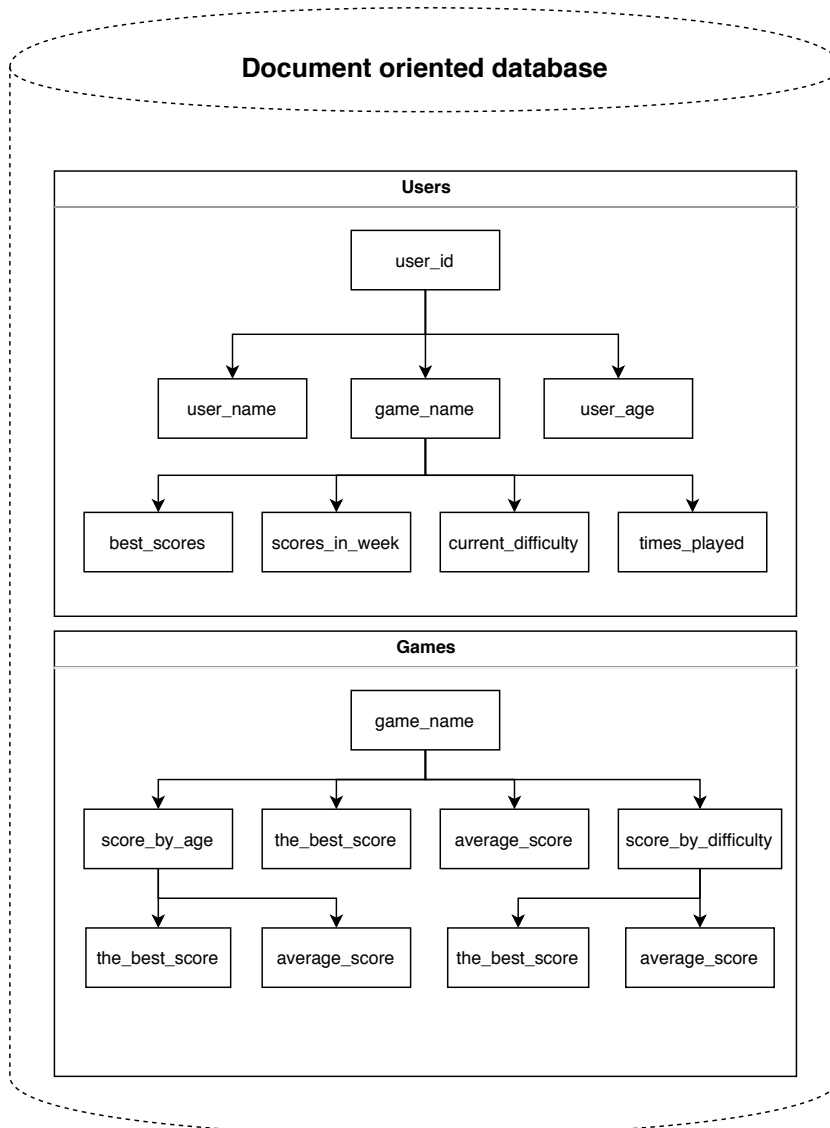


Figure 5.3: Designed database model with mappings

5.3 Programming language and framework

For the creation of mobile apps focused on training games, development frameworks, and engines are providing handy resources. A framework is generally speaking a collection of code packed into the library written in a specific language or even complex system of scripts, tools, and workflows for a specific platform. Overall, using development frameworks leaves more space to concentrate on the design and user experience rather than reinventing the wheel. A useful framework is mostly defined by the possibility to distribute to as many platforms as possible, has a unified API, and proper

documentation with an active community.

The app in this thesis is going to be programmed in Kotlin programming language. As of 2019, Kotlin has been chosen by Google as the preferred programming language for Android development [52]. Kotlin is interoperable with Java, supports multiplatform development, supports code safety, has a vast community, and its features allow to reduce boilerplate code. The popularity of Kotlin language has arisen since the last few years, and in last year of Stack Overflow survey, it ranks as fourth most-loved language [53].

The choice of the framework has been designated on LibGDX, which is one of the most widely used open-source Java game frameworks. Unlike the most popular full-fledged game engines, it lacks a visual editor and instead works as a library that focuses on providing utilities and abstractions that ease game development. Therefore previous coding knowledge with objective programming language is necessary for sufficient work, yet the programmer has more control with lower-level coding. As LibGDX is self-explaining that 2D and 3D are supported, it is mainly used for custom 2D mobile game projects.

Other rejected choice consisted of Unity engine, which is one of the most popular Android development tool requiring very low to none previous programming experience. It is a free versatile tool that allows more access to content with a full paid version. One of the most prominent qualities is the Unity Asset Store, where developers can obtain a massive amount of assets of their need for any game. However, Unity is more focused on 3D than 2D, and with its user interface, it is possible to create a simple game with no coding. As a matter of fact, by many comparisons between frameworks, picking the best technology eventually depends on the author's taste and his own needs.

6 Implementation

Implementation is going to be developed based on the previous design chapter. Chosen programming language and framework, mentioned in section 5.3, versions used, are 1.3.72 for Kotlin and 1.9.10 for LibGDX. In section 5.1 was said that minimal SDK API level for this app should be API 19 (Android 4.4 KitKat). From the prerequisites of prominently included bundles, LibGDX framework has unspecified SDK minimum level [54] and using Firebase requires at least API level 16 (Android 4.1 Jelly Bean) with compile SDK version 28 or later [55]. Unfortunately, during the implementation of the solution, the minimal level had to be changed to API 21 (Android 5.0 Lollipop), because of the problem described further in section 7.3.1. For the management of third party libraries, the project uses Gradle dependency management and build system. The author created all diagrams and visual designs in open source online diagramming software Diagrams.

The application itself is based on the second visual idea, which can be seen in the picture 6.1 as the first idea was more of a rough sketch. Some details from the picture were recreated for a more suitable presentational or technological solution, but the core stays the same.

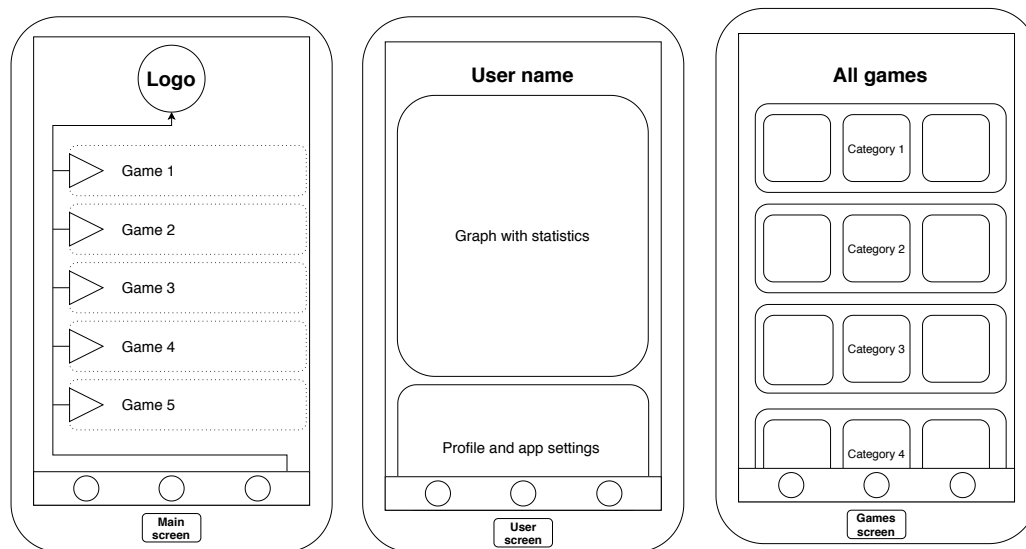


Figure 6.1: Prototype of main graphical user interface

6.1 Project structure

The tree structure shown below is a truncated version of the real project structure, but it is showing the essential heart body. LibGDX layout is structured into submodules, and for current purposes, only core and android submodule were created. The rough structure of the project is given as follows:

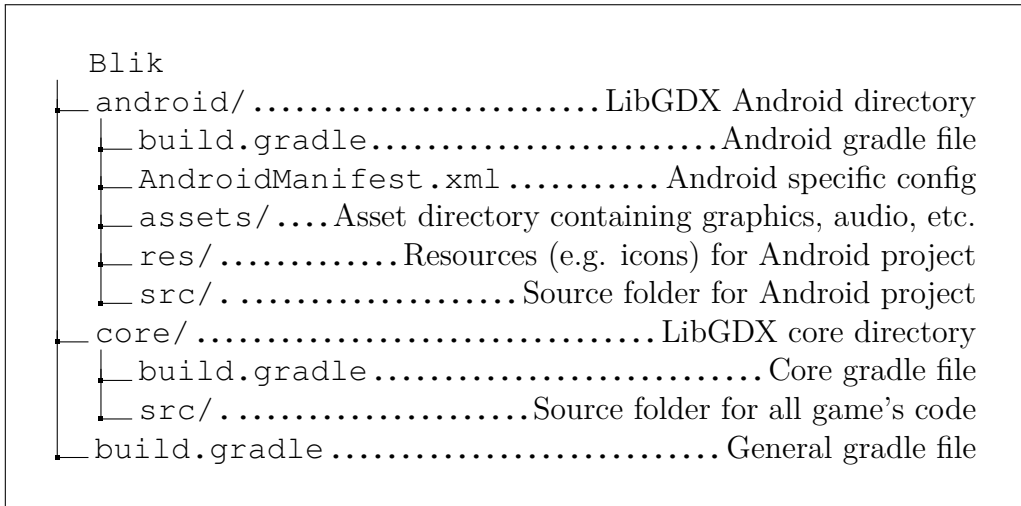


Figure 6.2: Core LibGDX layout

Taking the awareness of the android assets structure, it is fragmented into more specific directories. As the app itself contains tens of different textures, loading them all at once as one pack could lead to memory leaks which slow down mostly older devices. Memory management plays a big role and dividing assets to chunks, that are loaded together, is a way to go. Assets layout is therefore done in a way to present these chunks in directories, which can be seen in the figure 6.3.

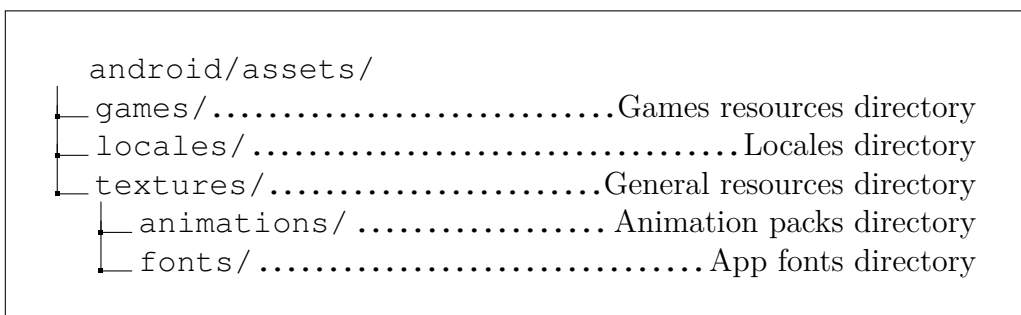


Figure 6.3: LibGDX assets folder layout

The general resources directory contains widely used textures and fonts in a skin pack, commonly used animations and app music. Games directory contains single games directories, where textures, music, sounds and physics unique for the game are present. Lastly, the locales folder contains all files that support the app with translations to other languages.

The last structure suitable to characterize is the core source directory 6.4. It is where all game's code is current which is then converted to other platforms by LibGDX interfaces. The comments for the structure below should be enough to understand what type of classes are included in each directory. Programming implementations of every class will be talked about later.

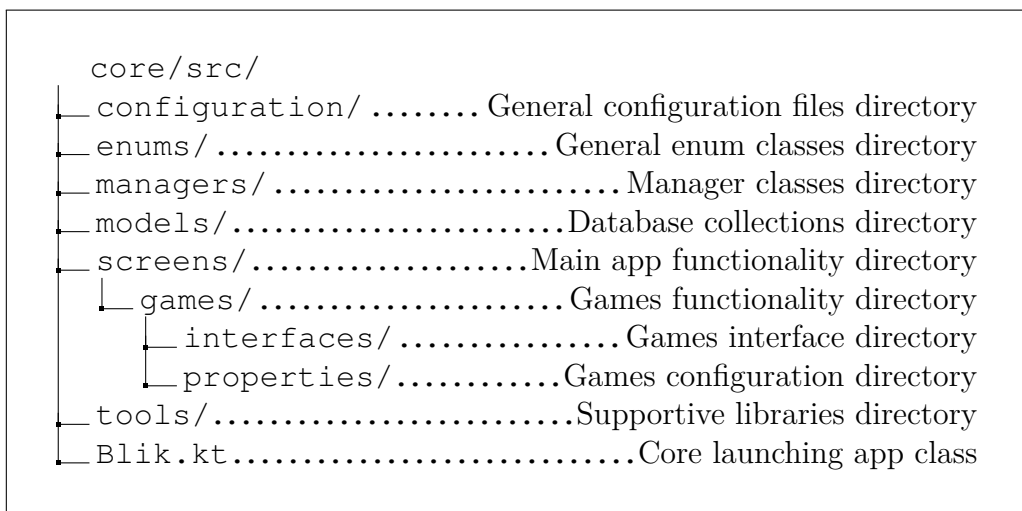


Figure 6.4: LibGDX core source folder layout

6.2 Firebase services

The Firebase platform, functioning as a mentioned server side of the project, consists of several product modules. These products, or so-called services, can be added and removed independently based on the app needs to construct the requested server functionality. Therefore in this section will describe each of the Firebase products used in this work. The versions of services used were often changed to match the recently developed ones, but the versions presented for each service is the version matching the app current release version.

With the priority number one, Firebase Realtime Database is a cloud-hosted NoSQL database, which builds the foundations of the app server.

The purpose of the database is clear, although to clarify, database stores and sync data in JSON format between users in real-time. The data ranges from user-specific information to global score statistics among all users. The version used is 19.3.1.

Another product which makes development easier is Firebase Authentication. Evident from the name, it is a service providing sign-in solution while covering multi-platform support, end-to-end identity solution, high security and more functionalities like account merging. It is thus allowing to set up a merge of, e.g. Facebook and Google account and still see it as single Firebase user with more social connections. As the application itself has implemented login logic for three different login options (Google, Facebook, custom email), this feature only required to set up handling OAuth2 client token flow with Firebase project and enabling the sign-in providers in the Firebase console and authentication system was complete without the need of implementing custom one. For the Google sign-in implementation, it is necessary also to include the Google Play authentication service. The Google Play services require to specify the SHA-1 fingerprint of the signing certificate to create OAuth2 client and API key for the app. The used version of Firebase Authentication is 19.3.2, and the version of Google Play authentication service is 18.0.0.

Lastly used product for app testing crash reports between a broader spectrum of testers is called Firebase Crashlytics. Anytime an app crashes to unknown error the logs are saved in the local device, and the next time user launches the application and is connected to the internet, Crashlytics automatically sends the crash report to the Firebase. If there are more types of unknown crashes, the Crashlytics automatically groups them into lists, which can be later prioritized depending on vital information about the crash. Whole Crashlytics works in realtime, so currently they are synchronized with Slack bot application which instantly reports any critical crashes to the author's private Slack room. Crashlytics version used in the project is 17.1.1

All those Firebase products used are presented in the web application, Firebase console, to allow a more comfortable option setting. For the authentication, chosen sign-in methods are enabled, and all logged-in users can be seen in a list with default information, where an example of user view can be seen in the picture 6.5. The full database entities are shown in a tree view based on the database model 5.3. The data is saved as key-value items, that can be changed, added or removed through the console. Any change made by the app can be seen in realtime, whereas a color highlights the changed rows for a brief second. The color of a highlighted row depends

on the action, green for insert, red for remove and yellow for an update. Crashlytics are nicely represented in the console as they show demographics of crashes along with detailed crash logs, affected devices and operating systems and app version. These functions hugely relieve the developer from the focus on side problems and with the web view, it allows to focus on the crucial development.

Identifier	Providers	Created	Signed In	User UID ↑
test@gmail.com		May 2, 2020	Jul 9, 2020	BGK6KhPDvfUnk9rVUyqwqWezBwS...
test@email.com		Jul 7, 2020	Jul 7, 2020	BKp9R80wxyYis4RMWkLcmfDEYd...
test3@gmail.com		Jul 8, 2020	Jul 20, 2020	DUMbNcZFtwVItagG1xOCYzy0qJ32 

Figure 6.5: Firebase authentication information

6.2.1 Firebase with LibGDX

In the documentation, Firebase has some code examples of how to use their products in many programming languages, including Kotlin. Unfortunately, the examples need to be implemented in the android or iOS project, which contradicts the LibGDX layout. To make cross-platform in LibGDX working, app code has to be written in the "core" folder. To make it work, LibGDX allows accessing these platform-specific APIs by defining specific implementation through standard API interface called as interface class. Although creating all those platform interfaces for each included Firebase service method would be possible, it would also be enormously time-consuming.

Time is of the essence which indicates to the use of GDX Fireapp 2.1.7 release made by a GitHub user known under name mk-5 [56]. It is LibGDX cross-platform API for Firebase, which would typically have to be made by hand. This library is still actively updated to the newest releases and has relatively consistent bug fixing. The last release was in 18.1.2020 and on the GitHub wiki consists of pretty good documentation to set up the library with each example of use. All around this is just a temporary solution, as it is not guaranteed that it will continue to have full support to future Firebase releases as only three contributors are working on it. Unluckily, some features are not implemented yet and indicated for future releases which consist of Facebook authorization. The API syntax is very similar to the official one since it works with promises API. Promises, firstly introduced as JavaScript promises, performs asynchronous processing and their advantage is to avoid callback hell (long nesting of asynchronous functions). An example of usage

of promises in code 6.2. What can be seen in the example, is the self-called best feature in GDX Fireapp chaining, which helps to remove unnecessary lines of code.

Facebook type of authorization is added as one of the option to log in into the app, and therefore, with GDX Fireapp not supporting this option, has to be done by using other solution called `gdx-facebook` from Thomas Pronold. It is LibGDX extension for cross-platform support for Facebook Graph API. The stable version used in the cognitive app is 1.5.0 last updated in 4.12.2018. This extension has four contributors and is in the same state as the previous library with well-described documentation on GitHub wiki. Currently well working, except it can not be counted on in the future with updates. Facebook API firstly only needs to log-in with permissions where basic permissions are for email, public profile and user friends, and for everything else, it is needed to request additional permissions when required. Mutually with log-in action, it is requested to set read or publish permission. After successful log-in, Facebook Graph API request is used to get JSON result with the detailed requested information about the user, e.g., email, name or id. With unique user id from Facebook Graph API, custom sign-in through token is done. Firebase has strict rules for logging-in with a token, which has to be done using JSON Web Token standard. Manually creating the JWT token is done using very popular JJWT library [57]. With JWT library, the custom token claims requisites as described in documentation [58] are created.

6.2.2 Users data gathering

Gathering the statistics from games played by the users is a vital part of the app. Data transferring to the realtime database is done with the GDX Fireapp API mentioned in the previous section.

Firstly, and most importantly, the Firebase itself has server located database rules, that determine access rights to the realtime database. That means the read and write requests have to pass the rules to be completed. The database consists of two main collections Users and Games. Every authenticated user has the right to write into and read his record defined by his id. Even though read request to other users game statistics is harmless, except for the administrator, it is forbidden, as the rules would have to be unnecessarily more complicated to avoid reading personal information. On the other hand, the games collection is public for reading to everyone for fetching overview statistics about the games. However, write rule to this collection is allowed only to the user that has the admin rights. Ad-

min rights can not be currently obtained other than by manually entering a value into the database from the security reasons. Only admin can update the overview statistics, which will be done manually from the admin app account with a future extension providing an automatically scheduled job or so-called "cron job". Above defined rules are written using Firebase custom JSON style syntax, which can be seen in the listing 6.1.

```
"rules": {
  "users": {
    ".read": "root.child('users').child(auth.uid).
      ↪ child('admin').exists()",
    "$uid": {
      ".read": "$uid === auth.uid || root.child('
        ↪ users').child(auth.uid).child('admin').
        ↪ exists()",
      ".write": "$uid === auth.uid"
    }
  },
  "games": {
    ".read": true,
    ".write": "root.child('users').child(auth.uid).
      ↪ child('admin').exists()"
  }
}
```

Code 6.1: Firebase database rules

Setting and reading a single value in the database is a relatively easy thing with promises. Nevertheless, calling promises in a cycle to read a list or whole map objects would be highly inefficient. As Firebase keeps data structured as JSON with covered Java types, they can be represented as a Map, which the API can convert to the object representation. This is called POJO conversion, and this app uses it for user information and game score fetching. When specified to do map conversion, during a promise read request, a list of retrieved values is checked against the instance of Fireapp Map API, which converts it to the specified type or drops it otherwise. This way, the read request can map all values into a single class object instead of getting all the values separately from the database. This conversion method can be even customized by creating its map conversion instance. The code 6.2 shows how to map conversion is called with the read request.


```
GdxFIRDatabase.inst().inReference("users/${GdxFIRAuth.inst
    ↪ ().currentUser.userInfo.uid}/${gameInfo.localeName}")
        .readValue(ScoreModel::class.java)
        .then(Consumer { scoreModel ->
            //Success - Retrieve score data class
        })
        .fail { error, throwable ->
            //Failure - Unable to load score model
        }
    }
```

Code 6.2: Read request to POJO conversion

6.3 Assets

To make a popular app, the visual and acoustic elements must appeal to a broad group of people. That includes textures, particle effects, animations, fonts together with added music and sounds. These files are called as assets which can be extended by other text files like locales. LibGDX recommends the use of AssetManager class for loading and managing assets within the project, as it has some helpful behaviors, explained in their documentation. This app uses this approach of a simple manager for loading all assets.

The ideas for the application's visual style were well thought out from the first moment of development. The winner for graphics style is a lightbulb with many variations on loading, finishing a game, showing demographics, and more. A huge part of app graphics, except icons, are own made products. The animations were created using Adobe After Effects, and the images were done in Paint 3D. The sounds for the app are also own made, as it was recorded on the author's laptop, using a built-in recording application, and then edited in the Adobe After Effects. Editing sound files were mostly done for removing noise, yet sometimes filters were added to create a unique sound.

6.3.1 Game assets

While developing a game, the first thing was always creating its specific textures. Commonly prototype textures were created at first and updated through the implementing phase. After fully finishing the game's implementation and testing the game, final touches were made to the used graphics in the game. As previously said, the graphics for games were created mostly by hand. Although for some games, e.g., "Find the owl," free clipart was downloaded because the author's drawing skills are not good enough to fulfill the

graphics idea.

Every game has its thumbnail in the Games screen. For each game, a few thumbnails were created, and the ideal one was picked. An example with Two Cars game thumbnails can be seen in the picture 6.6, where the leftmost is the currently applied thumbnail. The thumbnails were also updated with time to represent the actual graphics elements.

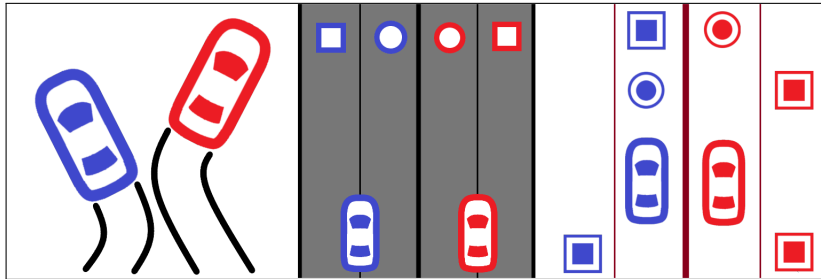


Figure 6.6: Two Cars game designed thumbnails

The sounds for the games were added to the app as the last thing ever as they are not core defining in any of the implemented games. Generally, sounds are the only addition that should help the users immerse more into the app. Adding music was on the mind as well. Nevertheless, adding music can be a more disruptive element and therefore is not something that has to be added into the app soon.

For games that are using physics are creating rigid bodies that react to collisions essential. Having a Box2D physics engine available in LibGDX, creating dynamic bodies went through many processes like creating shape points to copy image object edges, converting points from pixel units to world units, and decomposing shape into multiple convex polygons which physics engines usually use. These processes are very demoralizing and time-consuming. Hence a Physics Body Editor [59] is utilized for creating game physics files. The program consists of functions to draw shape points over a transparent image and automatically decomposes them into convex polygons. Formed physics file is the last significant asset required for particular games. The file is afterward transformed into the Box2D body, which is used in games as an object.

Two issues with the Physics Body Editor are that the physics body points are normalized, and the texture applied to the body is not, hence every time a physics object is presented, the texture scaling has to be recalculated according to the size of the body world size. The second one is setting the origin point, displayed as a red target in the image 6.7. Scaling and rotation transformation methods of physics body and sprite image (texture) are ap-

plied to origin location. By default, the origin is at the bottom left corner, but with normalized shape points, the rotation function is not identical with appropriately recalculated sprite rotation. Therefore, the origin is manually set to the center of the body and a sprite image, which fixes the issue with rotation in case of colliding with other bodies.

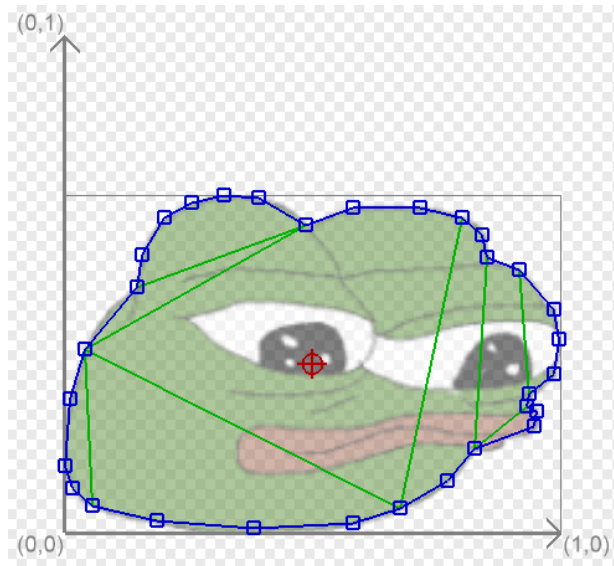


Figure 6.7: Flying Frog body shape drawn in Physics Body Editor

6.3.2 Localization

Since the app is created for the public, localization support is added. Currently, only two Czech and English languages are supported, but thanks to the LibGDX I18NBundle class implementation, more translations can be easily added. The locale files are a set of properties files that share the same base name with variants indicating language code, country code, and locale variant. Adding a new localization is hence done by translating the default file texts. An example of current properties files names are:

- blik.properties
- blik_cs.properties
- blik_en.properties

The `blik.properties` file is default properties file, which contains default contents. As each file contains key-value items, switching between languages does not require changes to the source code. If a key is not

found in the specific properties file, it is looked up in decreasingly specific properties files until found. Thus if "game_name" key is not found in `blik_cs.properties` it is looked up in `blik.properties` file.

Another advantage is that the texts can contain parameters or even supported options for plural forms written inside locale text. The text is automatically formatted when called, by comparing the format message choices to the inserted parameter value. This can be seen in code preview 6.3 below.

```
//blik.properties
game_score=You have achieved {0,number} {0,choice,0#
    ↪ nothing|1#point|1<points}.
..
//blik_cs.properties
game_score=Ziskal jsi {0,number} {0,choice,0#nic|1#bod|1<
    ↪ bodu}.
..

println(blikLocale.format("game_score", 25))
```

Code 6.3: Plural forms in localization

6.3.3 App textures

The app contains a large number of textures, starting from icons. The icons used are made by Stephen Hutchings [60] and are distributed under CC BY-SA 4.0 license. Only a few app icons are made by this project author, altogether with custom coloring and button skins to the icons. The font choice has fallen on osifont [61], which is a "Free TrueType font for CAD projects," as defined by the font author. The reason for this is an entirely free legal use as the font is under GNU GPL license version 2 with font exception. Another decisive impetus was full font support for forty-one languages, which gives future language expansion an open hand.

For mobile apps, the frequently used images can cause unwanted context switches. Therefore, it is more efficient to save the textures into a texture atlas, which is one image containing all those smaller images. Access to images in the texture atlas is done by getting custom texture coordinates from the atlas file. In the LibGDX, the use of texture atlases is even more extended to Skins. Skin class stores resources for the UI widgets defined using the JSON file. Creating a skin is also done by an external application called Skin Composer [62], where images, colors, nine-patches, fonts, and

alike are added and connected to UI widgets, and after that lets the user generate all necessary files. Skin files were also manually edited to allow the use of different font sizes in the application without unnecessary font scaling, which can cause font pixelating.

Ultimately, for the application to be more reputable, the animations were created. As mentioned, an Adobe After Effects program was used to create the animations for a specific event in the app:

- App launching
- Menu loading
- General loading (e.g., log-in connection)
- Game end

The animations were hand-drawn and custom animated based on the author's imagination. Most animations, including the core app visual idol, a lightbulb, were drawn using line shapes and then animated accordingly to fit the event intention. The animation is done by placing keyframes for the transform action in the timeline. In the picture 6.8 can be seen that position, scaling, and rotation are basic editing transforms for most animations. Most of the time, every visible shape in the animation has its transform actions, so for example, a loading animation, where background lines are falling, contains about seven different moving parts, each with its animation settings. When calculating the time estimate taken by creating graphics and animations for the whole application, it took approximately 1/5 of the total time spent on app development.

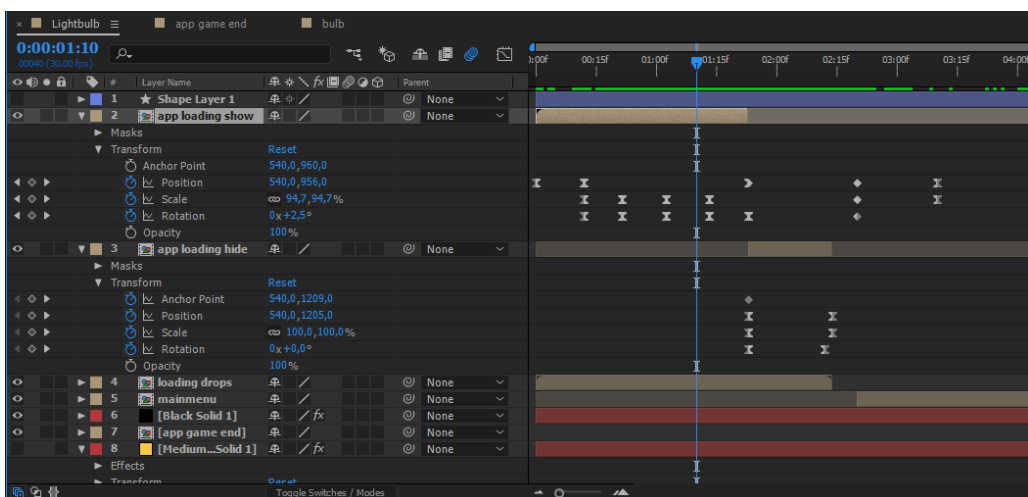


Figure 6.8: Adobe After Effects timeline with visible keyframes

6.4 App screens

The app contains six different screens, not counting the game's screens, that can be shown during its lifetime. The picture 6.9 shows the flow between screens, where can be seen from which screen can the app follow up to another. Core screens are divided into four screens: Menu, Games, Settings, and Statistics. Each of these four screens contains a navigation bar widget at the bottom of the screen, whose purpose is to provide the switching between them.

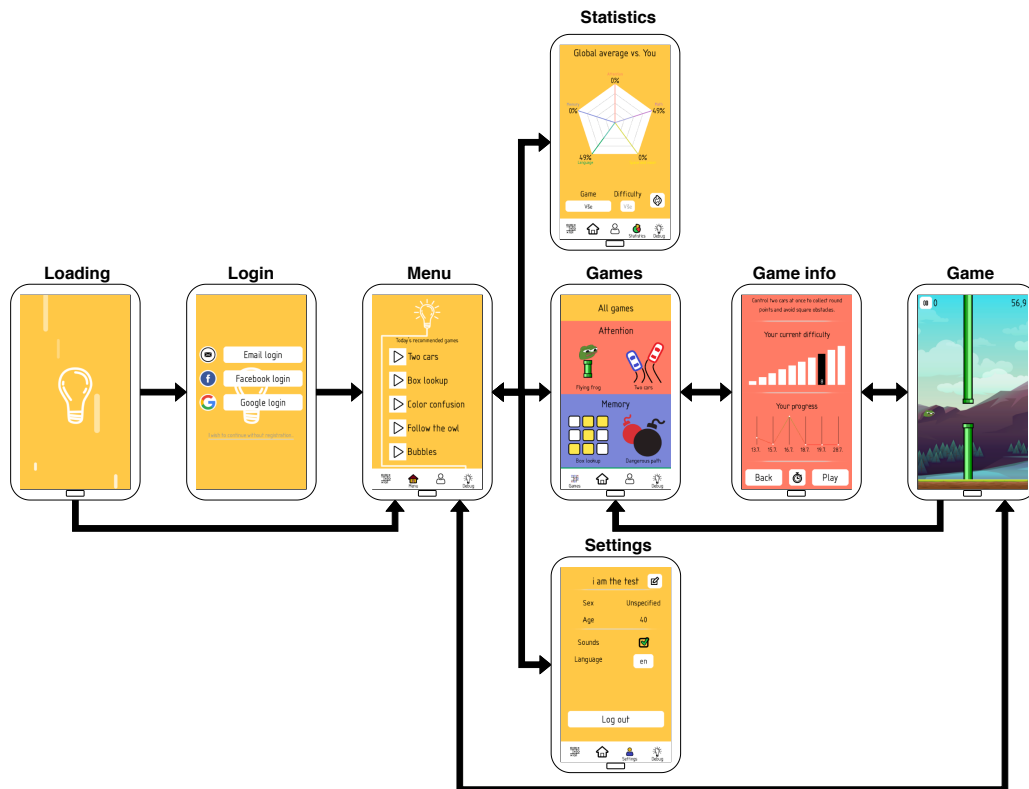


Figure 6.9: App screen flow diagram

The loading screen is the first screen shown when launching a game. It loads all necessary assets for the default front-end display. If the user is not logged in, the Login screen shows up. Otherwise, the user is automatically moved to the menu screen.

Login screen presents and processes all log-in options for the app. The user can also continue without logging in tapping the text button, "I wish to continue without registration.." at the bottom of the screen. The user then proceeds to the Menu screen.

The menu screen contains laid out recommended games. After the fade-in animation for the game is finished, it becomes touchable, immediately

starting the clicked game. There are five recommended games each day, one for every cognitive type. For now, each game is chosen randomly based on the day date. Therefore the recommended games stay the same for the whole day. Firstly, the game recommendation had to be done by choosing the user's worst progress games. Unfortunately, this failed because the collection of games is not big enough, so if someone is a little bit better in one game, the second game would be recommended every day, which can be irritating. If the recommended game for the current day is completed, the "Play" icon changes to "Completed" icon.

As the name suggests, the Settings screen is all about settings and options that can be changed in the app. At the top area, the username is shown, which not logged-in user can not change the default "User" name. If logged-in, additional personal information is shown, and it can be edited by tapping the "Edit" icon next to the username. The next options available for each user are sounds on/off and language settings. Sounds on/off checkbox is self-explanatory, and language settings display a list of languages from which the user can choose. Currently, there are only Czech and English language options. In case the user is not logged in, at the bottom of the Settings screen, the same log-in options as in the Login screen are presented.

For data presentation and comparison against other users, the Statistics screen was developed. All functions in this screen work only when the user is logged in and connected to the internet. Provided that these requirements are met, the user can initially see the radar chart of global cognitive types with his percentages of cognitive types compared. Those global statistics can be changed between the global top achieved scores or global average scores. Other than this, while comparing a specific game, which uses a bar chart. Examining a specific game lets the user compare against a specific category of age range or difficulties for the given game.

Games screen shows a collection of all playable games in a scroll panel. Tapping any game sends the user to a specific Game information screen. The games are split into five cognitive categories, where every category holds onto one specific color:

- Attention = Red
- Memory = Blue
- Language = Green
- Space and vision = Yellow
- Math = Violet

The Game information screen presents user statistics for the specific game. Furthermore, the navigation bar is replaced with a user interface for returning to the Games screen, to launch the game and to set custom game properties. At the top, the screen contains a game thumbnail, game description, then the user's current difficulty, and if at least one game played, the weekly chart progress with top scores is shown under difficulty chart. When tapping the "Timer" icon button at the bottom middle screen, the user is moved to screen where he can change to play with custom difficulty that he already achieved and custom timer. Playing a game with a custom timer other than one minute is not recorded and does not count towards the user statistics.

6.5 Games

Based on the game design (Chapter 4) nine games are implemented. Every game holds the same basic GUI, which is at the top of the screen. It consists of the "Pause" icon button, score label, and game timer. Concurrently every game needs to have indispensable screen methods, like render, hide, or dispose. All games, therefore, inherit common abstract class `BlikGameScreen`.

The abstract `BlikGameScreen` implements interfaces for classic screen methods, inputs, contact listener for physic objects, contacts, and general game information properties. It also contains abstract methods for game-specific events, like creating base game UI, game end, logic step, or sprites redraw. The whole countdown and timer process is also calculated in this class. Most importantly, the render method, which forms the main game loop, works with all those abstract methods, so they must be specified in the game class. Lastly, inputs for going back using HW or SW buttons are overridden, so going back is not exiting the application and instead reacts to the app by moving to the previous screen.

Each game has its configuration file with constant specific settings, including textures name, difficulty constants, game constants, and game preference names for local saves. The requirement is including two interfaces, where one that is already included in inherited parent class `BlikGameScreen`. That interface holds general properties about the game as description, thumbnail, cognitive type, and duration. The second interface is for specific game information that needs to be accessed as a static value. It includes property like locale game name and mostly shared preference names for saving information to the local device outside the game class. Even though the implementation regularly followed the games design, more specific imple-

mentation details about each game are described below.

6.5.1 Box lookup

This game features a matrix where fields to be remembered are displayed in yellow color. The game starts with flipping the random fields to remember and hiding them after a two-second delay. The number of fields to remember is based on half the number of matrix fields minus difficulty coefficient. Tapping the fields reveals the correct or incorrect move, as tapping the wrong field fills the field with red color. If the user has tapped the wrong field, the whole board is revealed, and the game continues to show a new board to remember with lowering the difficulty coefficient.

During the development, the colors and matrix boxes graphics was re-worked to be sharper for better visibility. The problem ascended with square matrixes that been so large that fields were tiny. Therefore the users with bigger fingers could not be sure to tap the right field. This functionality has been altered, so only seven fields in a row can be seen at maximum, while the matrix is expanded by adding rows vertically. The development progress can be seen in the pictures 6.10.

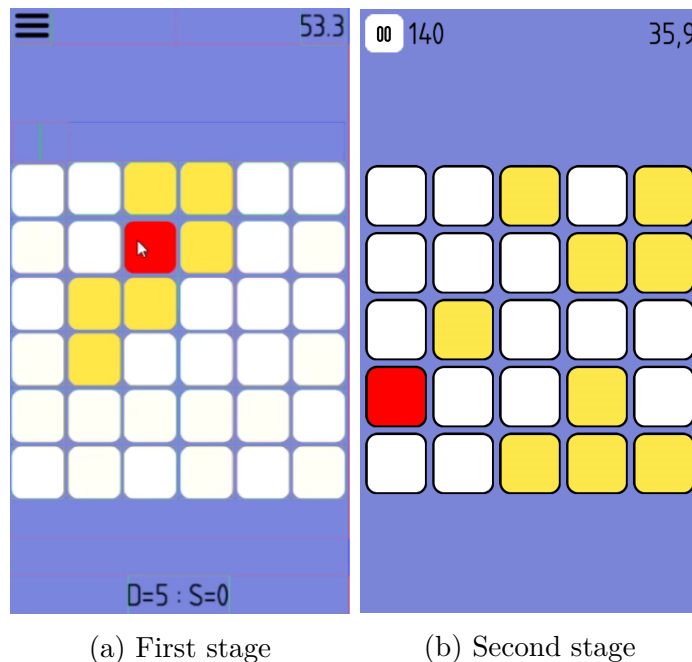


Figure 6.10: Box lookup game development stages

6.5.2 Dangerous path

Dangerous path game is also based on the same matrix generation method as the Box lookup game. The main change is in the gameplay, where a path is shown for a few seconds between two points, and every other field is a dangerous bomb, which should not be touched. After hiding, the user has to draw the remembered path between those two highlighted points. The path length is defined by difficulty constant and size of the matrix. As it seems that the game has a pretty similar visual presentation, the involved cognitive abilities are slightly different. Hence this game entails more visual-spatial imagination.

The core implementation revolves around path generation on the given matrix. The path can not be generated to go over itself and should not be generated with more than two path neighbors. Therefore, the generation is not as simple as it seems at first, and a recursive function is used. The path is gradually added to the matrix and also manages the visited matrix of previously scanned moves. Sometimes the path can generate paths that spiral into itself and gets to the dead point, so it tries a new generation. The more times this happens, the game could suffer a lag, which is solved by adding a deadlock condition, which lowers the path length, which hinders the possibility of the next path generation loop. The path moves only in four cardinal directions.

6.5.3 Flying frog

The first attention game focuses on reaction speed by recognizing two different stimuli. This game uses physics to give the user more experience when failing. The frog can crash into the obstacle or ground when moving, if the user taps too low. So not only reaction, but even precision is critical. The primary user goal is to fly through the hole in tube obstacles, which is predicted by green arrow and dodge obstacles that are predicted out by the red arrow. With higher difficulties, fast recognition between the wrong red incoming obstacle and good green is vital.

Tapping on the screen moves the frog using a decreasing function. Thus the frog slows down at the end of the tap. If the frog hits any obstacle, it falls, waits until the obstacles go through, and the frog resets at the left middle of the screen. The obstacles are generated using the Poolable interface, which means the memory is not overloaded by generating new texture objects, but the obstacle object is reused if set as dead. The obstacle is set to dead when it is entirely hidden behind the left screen edge. As this was the first

game developed, the stages of development were carefully monitored, and the outcome can be seen on the figures 6.11.

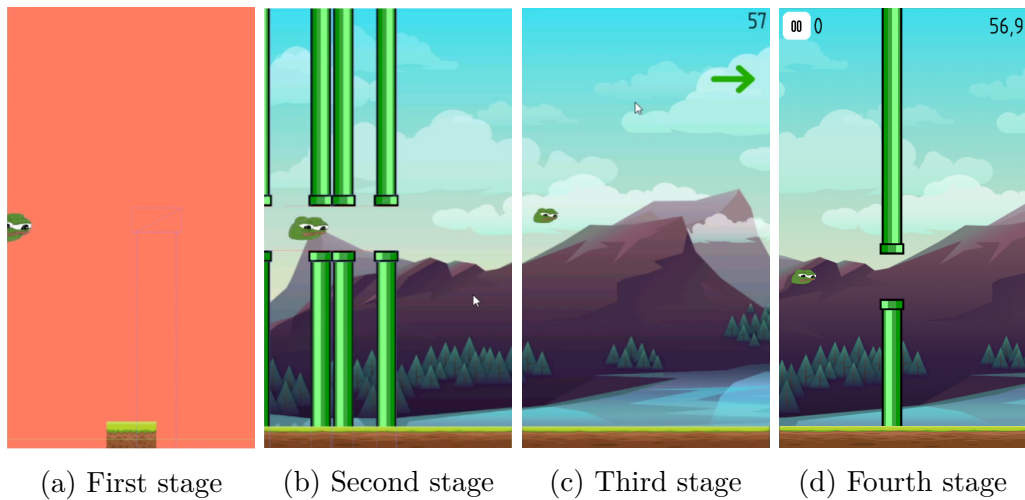


Figure 6.11: Flying Frog game development stages

6.5.4 Two cars

The second attention game focuses more on alternating attention, where two cars at the time must be tracked. The main goal is to catch all circular shapes and dodge all square obstacles coming down on two road lanes for each car. The game has a very similar approach with Flying Frog game on obstacles Poolable generation, the usage of physics, and game resetting if failing the primary goal. The movement of cars and shapes speeds up with higher difficulty. The cars are switching lanes only by tapping at the left or right side of the screen assigned for each car.

The main complication was unexpectedly graphics options. All assets had to be done well for excellent visibility, and many-colored combinations were tried out. The shapes contained filled color textures to a variety of symbols inside the shape. The road color after all attempts stayed unchanged to attention type color setting, and the color of lanes has been filled into darker shades of red.

6.5.5 Follow the owl

Follow the owl game is entirely based on Corsi Block tapping test as specified in the design section 4.3.1. On the screen are randomly positioned ten boxes, where the owl can appear. After a while, the owl path is generated, and then

it starts appearing randomly in the boxes. The user has to remember the boxes' sequence, where the owl has appeared and repeat it by tapping them in the correct order. After finishing or failing, the screen is reset with newly random positioned boxes, and the owl starts showing again.

To correctly generate a sequence for an owl, it can not appear at the same spot as before, because that could be confusing. Hence, the owl has to change position every time. With increasing difficulty, the owl appears at more positions and displays for lesser time.

6.5.6 Hiding mosquito

Again, this game uses the same generating matrix as Box lookup and Dangerous path games. The mosquito appears in one field and hides right after a brief while. Then the grid massively lowers visibility and shows sequentially eight cardinal directions, where the mosquito moved. The user remembers all those directions, and after matrix reappearing, he has to recall them to map a path the mosquito went. Finally, he has to tap the position in the matrix, where the mosquito is present.

The game was changed a lot for its direction display. Firstly the directions were big as one matrix field in the middle of the screen with the whole matrix shown. The testers reported this as "fairly easy" to do since the user can point at the boxes the whole time without much thought. The matrix visibility was hindered enough to be a little blank, yet the small direction arrow still allowed focusing more on the movement around the boxes and not focusing on directions. Lastly, the direction arrow was enlarged to fit the whole screen, so the user has to focus on the arrow and not the matrix fields.

6.5.7 Color confusion

The most confusing game for many testers and the most popular is based on the Stroop Color and Word Test. The implementation holds to the first designed idea from the picture 4.4. The cards in the middle of the screen are showing color label text. The user must remember the font color of the card text, which is the answer color text from the bottom provided options.

The maximum number of colors is currently ten, where each should be distinct from each one. The color texts have to generate accordingly to the difficulty. For example, in the fourth difficulty, there are five active colors, yet only four text answers. The options always have to contain the answer text color, yet it can have randomly set font color from the five possible

colors. Middle cards are Poolable objects, so they are reused with a different color each time. If the user taps the right color, the card shows a checkmark and is thrown to the right side of the screen. Otherwise, the red cross shows, and the card is thrown to the left side of the screen.

6.5.8 Bubbles

The bubbles game consists of circle shapes, so-called bubbles, that have a number. At the bottom left corner is the target sum, that has to be accumulated by tapping the bubbles, which are added to the current sum displayed at the bottom right corner. If the current sum equals the target, the bubbles are destroyed, and a new target sum is generated.

The bubbles are also generated as a Poolable object. Each difficulty has its azure overflow line, where the bubbles are generated and falls to the ground. If any bubble is above this line at any time, the difficulty lowers with half active bubbles disappearing. The difficulty is raised when the number of bubbles is lower than three, and a new bunch of bubbles appears at once, dependent on difficulty. The target sum is generated from the current random active bubbles. Therefore the user can not ever experience waiting for new numbers.

The physics side of the circle was, for the first time, hard to implement because to simulate a circle physics with rotating texture, it needs more properties to set. The friction, density, and restitution had to be set precisely because, from many tries, unrealistic physics was experienced like crazy bouncing, no rotations at all, falling through the floor, gaining acceleration when colliding, and more. Also, the numbers were implemented as a label connected to the bubble object, but unfortunately, it had unwanted positioning, as each number font had different proportions and needed text calculations for each possibility. Each number is now included as a single texture, a more smooth and more comfortable approach. The development stages can be seen in figure 6.12.

6.5.9 Estimation

The last game focused on fast math skills trains the estimation guess from presented numbers. From the presented numbers on the right side of the screen, the user has to touch the axis position where the sum of the numbers equals the axis number. Three possible outcomes can show precise hit, good hit, and miss. The user rises higher in difficulty with more precise hits.

This game felt the most problematic as drawing the dynamically adapt-

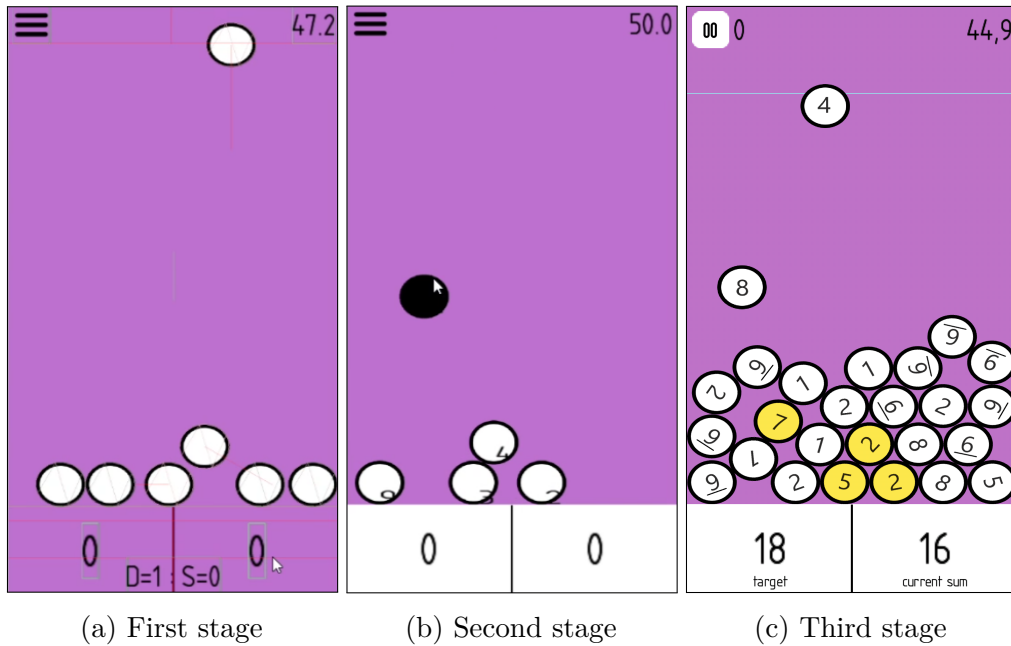


Figure 6.12: Bubbles game development stages

ing axis to the screen felt like a puzzle. Each axis point is equal to the square center, which has a different size depending on the number value. Therefore offsets for each square differ as the square textures are positioned from the bottom left corner and not the center. When touching the screen, the screen coordinates have to be calculated to camera coordinates, and these coordinates have to be converted into axis values depending on the graphical axis positioning, which is set in its world camera coordinates. Having this fixed up, other updates to the game were simple, as the sum of the generated numbers can not be higher than the axis maximum, which is static. The axis minimum is zero, and the axis maximum is forty.

7 Testing of implemented app

This chapter will present that testing was one of the main procedures while developing an app. The app was continually tested during the whole time of development by many varieties of test approaches, users, and devices. Included are following mini-scenarios, monkey tests, continual alpha/beta release testing, and even watching users work with the app for the first time without telling them anything about it.

7.1 Mini-scenarios

One type of testing is provided with below-described mini-scenarios. This form shows the exemplary outcomes and feedback for app controls and how they will react to a specific input.

7.1.1 App launch

This scenario tests successful launch of the application to the main menu screen. To get to the main menu, the user has to get through log-in system with one of three available options: without logging in, with logging in or with previously logged in account.

Scenario - App start

After the user launches the app:

1. Loading screen shows up.
 - (a) User was previously logged in:
Continue to step 3.
 - (b) User was not previously logged in:
Continue to step 2.
2. Login screen shows up.
 - (a) User logs in using presented options:
Continue to step 3.
 - (b) User continues without logging in:
Continue to step 3.

3. Menu screen shows up.

The user is successfully on the menu screen logged in when he uses a log in option or was previously logged in. If he continues without logging in, he still successfully gets into the menu screen. In all cases, the user can use every core app functionality. The application was tested and worked accordingly to the scenario.

7.1.2 User authentication

Scenarios that put to the test every login screen process in detail. Furthermore, tests also use logout action, which allows logging in using other options.

Scenario - Google login

User is present on the login screen and will use Google account to log in:

1. User taps the Google login button.
 - (a) Has internet connection:
Continue to step 2.
 - (b) Has no internet connection:
Refer to Fallback - Login **A**.
2. Google authentication dialog window shows.
 - (a) User logs in to Google following the dialog instructions:
Continue to step 3.
 - (b) User cancels the log-in process:
Refer to Fallback - Login **B**.
3. Login screen animation waits until the login process is completed and receives user data.
4. Login is completed, login screen changes to the menu screen.

The user is logged in with Google account, on the menu screen, and can use the core app functions. When he fails to log in by any means, he will still be present on the login screen to continue without logging in or login using other options. The application was tested and worked accordingly to the scenario.

Scenario - Facebook login

User is present on the login screen and will use Facebook account to log in:

1. User taps the Facebook login button.
 - (a) Has internet connection:
Continue to step 2.
 - (b) Has no internet connection:
Refer to Fallback - Login **A**.
2. Facebook login authentication dialog window shows.
 - (a) User logs in to Facebook following the dialog instructions:
Continue to step 3.
 - (b) User cancels the log-in process:
Refer to Fallback - Login **B**.
3. Login screen animation waits until the login process is completed and receives user data.
4. Login is completed, login screen changes to the menu screen.

The user is logged in with Facebook account, on the menu screen, and can use the core app functions. When he fails to log in by any means, he will still be present on the login screen to continue without logging in or login using other options. The scenario was tested and is working correctly.

Scenario - Email login

User is present on the login screen and will use email account to log in:

1. User taps the Email login button.
2. Email login screen shows.
 - (a) User has internet connection, inputs registered email with password and taps the "Login" button:
Continue to step 3.
 - (b) User has no internet connection and taps the "Login" button:
Refer to Fallback - Login **A**.
 - (c) User has internet connection, does not input anything and taps the "Login" button:
Refer to Fallback - Login **C**.

- (d) User has internet connection, inputs wrong email format and taps the "Login" button:
Refer to Fallback - Login **C**.
- (e) User has internet connection, inputs correct email format and taps the "Login" button:
Refer to Fallback - Login **D**.
- (f) User has internet connection, inputs correct email format but wrong password and taps the "Login" button:
Refer to Fallback - Login **E**.

3. Login screen animation waits until the login process is completed and receives user data.

4. Login is completed, login screen changes to the menu screen.

The user is logged in with an email account, on the menu screen, and can use the core app functions. When he fails to log in because of no internet connection, he will still be present on the login screen to continue without logging in or log in using other options. If he fails to log in by failing inputs, he will stay on the email login screen to try again or go back to the login screen. Tested as described and app is working correctly.

Scenario - Continue without login

User is present on the login screen and does not want to log in:

- 1.** User taps on the bottom text button: "I wish to continue without registration.."
- 2.** User is not logged in, login screen changes to the menu screen.

Tested scenario shows the user is not logged in on the menu screen and can use the core app functions. Therefore it works correctly.

Fallback - Login

This section refers to the failed actions mentioned above. Depending on the fallback, the user is sent to a specific step in the executed scenario.

- A.**
 - a) Error label shows "No internet connection".
 - b) Return to step 1.
- B.**
 - a) Error label shows "Login failed".

- b) Return to step 1.
- C. a) Error label shows "Email address is not valid".
b) Return to step 2.
- D. a) Error label shows "Password is not valid".
b) Return to step 2.
- E. a) Error label shows "Wrong credentials".
b) Return to step 2.

Evaluation is equal to the currently executed scenario above.

Scenario - Logout

When the user is logged in by any option and present on the menu screen. Scenario tests if the can successfully log out of the app.

1. User moves to the Settings screen.
2. User taps the "Logout" button on the bottom of the screen.
3. User-specific settings (age, gender) disappear and nickname resets to universal name in the settings screen and the "Logout" button is replaced by login options.
4. User is logged out.

The scenario was tested and is working correctly. The user is successfully logged out and present on the settings screen. Next time when launching an application, the login screen appears as the login token is removed.

7.1.3 User registration

As two of log in options are through common social media that holds its own registration methods, the app itself allows custom registration in form of email login. This action creates an email account that can be used to log in into the app.

Scenario - Email registration

Scenario tests new email account registration process and immediate possibility to use the account. User is present on the email login screen without having an account:

1. User taps the "Create new account" text button on the bottom of the screen.
2. Email registration screen shows.
 - a) User has no internet connection and taps the "Create account" button:
Refer to Fallback - Registration **A**.
 - b) User has an internet connection, does not input anything and taps the "Create account" button:
Refer to Fallback - Registration **B**.
 - c) User has an internet connection, inputs wrong email format and taps the "Create account" button:
Refer to Fallback - Registration **B**.
 - d) User has an internet connection, inputs correct email format and taps the "Create account" button:
Refer to Fallback - Registration **C**.
 - e) User has an internet connection, inputs correct email format, password with less than six characters and taps the "Create account" button:
Refer to Fallback - Registration **C**.
 - f) User has an internet connection, inputs correct email format, password with more than six characters and taps the "Create account" button:
Refer to Fallback - Registration **D**.
 - g) User has an internet connection, inputs correct email format, password with more than six characters, "Confirm password" that does not match the password and taps the "Create account" button:
Refer to Fallback - Registration **D**.
 - h) User has an internet connection, inputs correct email format, password with more than six characters, "Confirm password" that does match the password, and taps the "Create account" button:
Continue to step 3.

3. Registration screen animation waits until the registration process is completed and receives user data.
4. Registration is completed, the user is logged in, and registration screen changes to the menu screen if the registration was made on the login screen. Otherwise, the screen changes to settings screen if the registration was made starting from that screen.

The user is logged in with a newly registered email account. This account is currently not checked if real or not, but future logging using this account is possible. Failing to register the account by failing on the fallback problems will make the user stay at the email registration screen. The application was tested and worked accordingly to the scenario.

Fallback - Registration

Failed actions in registration are resolved here, transferring the user to the pointed out registration step.

- A. a) Error label shows "No internet connection".
b) Return to step 2.
- B. a) Error label shows "Email address is not valid".
b) Return to step 2.
- C. a) Error label shows "Password is not valid (At least 6 characters)".
b) Return to step 2.
- D. a) Error label shows "Passwords must match".
b) Return to step 2.

User is with every option sent to email registration screen and therefore it works as intended.

7.1.4 User settings

These scenarios test all settings that can be changed in the settings screen for the app. Logged in user has more options to change side by side with options available for not logged in user. Logged in user has the option to change his personal information.

Scenario - Logged out user - Language change

User is present on the settings screen and is not logged in:

1. User taps the Language button.
2. Language change screen shows.
3. User taps the language of his choice from the select box.
4. User confirms the change by tapping the "Accept changes" button.
5. The language of the whole app changed to the cast one.

The labels, language tag button, and navigation bar text in settings screen changed appropriately to the language chosen. When moving to the menu screen, all game names are in the chosen language, and the games screen with statistics screen is also translated. The application was tested and worked accordingly to the scenario.

Scenario - Logged out user - Sound turn on

User is present on the settings screen, is not logged in and Sounds checkbox is unchecked:

1. User taps the Sounds checkbox.
2. Sounds checkbox is checked. The sound is turned on.
3. User can hear menu music with the device's system volume set.

Tests proven that the sounds of the app can be heard. Menu music and in-game sounds are playing. The application was tested and worked accordingly to the scenario.

Scenario - Logged out user - Sound turn off

User is present on the settings screen, is not logged in and Sounds checkbox is checked:

1. User taps the Sounds checkbox.
2. Sounds checkbox is unchecked. The sound is turned off.
3. User can not hear menu music.

Tests proven the sounds of the app can not be heard. Menu music and in-game sounds are not playing even when changing the device's system volume. The application was tested and worked accordingly to the scenario.

Scenario - Logged in user - User information change

All scenarios above are available for the logged-in user. Additional options are open for logged-in users, as personal information can be changed, which is what this scenario tests. User is present on the settings screen and is logged in:

1. User taps the "Edit" button from the right next to the nickname on the top of the settings screen.
2. Personal information edit screen shows.
 - a) User changes name or sex or age or any combination of them and taps the "Accept changes" button:
Continue to step 3.
3. Screen changes to the settings screen, and the user has changed his personal information.

The user has changed personal information on his local device, and it can be seen in settings screen as name, sex, and age labels represent his picked selections. If online, the information is instantly updated to the database. If offline, next time the user goes online, the information is updated. The application was tested and worked accordingly to the scenario.

7.1.5 Games information

This section contains scenarios that test the game information screen elements. It involves difficulty display, week scores graph, and top ten scores. Types of tests will focus on the online data synchronizing and element changes after playing any game.

Scenario - Finishing the game - First time

The data from games are saved into the preferences in the local device. A general overview of previously played games are displayed in the game information screen, and this scenario checks changes of this information after playing a game. User is present on the game information screen:

1. The picture 7.1a shows the information before the first game.
2. User taps the "Play" button and plays the full game.
3. After the game is finished with the score "510", the screen changes to the games screen.

4. Tapping the thumbnail of the currently played game changes screen to the game information screen.
5. Game information is accordingly updated, and after the first game played, new information about week progress and top ten scores is shown. This can be seen in the picture 7.1b.

The test has successfully updated game information based on the score achieved in the first game. Moreover, with the first score, user progress and score ladder are created and shown. Game advanced to new difficulty, the top ten ladder has been updated with a new score, and the weekly progress graph shows user's improvement to previous day 6. August as the game was played on 7. August. The application worked accordingly to the scenario.

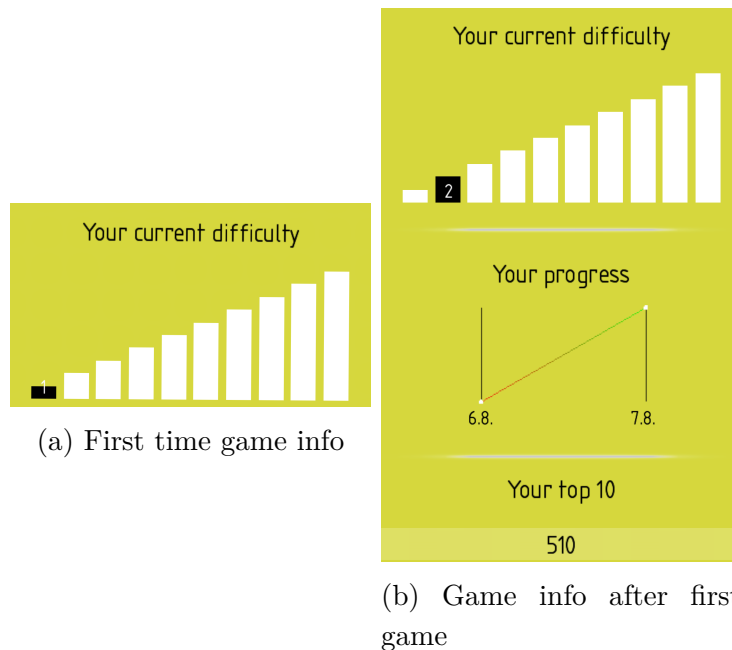


Figure 7.1: Game information changes for the first time played game

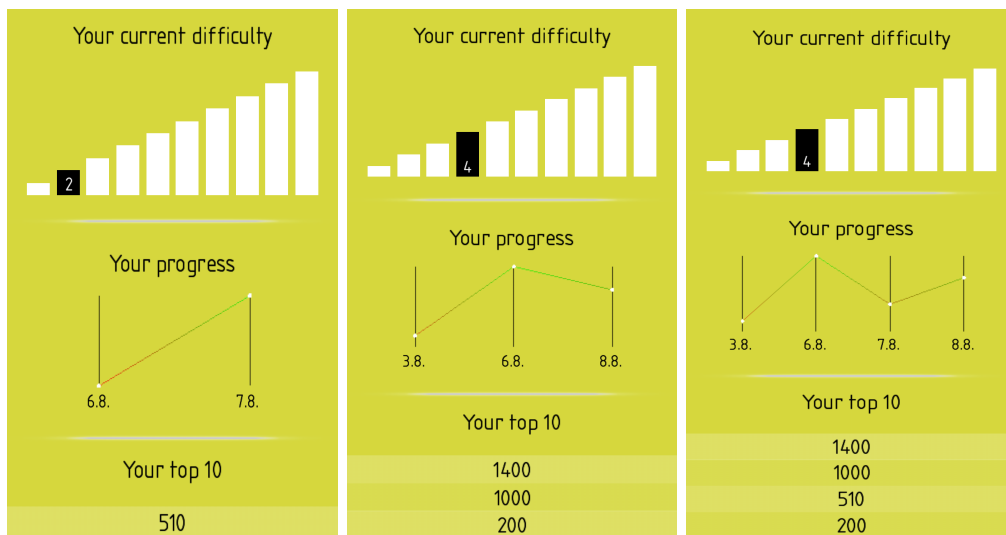
Scenario - Online synchronization

The user can remember his account credentials from the past or regain internet connection after playing a game. Therefore, game information synchronization is essential to merge locally stored data with the database ones as this scenario tests that.

1. The picture 7.2a shows the local stored game information and the picture 7.2b shows the database stored information.

2. The user logs in during app launch or after launch in the settings screen using one of the options.
3. After logging in, the user moves to the games screen and taps the game thumbnail to open the game information screen.
4. Game information is accordingly updated and also fused to form a new united data for both saved states: online and local.

From the picture 7.2c is apparent that the merge finished successfully between local and online data. Current difficulty changed to the higher one; weekly progress chart has linked all achieved scores throughout the days, and the top ten ladder has sorted all the scores in descending order. Merged data are automatically updated to the database and saved to the local device, thus going offline will not remove acquired online data. The application worked accordingly to the scenario.



(a) Local game info (b) Online game info (c) Merged game info

Figure 7.2: Online synchronization of two game information states into one combined

Scenario - Finishing the game - Already played

While having already finished the game for the present-day, another game plays has to be noticeable. This test check changes if the game information properly changes after finishing a game, while having already played game or games in the current day. User is present at the game information screen:

1. The picture 7.3a shows the information before playing the game.
2. User taps the "Play" button and plays the full game.
3. After the game is finished with the score "1800", the screen changes to the games screen.
4. Tapping the thumbnail of the currently played game changes screen to the game information screen.
5. Game information is accordingly updated. Updated information can be seen in the picture 7.1b.

The test game was played on 7. August hence after the game has been played, the score update in week progress chart was made to that date. Also, the top ten ladder has been updated with the new score fitting to the sorted list. The difficulty has a higher value because no mistakes were made in the gameplay. The game information was correctly updated. The application worked accordingly to the scenario.

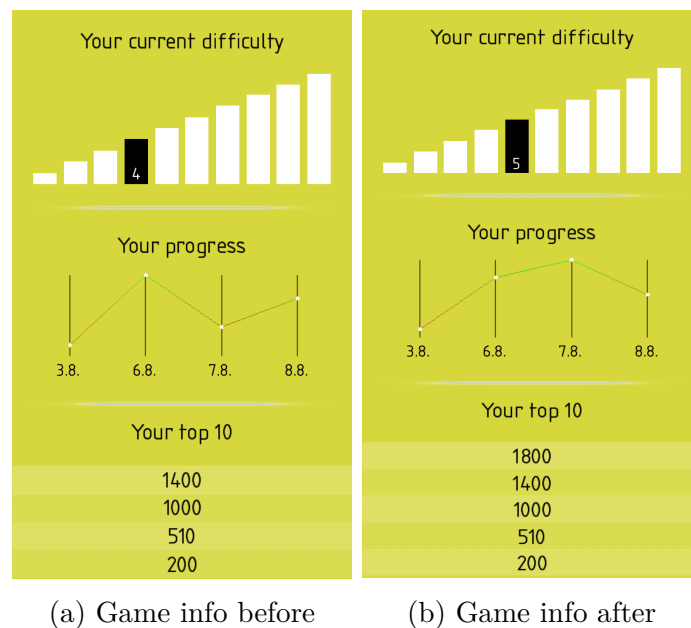


Figure 7.3: Game information changes for the first time played game

7.1.6 Games activities

While playing a game, some unexpected actions can occur, or the user wants to pause the game for another reason. Besides, the games can be launched

from two different screens, which have different behavior before and after the game. This subsection contains scenarios that have to do with those types of actions.

Scenario - Specific game launch

Starting a game from a specific game information screen has to return the user to close related screens after they are done with it. Screen change can be done by finishing the game or exiting it in the middle of the gameplay. This scenario checks these operations. User is present at the game information screen:

1. The user taps the "Play" button.
 - (a) The user taps the screen to start the game:
Continue to step 2
 - (b) The user taps the "Pause" icon button in the top left corner or taps the HW or SW "back" button:
Refer to Fallback - Return **A**.
2. The game starts working.
 - (a) The user taps the "Pause" icon button in the top left corner or taps the HW or SW "back" button:
Continue to step 3
 - (b) The game is completed and ends:
Refer to Fallback - Return **B**.
3. Paused game screen shows up.
 - (a) The user taps the "Continue" icon button at the top of the screen:
Continue to step 2
 - (b) The user taps the "Home" icon button at the bottom of the screen or taps the HW or SW "back" button:
Refer to Fallback - Return **A**.

At all possible outcomes, the screen changes as sought. Therefore they return to the game information screen in case of game exit and to the games screen if the game is finished. The application worked accordingly to the scenario.

Scenario - Recommended game launch

Starting a recommended game from the menu screen changes behavior when exiting or finishing the game. The proper screen has to show, which is checked by this scenario. User is present at the menu screen:

1. The user taps the "Play" icon button or game name label.
 - (a) The user taps the screen to start the game:
Continue to step 2
 - (b) The user taps the "Pause" icon button in the top left corner or taps the HW or SW "back" button:
Refer to Fallback - Return **C**.
2. The game starts working.
 - (a) The user taps the "Pause" icon button in the top left corner or taps the HW or SW "back" button:
Continue to step 3
 - (b) The game is completed and ends:
Refer to Fallback - Return **C**.
3. Paused game screen shows up.
 - (a) The user taps the "Continue" icon button at the top of the screen:
Continue to step 2
 - (b) The user taps the "Home" icon button at the bottom of the screen or taps the HW or SW "back" button:
Refer to Fallback - Return **C**.

At all possible outcomes, the screen changes to menu screen which is requested and thus working correctly.

Fallback - Return

- A.** The user returns to the played game information screen.
- B.** The user returns to the games screen.
- C.** The user returns to the menu screen with recommended games.

Scenario - Game interruption

Sometimes while playing a game on the mobile device, the user can be interrupted or needs to leave the app immediately by clicking the home button. The app should act accordingly and pause the game until the user returns. The interruption is arranged by calling the user while in-game and leaving the game using the home button as the test check if the game is properly paused. The user has launched the game:

1. The user taps the screen to start the game, and the game starts working.
2. The game is interrupted by e.g., a phone call, which is answered or by clicking HW or SW "home" button.
3. Paused game screen pops up. The user can afterward continue, restart, or quit the game.

The paused game screen is shown when interrupted and waits until the next action, which is correctly working and desirable behavior in the app.

Scenario - Game reset

Sometimes the user needs to restart the game because he stared off into space, got interrupted and lost a few seconds in response, or just played badly and wants to try right again. Availability to restart the game is therefore provided in the paused game screen. This scenario tests, whether the restart of the game is performed correctly. The user has launched the game:

1. The user taps the screen to start the game, and the game starts working.
2. The user taps the "Pause" icon button in the top left corner.
3. Paused game screen pops up, and the user taps the "Restart" icon button.
4. The game is restarted, and game properties rollbacks to the starting point.

The game is successfully restarted to the previous launch state and waits to be started. Difficulty rollbacks to the origin value and other specific game states, e.g., obstacle spawning speed, also roll back to its origin value. The

game is in the unrecognizable state to the previously launched game, which correctly passes the test. The application was tested and worked accordingly to the scenario.

7.1.7 User statistics

The last scenario tests the correct display of the statistics screen, whose purpose is to compare the user data to the global app users' data. The statistics screen works solely when logged in and connected to the internet. Therefore those activities are tested. When comparing the specific game data, accurate scores are shown, which must be managed appropriately and refreshed after that specific game is played.

Scenario - Logged in/out

The connection and authentication tests are done by this scenario. User is present on the menu screen and is not logged in:

1. The user taps the "Statistics" icon in navigation panel.
 - (a) Has internet connection:
Refer to Fallback - Statistics **A**.
 - (b) Has no internet connection:
Refer to Fallback - Statistics **A**.
2. The user moves to settings screen by tapping the "Settings" icon in navigation panel and successfully logs in using one log-in option.
3. The user taps the "Statistics" icon in navigation panel.
 - (a) Has internet connection:
Continue to step 4.
 - (b) Has no internet connection:
Refer to Fallback - Statistics **B**.
4. Radar chart shows up and displays global cognitive type data comparison against the user's data.

Test approves that the user successfully sees charts after logging in and connecting to the internet. Logged out user or device without connection does not allow to see statistics functions and provides error texts. The application worked accordingly to the scenario.

Scenario - Logged in user - Game played

The last scenario is focused on data behavior in the statistics screen. The user's data compared against the global and category filters has to change after achieving a significant score in a specific game. From the start, the user never played any game. The user is present on the statistics screen, is logged in and has internet connection:

1. The user sees 0% in the radar chart for attention cognitive type. Zero is as well in specific global bar charts with Two Cars game filter.
2. The user goes to the games screen and starts the Two Cars game.
3. After finishing the game with a 980 score achieved, the user moves back to the statistics screen.
4. Radar chart shows up and displays user's 15% for global average attention cognitive type and 8% for global top attention cognitive type.
5. The user filters out specific Two Cars game.
6. The global best bar chart now shows progress with a score of 980 towards 4899 global best game score, and the global average bar chart now shows progress with a score of 490 towards 1002 global average game score.

The user data in the statistics screen has been successfully updated after the game was played. The user's specific game top and average scores changed accordingly as average are calculated with the first time app launch of score 0 for each game. Global percentage changed correctly based on the previous data and estimated other attention games global data. The application worked accordingly to the scenario.

Fallback - Statistics

- A.
 - a) Error label shows "You are not logged in. Log in to compare with other users."
 - b) Return to step 2.
- B.
 - a) Error label shows "There is no internet connection. To compare with other users, connect to the internet."
 - b) Return to step 2.

7.2 Continuous testing

Since the menu screen implementation was completed, the first testing has begun. At first, the author tested the application functionality by himself on his Samsung S6 Edge android device and three different emulators. The most used testing device Samsung S6 has an operating system Android 7.1 (API 25) and resolution 1440 x 2560. Emulators were mostly used for doubtful situations of testing, so they are mostly only for experimental tests. The first emulator is Nexus S with resolution 2560 x 1800 and Android 5.0 (API 21), the second one is Pixel 2 XL with 480 x 800 resolution, and latest Android 10.0 (API 29) and the third one is Nexus 5X with resolution 1080x1920 and Android 6.0 (API 23). The third emulator was rarely used due to the similarity with the authors' phone. Therefore, it was used only when the author's phone could not be used.

Even though single person testing with three emulators have shown many major bugs and problems with the app in the development progress, it did not represent many other even more critical problems in other devices. Since the app contained the first playable game, the real testing has begun as the application has been expanded between a narrowed group of testers. This group of three testers was in the age category of 20-30 with in-depth knowledge in the IT field, and they used their phones as testing devices.

With the expansion of playable games in the app, the increase of testers was directly proportional. At the current time, fifteen testers officially created a sign-in account that they are randomly accessing and trying an app by playing provided nine games or by just goofing around. This number is only defined by the Firebase authentication sign-in view with removing duplicate users under different accounts. When encountering a bug or something they would like to change, they mostly write the author through private social media about it or speak about it face to face.

Another testing was made with users that did not want to log in the application, yet were eager to try the app. This type of testing was made with personal meetings while handing them the mobile device with only information about how to launch the application and what the application is about. The testing was then done by looking at the user trying to learn with the app. He was not provided with any information about controls or saying what to do. Using this approach, even though it was not recorded or written down, was informative about how the user uses the app in the first steps. If they were finished with the app, the summary reactions were written down for later bug fixing or improving. In the next development session, the focus was on fixing these bugs and making new features and functionalities

that were not in the application last time, focusing on testers' conclusions.

The testers, who were not told what to do, many of them managed to get through the login part. Afterward, they stayed at the menu screen and mostly tapped one of the recommended games. As games did not have until the last release manual instructions on how to play them, many users were confused about what to do, yet found how to play them in one or two retries. Feedback to this was that the games do not have any instructions on how to play them, and there is a description of games in games screen, but they did not know about it. Afterward, they got the hang of the three icon navigation bar. The settings screen is pretty much self-explaining, and nobody had problems with that. Games section was always pleasant for them, probably because of the vast content of game thumbnail images. After touching the game in games screen and showing the game description in their language, all testers immediately start to read it. It was surprising, yet logical for games that train the brain.

From the results of testing mentioned above, some changes to graphical user interface were done, in the form of animations, recommended games icon changing after played, or automatically adding manual instructions to the game that is played for the first time. Even when asked elderly testers with vision handicaps, font sizes, and overall visibility, they were okay about it and said it was good enough. However, after some testing sessions, the highest achievement was that the testers went to share the application with other people, even though the application still lacked much content. Altogether, the database currently consists of eighteen different user records containing statistics from their progress. This number of records is enough for the exemplary demographics part. The full tester list can be seen in appendix A.

7.3 Cracking up the problems

Implementation is not always going as planned, and while testing, many problems emerge. Situations like this were encountered during the testing phases and commonly fixed immediately. Unfortunately, the problem was sometimes big enough, which could not be fixed right away and took a few days to find a solution. This section discusses these more significant issues and describes the final solution.

7.3.1 Minimal API level

As one of the main issues was changing the first choice for minimal SDK API level, API 19 (Android 4.4 KitKat), to API 21 (Android 5.0 Lollipop). The reason for this was an issue during the implementation of internationalization and localization using LibGDX I18NBundle class. This bundle is responsible for storing and fetching strings of different translations for the app. Notwithstanding, some of the core class methods require the use of API 21. So as the application was designed to provide more language options, the minimal API had to be changed to 21 while accepted the loss of 4% of Android device users (from the picture 5.2).

7.3.2 Fonts

Using font in the app has been imagined as a straightforward thing. However, it is the other way around. Font style, font color, generating font as .fnt file, size of the generated font, font scaling, and camera viewport size are essential for the subsequent use of the fonts in the app.

Choosing the right font style is necessary to fit the app design. Nevertheless, more importantly, the font has to contain the possibility of diacritics language support used in this app, and not many fonts have this option. Altogether generating a font to .fnt file is vital as the LibGDX skin accepts only these types, and it could not be used otherwise. Generating .fnt files is done using the free bitmap font packing tool Hiero, which can be seen in the picture 7.4. Moreover, generating the font in other colors than white disables the option to change font color programmatically in the app, as only white color is interchangeable. Conjointly, the font is scaled depending on the app usage, but the scaling is done setting the fixed size, instead of scaling it in code. Lastly, the camera viewport, mostly the virtual screen size, has a considerable impact on the font zoom, and as imagined, the zoomed font is pixelated.

Fixing all those problems with fonts took a bulk amount of time and is still not perfect yet certainly in a highly acceptable state.

7.3.3 Emulator

Emulator usage is overall not recommended in general by the android developers. With the performance differences that are dependent on the provided computer hardware, some other issues were seen too late after early testing. As the color design has a very high impact on the user, background color, games color, and other app colors have been carefully considered. Trying

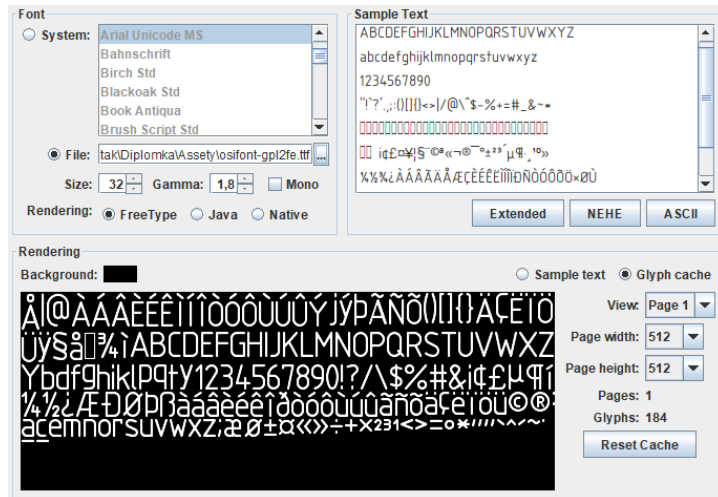


Figure 7.4: Hiero application graphical interface

the right colors was also tested on the emulator as the first simple approach for testing. Unfortunately, this was the wrong choice, as some of the emulator devices had a slightly transparent screen, which made the colors in the background blend with the emulator device screen. The color blending created an illusion effect on the finishing color details when changing the computer background, as the color has never stayed the same, and this confused the author for some time. After recognizing the error, emulators were not trusted for developing a frontend of the app.

7.3.4 Device texture blackouts

The first extensive problem, which affected around half of the mobile devices, was an utterly black texture for in-game animations since the early testing stages. This problem was firstly ignored for a few months as the reality seemed that only a tiny sample of devices is affected, which will be negligible. Unluckily with more testing devices, the problem elevated. Since then, the solution was detected, where OpenGL `GL_buffer` for older devices did not support textures with higher resolution than 2000x2000 px. As the animations are done in high resolution with texture atlas of 4000x1000 px resolution or higher, the `GL_buffer` ignored the texture and drew it as a black box (picture 7.5a). All animations were afterward lowered to the max size of 2000x2000 px and mostly split into more images of this size that has to be loaded separately.

7.3.5 Framebuffer

For the graph module added to scroll pane in the game information screen, a LibGDX Framebuffer class had to be used. Generally, it is a memory buffer containing specified pixel data serving as an object for the app. The framebuffer use is described in the book [63] and has not been understood for a long time as the drawing's behavior was completely different. The sprite batch, used for drawing inside the framebuffer, has inversed drawing, as the [0, 0] coordinates were at the top left corner and not bottom left. Also, to use it in scroll pane, an automatically refreshed Y-axis offset has to be inserted, as the element can not be attached to the LibGDX Scene2D graph building user interface. The book did not mention those changes and left the author confused for a few days.

7.3.6 Display cutout alias Notch

Other visual problems that felt unnecessary until the late stages of development are the Notch device black bars. The testers did not mention it for some time, but as the app came closer to the finished state, the notch owners started to complain about the black bars, as seen in picture 7.5b. Since the screen area is a rare commodity for some, users with notch devices do not have such problems, because the screen surface is extended around essential sensors in the front of the device. Casually this place is unused by many apps and needs to be defined explicitly that the app can extend its screen to use this space. Nevertheless, it means the developer has to figure out the display aspect that adjusts to a new extended screen and can overlap with the front sensors. The app had to be updated to show all information while dodging areas that could be hidden by notch sensors and, therefore, working in a safe area.

7.3.7 Elderly people game test reports

First polished Flying Frog game reviews from elderly users shown some unexpected behavior and misunderstanding. They recognized how the game works and what is expected from them to do, yet many were angry at the first version of functionality.

The first version looked the same as the final one. The difference was when the older user tapped on the screen, mostly around the Y-axis level of where the green arrow was located, the user waited until the character flies through the obstacle. However, the author expected a precise tap, which should force the users to align the character to the center of the obstacle

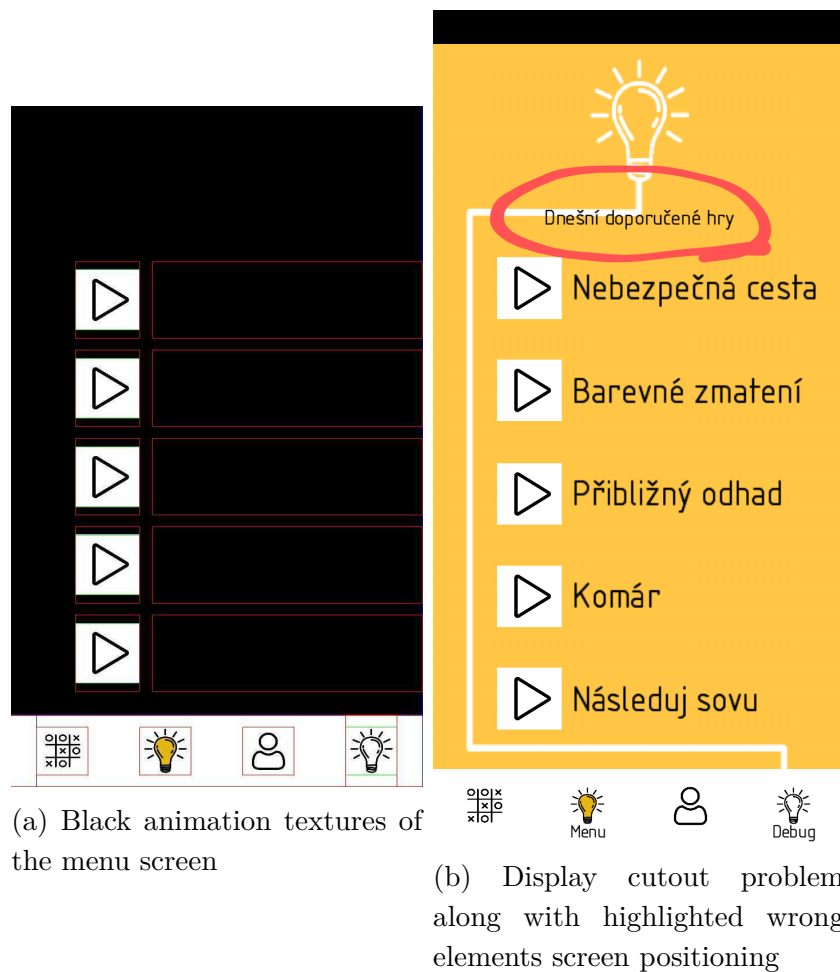


Figure 7.5: Screenshots as error reports from the testers

hole. Many times, the character hit edge to edge of the obstacle, meaning failed reaction, which was very frustrating for the users, since they said they "Tapped the correct spot!".

Ultimately from the replies, two possible solutions for this were devised:

- Dynamically change the hole space between obstacles with difficulty.
- Tapping the Y-axis level or the surrounding, where the center of the hole is, centers the character to the Y-axis level of the hole center.

Asking them what they would imagine happening while envisioning the game's cognitive skill purpose, directed at the result of the second option. Therefore, the game was changed that tapping the surrounding of the hole center, based on the difficulty, centers the character to fly through the hole between two obstacles without problems. Going back to fix, edit, and upgrade this game took more working days overall than every other game.

8 Future improvements

Developing an app while focusing on such a broad range of work leaves many opportunities to improve it in the future. So altogether with already presented new game ideas (Section 4.6), that can be implemented to extend the current app, next mentioned items are ideas for the future.

8.1 Publish app

Not an improvement as a very desirable requirement in the future to fully finish this project is to publish it to Google Play Store. The app itself is ready to be published, yet creating a developer account on Google Play has a requirement of 25\$ as a registration fee. Publishing was discussed with the thesis supervisor, whereas the University of West Bohemia could provide a developer account to publish this app. Unfortunately, conclusions were that a long-term, complicated procedure would accompany the university monetary support for this project. Instead of being concerned about overseeing this process, creating and publishing will be done with a personal account, which is not created to this day. For the time being, the creation of a Google Play developer account is currently on hold, and the author is waiting to see if any on-site research group is going to be interested in the application.

8.2 Email confirmation and forgotten password

Authentication using email allows anyone to create custom login identification to the app using any email of their choice. Sign-in also involves creating a password that is associated with the email. This option can be spammed with unwanted accounts, so this should be protected with some token confirmation sent to email. Using token confirmation also proves that the email is real. Sadly if the user forgets this password, the app does not have any automatic process to recover it, and the user has to log in through another option or send an email to the support asking the for sending a password recovery email. Since all progress data is saved to the local device, none progress data is lost, and they are automatically synchronized with a new logged-in option. If the user chooses to send an email to the support, the admin with access rights to the Firebase console can with provided email

name manually send them a password recovery email, which can be seen in the picture 8.1. Creating an automatic password recovery mechanism for this type of login is another improvement to look forward to.

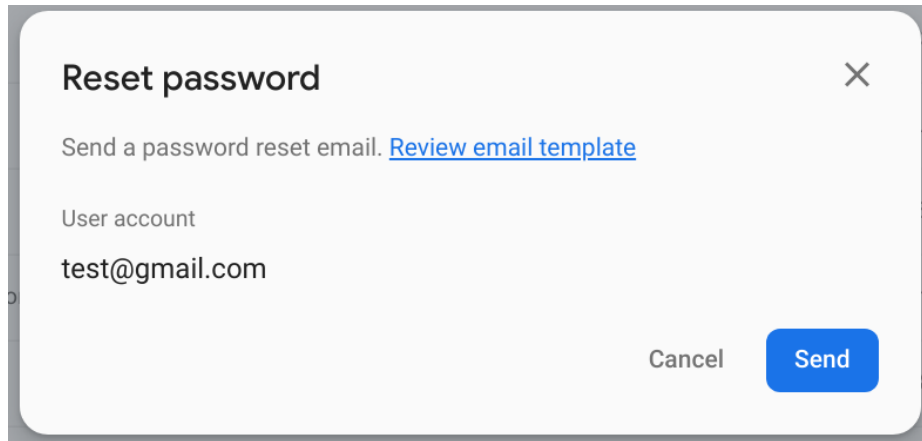


Figure 8.1: Firebase console manual password recovery dialog

8.3 Custom solution to LibGDX Firebase API

Another improvement is to recreate own custom solution to all supportive libraries for Firebase services. Getting rid of these libraries and making own custom solution weights a medium priority, as it is still a huge amount of work for one person team. Being reliant on other not so popular libraries which addresses the cross-platform API for Firebase is not suitable for the future app development. Making own solution would make the app more effective and even customizable, because Firebase service methods could be focused on the app specific needs. Nevertheless this upgrade will not have a high priority, due to the fact that the time estimate to develop such solution will approximately correspond to half-time spent on the developing of this whole app.

8.4 iOS launch

Using LibGDX framework to develop this app was a decision not only made by the authors own preferences, but cross-platform possibilities were also taken into consideration. The main code is like a core to all other platforms united under one API. However, every platform has its own specific needs that must be defined in its interface. Even though the app was not developed on Desktop platform, Android and Desktop platforms are easy

to deploy right from the source code with a very little to none problems at launch. Unfortunately deploying and launching an iOS app needs access to Xcode development environment, which is available only for macOS operating system. Altogether there are two different backends available for deploying LibGDX app to iOS, and without testing, it is uncertain which one is better to use. As in the current state, the author does not have access to a macOS device nor any other Apple device to test it on. It is thus making it impossible to release the app without proper testing, as new iOS-specific errors could arise.

8.5 Web application

An alternative to present the statistics for the user from the app by using a web application. This idea was created in the mid-stages of development, but unfortunately, there was not enough time to create a whole web page with authentication and visualisation by using graphs for presenting the data. The good thing about this is that all data can be acquired from Firebase database, which has already well-documented API. Creating a web application should then be a convenient improvement to increase popularity among the people as this could raise promoting of the app.

8.6 Data sharing and extended data gathering

Gathering of the information from the games is currently done only from a logged-in user. This could be changed that even anonymous users if connected to the internet, would send their anonymous data to the database. This idea is just experimental, yet it would be highly beneficial for the whole statistics summaries and possible uses of other scientific projects. With this goes hand in hand providing a public API for database data with private key permission. A researcher could request access to the database, who would be granted access once proving the data handling qualification. Requesting access to the database should be done through the previously mentioned website.

9 Summary

The analysis of the existing cognitive training applications has proven that the most popular ones are very commercial based. It confirms that these applications are mostly made by teams that are doing it for financial profit, although it is understandable that teams creating those high-quality apps will not do it for free. Concurrently none of the apps is supported the Czech language, which can repel the usage of cognitive training apps in that country. Based on this knowledge, an entirely free app, Czech localization, and public API are the options that are vital for a new cognitive training app.

After the analysis, possible training games with general functionality and game scenarios were designed. The games are separated into five categories, which fits as closely as possible to the core fields of cognitive training. Some games are based on highly rated medical-psychological publications that invented specific cognitive training tasks (e.g., The Stroop Color and Word Test). The scenarios describe how to game should act and how the user statistics for the game are calculated.

From the design, the games were subsequently implemented as part of the free cognitive training app, which is created using a trendy open-source, cross-platform framework with extensions for database connection and physics engine. The app comprises of its log-in option, which allows the user to compare his statistics with all logged-in users within the mobile app. On the contrary, the user stores all his data locally to see his local improvement. Furthermore, the app recommends daily games to play, yet allows the user to play any game. Most of the app graphics and sounds are own made along with all animations that are deepening the user experience. The Firebase cloud-hosted NoSQL database was selected to store the users' information and game statistics.

Mini-scenarios test the app with stated inputs and all possible outputs. During the development, continual testing was also done by giving tasks to a wide variety of testers. Continual testing helped to fix some widespread problems, which affected a broad range of mobile devices. Overall the developed app fits all required functionalities and works as intended. In the current state, some improvements for the future are suggested, and the solution can be used for practicing the cognitive functions of any age group.

List of abbreviations

ADHD	Attention deficit hyperactivity disorder
API	Application Programming Interface
BaaS	Backend as a service
COVID19	Coronavirus disease 2019
GB	Gigabyte
GiB	Gibibyte
GUI	Graphical user interface
HW	Hardware
IT	Information technology
IQ	Intelligence quotient
MBI	Mensa brain index
OS	Operating system
POJO	Plain Old Java Object
SW	Software
UI	User interface
UUID	Universally unique identifier

Bibliography

- [1] POUSHTER, J. – BISHOP, C. – CHWE, H. *Social Media Use Continues To Rise in Developing Countries, but Plateaus Across Developed Ones* [online]. Pew Research Center, 2018. [cit. 2020/04/17]. Available from: https://www.pewglobal.org/wp-content/uploads/sites/2/2018/06/Pew-Research-Center_Global-Tech-Social-Media-Use_2018.06.19.pdf.
- [2] ZHAO, J. – FREEMAN, B. – LI, M. *Can Mobile Phone Apps Influence People's Health Behavior Change? An Evidence Review* [online]. J Med Internet Res, 2016. [cit. 2020/04/17]. Available from: <https://doi.org/10.2196/jmir.5692>.
- [3] BYAMBASUREN, O. – BELLER, E. – GLASZIOU, P. *Current Knowledge and Adoption of Mobile Health Apps Among Australian General Practitioners: Survey Study* [online]. JMIR Mhealth Uhealth, 2019. [cit. 2020/04/20]. Available from: <https://mhealth.jmir.org/2019/6/e13199>.
- [4] J., C. *Leading health and fitness apps in the Google Play Store worldwide in March 2020, by number of downloads* [online]. Statista, 5 2020. [cit. 2020/05/05]. Available from: https://www.pewglobal.org/wp-content/uploads/sites/2/2018/06/Pew-Research-Center_Global-Tech-Social-Media-Use_2018.06.19.pdf.
- [5] *Mobile Fact Sheet* [online]. Pew Research Center, 2019. [cit. 2020/04/20]. Available from: <https://www.pewresearch.org/internet/fact-sheet/mobile/>.
- [6] *Internet/Broadband Fact Sheet* [online]. Pew Research Center, 2019. [cit. 2020/04/20]. Available from: <https://www.pewresearch.org/internet/fact-sheet/internet-broadband>.
- [7] HILL, N. L. et al. *Feasibility study of an attention training application for older adults* [online]. International Journal of Older People Nursing, 2015. [cit. 2020/04/20]. Available from: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/opn.12092>.
- [8] TURNER, A. M. et al. *A Closer Look at Health Information Seeking by Older Adults and Involved Family and Friends: Design Considerations for Health Information Technologies* [online]. AMIA Symposium, 2018. [cit. 2020/04/20]. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371280/>.

- [9] *A Consensus on the Brain Training Industry from the Scientific Community* [online]. Max Planck Institute for Human Development and Stanford Center on Longevity, 2014. [cit. 2020/04/21]. Available from: <https://www.cognitivetrainingdata.org/the-controversy-does-brain-training-work/stanford-letter/>.
- [10] *Cognitive Training Data Response Letter* [online]. Parnassus Publishing, 2014. [cit. 2020/04/21]. Available from: <https://www.cognitivetrainingdata.org/the-controversy-does-brain-training-work/response-letter/>.
- [11] *How to tell the difference between a fun-but-useless “brain game” and an effective, worthwhile cognitive training program* [online]. Parnassus Publishing, 2015. [cit. 2020/04/22]. Available from: <https://www.cognitivetrainingdata.org/how-to-evaluate-a-brain-training-program/>.
- [12] GOGHARI, V. M. et al. *Attitudes and Beliefs Toward Computerized Cognitive Training in the General Population* [online]. *Frontiers in Psychology*, 2020. [cit. 2020/04/21]. Available from: <https://www.frontiersin.org/article/10.3389/fpsyg.2020.00503>.
- [13] VERGANI, L. et al. *Training Cognitive Functions Using Mobile Apps in Breast Cancer Patients: Systematic Review* [online]. *JMIR Mhealth Uhealth*, 2019. [cit. 2020/04/17]. Available from: <https://doi.org/10.2196/10855>.
- [14] TACCHINO, A. et al. *A New App for At-Home Cognitive Training: Description and Pilot Testing on Patients with Multiple Sclerosis* [online]. *JMIR mHealth uHealth*, 2015. [cit. 2020/04/22]. Available from: <https://doi.org/10.2196/mhealth.4269>.
- [15] RUIZ-MANRIQUE, G. – TAJIMA-POZO, K. – MONTAÑES-RADA, F. *Case Report: “ADHD Trainer”: the mobile application that enhances cognitive skills in ADHD patients* [online]. *F1000Research*, 2015. [cit. 2020/04/22]. Available from: <https://doi.org/10.12688/f1000research.5689.3>.
- [16] BEEK, J. J. W. et al. *Tablet App Based Dexterity Training in Multiple Sclerosis (TAD-MS): Research Protocol of a Randomized Controlled Trial* [online]. *Frontiers in Neurology*, 2019. [cit. 2020/04/22]. Available from: <https://doi.org/10.3389/fneur.2019.00061>.
- [17] AREAN, P. A. et al. *The Use and Effectiveness of Mobile Apps for Depression: Results From a Fully Remote Clinical Trial* [online]. *J Med Internet Res*, 2016. [cit. 2020/04/22]. Available from: <https://doi.org/10.2196/jmir.6482>.

- [18] CROISILE, B. *Memory stimulation. Which scientific benefits? Which exercises?* [online]. La Revue de Gériatrie (French Geriatrics Journal), 2006. [cit. 2020/03/09]. Available from: <http://www.happy-neuron.com/rsc/hn4/docs/French%20Geriatrics%20Journal%20-%20Dr.%20B%20Croisile%20Publication.pdf>.
- [19] KLUCKÁ, J. – VOLFOVÁ, P. *Kognitivní trénink v praxi*. Grada Publishing, 2009. ISBN 978-80-247-5580-9.
- [20] ATKINSON, R. C. – SHIFFRIN, R. *Human Memory: A Proposed System and its Control Processes* [online]. Elsevier, 1968. [cit. 2020/02/28]. Available from: <https://www.sciencedirect.com/science/article/pii/S0079742108604223>.
- [21] MILLER, G. A. *The magical number seven, plus or minus two: some limits on our capacity for processing information*. [online]. Psychological Review, 1956. [cit. 2020/03/20]. Available from: <https://psycnet.apa.org/record/1957-02914-001>.
- [22] SQUIRE, L. R. – KNOWLTON, B. – MUSEN, G. *The Structure and Organization of Memory* [online]. Annual Review of Psychology, 1993. [cit. 2020/04/09]. Available from: <https://www.annualreviews.org/doi/pdf/10.1146/annurev.ps.44.020193.002321>.
- [23] TULVING, E. – DONALDSON, W. – BOWER, G. H. *Organization of memory*. Academic Press, 1972. Available from: http://alumni.media.mit.edu/~jorkin/generals/papers/Tulving_memory.pdf. ISBN 9780127036502.
- [24] SOHLBERG, M. M. – MATEER, C. A. *Introduction to cognitive rehabilitation: theory and practice*. Guilford Press, 1989. Available from: <https://archive.org/details/introductiontoco00sohl>. ISBN 978-0898627381.
- [25] KULIŠŤÁK, P. *Neuropsychologie*. Portál, 2003. ISBN 978-80-7367-891-3.
- [26] OPENCLIPART-VECTORS. *Rubik's cube* [online]. Pixabay. [cit. 2020/04/12]. Available from: https://cdn.pixabay.com/photo/2013/07/13/10/21/rubiks-cube-157058_960_720.png.
- [27] R., J. *Tangram* [online]. FreeImages. [cit. 2020/04/12]. Available from: <https://www.freeimages.com/photo/tangram-1-1422588>.
- [28] HIJAZI, M. M. K. *Attention, Visual Perception and their Relationship to Sport Performance in Fencing* [online]. Journal of human kinetics, 2013. [cit. 2020/03/28]. Available from: <https://doi.org/10.2478/hukin-2013-0082>.

- [29] CLIFTON, C. et al. *Language Comprehension and Production* [online]. Comprehensive handbook of psychology, 2013. [cit. 2020/04/09]. Available from: https://www.researchgate.net/publication/269700690_Language_Comprehension_and_Production.
- [30] MARSLÉN-WILSON, W. D. – WELSH, A. *Processing interactions and lexical access during word recognition in continuous speech* [online]. Cognitive Psychology, 1978. [cit. 2020/04/09]. Available from: [https://doi.org/10.1016/0010-0285\(78\)90018-X](https://doi.org/10.1016/0010-0285(78)90018-X).
- [31] *Written And Spoken Language* [online]. HAPPYneuron. [cit. 2020/04/09]. Available from: <http://www.happy-neuron.com/brain-and-training/language>.
- [32] DIAMOND, A. *Executive Functions* [online]. Annual Review of Psychology, 2013. [cit. 2020/04/12]. Available from: <https://doi.org/10.1146/annurev-psych-113011-143750>.
- [33] HOFMANN, W. – SCHMEICHEL, B. J. – BADDELEY, A. D. *Executive functions and self-regulation* [online]. Trends in Cognitive Sciences, 2012. [cit. 2020/04/12]. Available from: <https://doi.org/10.1016/j.tics.2012.01.006>.
- [34] MIYAKE, A. et al. *The Unity and Diversity of Executive Functions and Their Contributions to Complex “Frontal Lobe” Tasks: A Latent Variable Analysis* [online]. Cognitive Psychology, 2000. [cit. 2020/04/12]. Available from: <https://doi.org/10.1006/cogp.1999.0734>.
- [35] *About Lumosity* [online]. Lumos Labs, 2019. [cit. 2019/08/19]. Available from: <https://www.lumosity.com/en/resources>.
- [36] *Lumosity Android* [online]. Lumos Labs. [cit. 2019/08/19]. Available from: <https://www.lumosity.com/en/resources/>.
- [37] *CogniFit* [online]. CogniFit, 2019. [cit. 2019/08/20]. Available from: <https://www.cognifit.com>.
- [38] *About Elevate* [online]. Elevate, 2019. [cit. 2019/08/20]. Available from: <https://www.elevateapp.com/about>.
- [39] *Press info about NeuroNation* [online]. Berliner Synaptikon GmbH, 2019. [cit. 2019/08/20]. Available from: <https://sp.neuronation.com/en/press/>.
- [40] *Mensa Brain Training* [online]. Barnstorm Games Limited, 2020. [cit. 2020/03/11]. Available from: <https://apps.apple.com/us/app/mensa-brain-training/id768606792>.

- [41] *Flappy Bird* [online]. IGN, 2013. [cit. 2020/05/08]. Available from: <https://www.ign.com/games/flappy-bird>.
- [42] GAMING, N. *2 Cars: Simple Yet Addictive* [online]. RedBull, 2014. [cit. 2020/05/08]. Available from: <https://www.redbull.com/in-en/2-cars-simple-yet-addictive>.
- [43] KESSELS, R. et al. *The Corsi Block-Tapping Task: Standardization and Normative Data* [online]. Applied neuropsychology, 2000. [cit. 2020/05/08]. Available from: https://doi.org/10.1207/S15324826AN0704_8.
- [44] SCARPINA, F. – TAGINI, S. *The Stroop Color and Word Test* [online]. Frontiers in Psychology, 2017. [cit. 2020/05/08]. Available from: <https://www.frontiersin.org/article/10.3389/fpsyg.2017.00557>.
- [45] ERIKSEN, B. A. – ERIKSEN, C. W. *Effects of noise letters upon the identification of a target letter in a nonsearch task* [online]. Perception & Psychophysics, 1974. [cit. 2020/06/17]. Available from: <https://doi.org/10.3758/BF03203267>.
- [46] *Mobile Operating System Market Share Worldwide* [online]. StatCounter, 3 2020. [cit. 2020/04/23]. Available from: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.
- [47] *Distribution dashboard* [online]. Google Developers, 4 2020. [cit. 2020/04/23]. Available from: <https://developer.android.com/about/dashboards>.
- [48] *Mobile & Tablet Android Version Market Share Worldwide* [online]. StatCounter, 3 2020. [cit. 2020/04/23]. Available from: <https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide>.
- [49] *Firebase* [online]. Google Developers, 2020. [cit. 2020/05/02]. Available from: <https://firebase.google.com>.
- [50] CARBONNELLE, P. *TOPDB Top Database index* [online]. 2019. [cit. 2020/05/02]. Available from: <http://pypl.github.io/DB.html>.
- [51] *Pricing plans* [online]. Google Developers, 2020. [cit. 2020/05/02]. Available from: <https://firebase.google.com/pricing>.
- [52] LARDINOIS, F. *Kotlin is now Google's preferred language for Android app development* [online]. TechCrunch, 5 2019. [cit. 2020/05/02]. Available from: <https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/>.

- [53] *Developer Survey Results* [online]. Stack Overflow, 2019. [cit. 2020/05/02]. Available from: <https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted>.
- [54] ZECHNER, M. *Setting up Environment* [online]. 2013. [cit. 2020/20/07]. Available from: <https://libgdx.badlogicgames.com/documentation/gettingstarted/Setting%20Up.html>.
- [55] *Add Firebase to your Android project* [online]. Google Developers, 2020. [cit. 2020/20/07]. Available from: <https://firebase.google.com/docs/android/setup>.
- [56] MK-5. *GDX Fireapp* [online]. 2020. [cit. 2020/20/07]. Available from: <https://github.com/mk-5/gdx-fireapp>.
- [57] HAZLEWOOD, L. *Java JWT: JSON Web Token for Java and Android* [online]. 2020. [cit. 2020/22/07]. Available from: <https://github.com/jwtk/jjwt>.
- [58] *Create Custom Tokens* [online]. Google Developers, 2020. [cit. 2020/22/07]. Available from: <https://firebase.google.com/docs/auth/admin/create-custom-tokens>.
- [59] RIBON, A. *Physics Body Editor* [online]. [cit. 2020/07/31]. Available from: <https://code.google.com/archive/p/box2d-editor/>.
- [60] HUTCHINGS, S. *Typicons* [online]. [cit. 2020/07/30]. Available from: <https://github.com/stephenhutchings/typicons.font>.
- [61] HIKIKOMORI82. *Osifont* [online]. [cit. 2020/07/30]. Available from: <https://github.com/hikikomori82/osifont>.
- [62] RAELEUS. *Skin Composer* [online]. [cit. 2020/07/31]. Available from: <https://github.com/raeleus/skin-composer>.
- [63] NAIR, S. – OEHLKE, A. – SAFARI, a. O. M. C. *Learning LibGDX Game Development - Second Edition*. Packt Publishing, 2015. ISBN 9781783554775.

Appendices

A List of all testers

The technological skill level is scaled as:

- **High**

Installs and launch the app all alone. Gives detailed responses to visual design, functions, and performance in the app. Sometimes provides help with bug fix solution.

- **Medium**

Can install and launch the app without little to no guidance. Gives response to visual design and functions of the app. Sometimes provides screenshots with display bugs with little information about it.

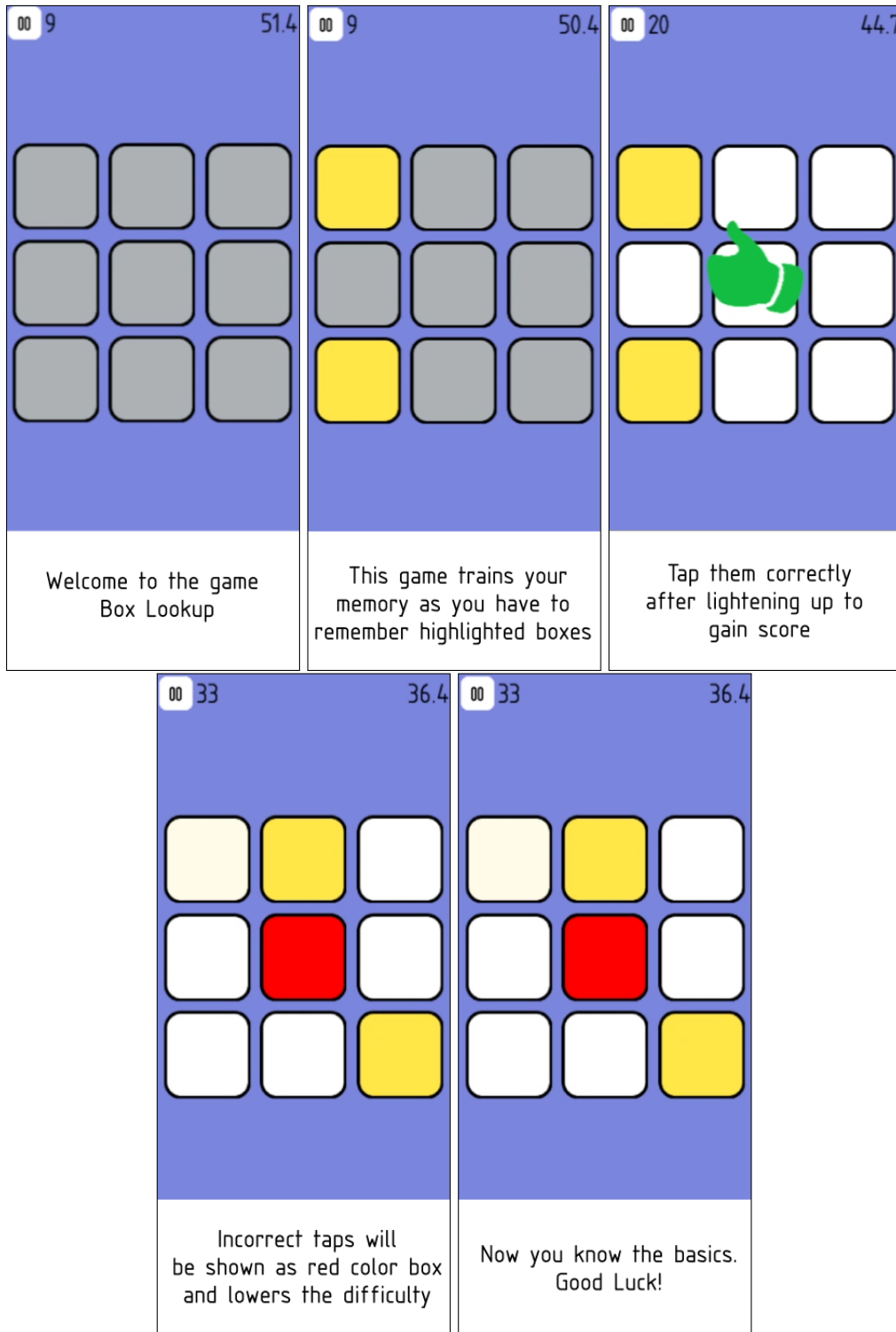
- **Low**

Needs full guidance with *.apk installation to app launch. Puts a high-value opinion on app animations and visual elements than on functionality.

Age category	Gender	Mobile device	Log-in provider	First app test	Technological skill level
20-30	Male	Samsung Galaxy S6 Edge	Google, Facebook, Email	Mar 17, 2020	High
20-30	Male	Motorola One Lite	Google	Apr 16, 2020	High
20-30	Male	Huawei P20 Lite	Google	Apr 16, 2020	High
20-30	Male	Sony Xperia Z3 Compact	Google	Apr 20, 2020	High
40-50	Female	Samsung Galaxy A10	Google	Apr 22, 2020	Medium
50-60	Male	Blackview BV6000	Google	Apr 30, 2020	Low
20-30	Female	Xiaomi Redmi Note 7	Google	May 2, 2020	Medium
70-80	Male	Samsung Galaxy S4	Google	Jun 28, 2020	Medium
20-30	Female	Samsung Galaxy J5	Email	Jun 28, 2020	Medium
10-20	Female	Huawei P9 Lite Mini	Google	Jul 3, 2020	Medium
40-50	Male	Xiaomi Pocophone F1	Email	Jul 7, 2020	High
20-30	Female	Sony Xperia L1	Google	Jul 7, 2020	Low
10-20	Female	Huawei Y6 II compact	Google	Jul 8, 2020	Medium
20-30	Male	Huawei P9 Lite	Google	Jul 12, 2020	Medium
20-30	Male	Huawei Honor 7 Lite	Google	Jul 17, 2020	Medium

B Games documentation

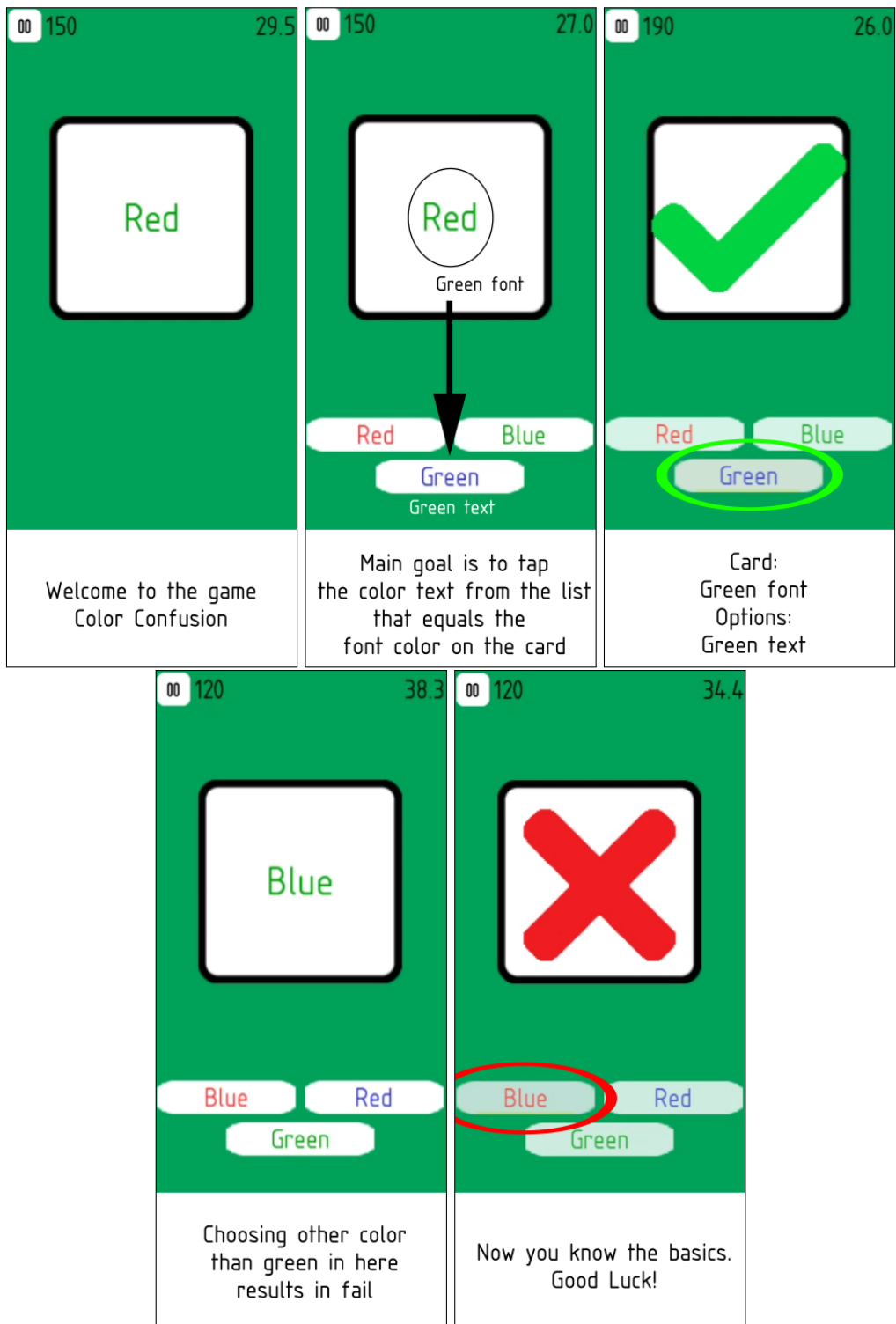
Box lookup



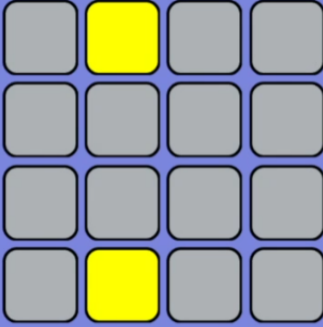
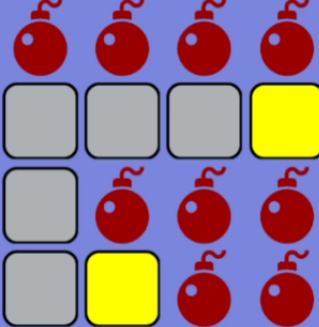
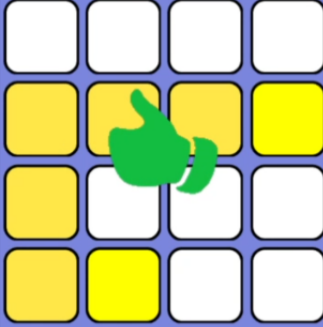
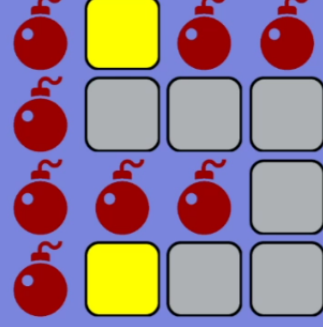
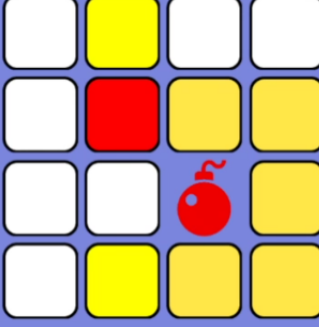
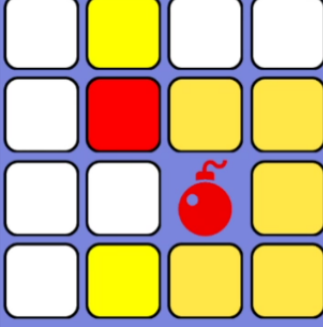
Bubbles

<p>00 0 57.6</p> <p>22 target 0 current sum</p> <p>Welcome to the game Bubbles</p>	<p>00 0 57.6</p> <p>22 target 0 current sum</p> <p>Bubbles are falling from the sky (line). You must tap the bubbles to sum up to target</p>	<p>00 0 55.0</p> <p>22 target 21 current sum</p> <p>Current sum of all tapped bubbles is shown on the other side of the target</p>
<p>00 210 51.1</p> <p>13 target 0 current sum</p> <p>Summed bubbles dissappear when they are equal to the target.</p>	<p>00 210 51.1</p> <p>13 target 0 current sum</p> <p>Do not let bubbles overflow through the sky!</p>	<p>00 210 51.1</p> <p>13 target 0 current sum</p> <p>Now you know the basics. Good Luck!</p>

Color confusion



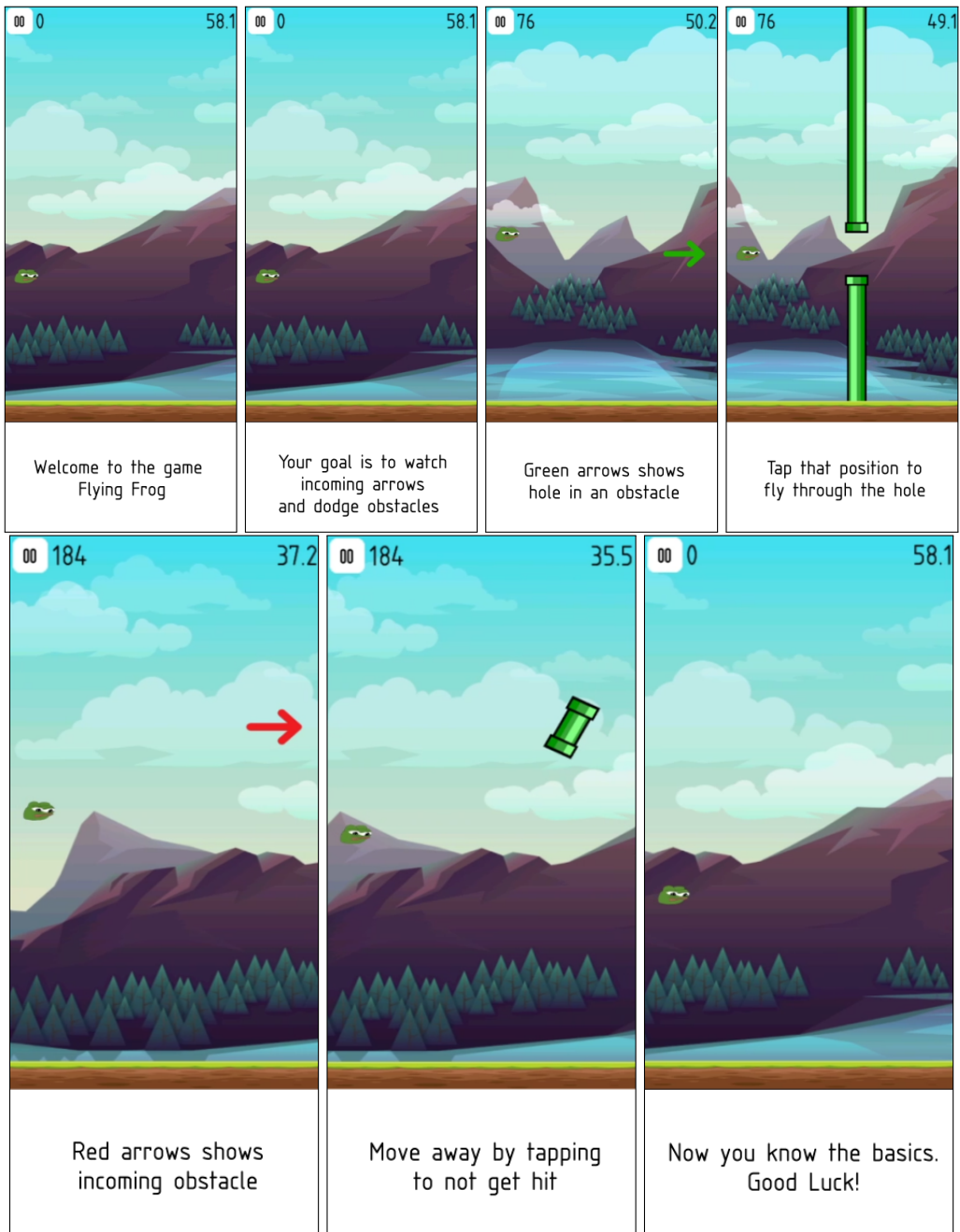
Dangerous path

<p>00 60 51.4</p>  <p>Welcome to the game Dangerous path</p>	<p>00 0 57.2</p>  <p>Your main goal is to remember the path between highlighted points</p>	<p>00 60 52.5</p>  <p>Then draw it as you remebered</p>
<p>00 60 49.0</p>  <p>If you forget the path..</p>	<p>00 125 40.9</p>  <p>you will explode, which will show you the missed places in path</p>	<p>00 125 40.9</p>  <p>Now you know the basics. Good Luck!</p>

Estimation



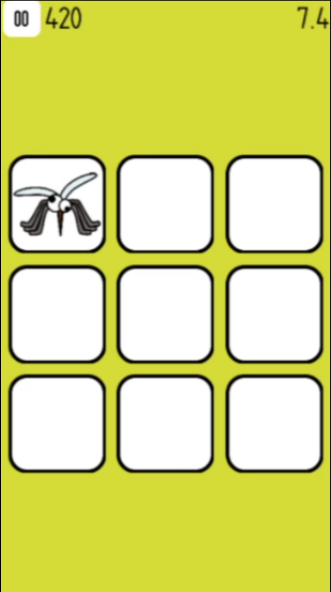
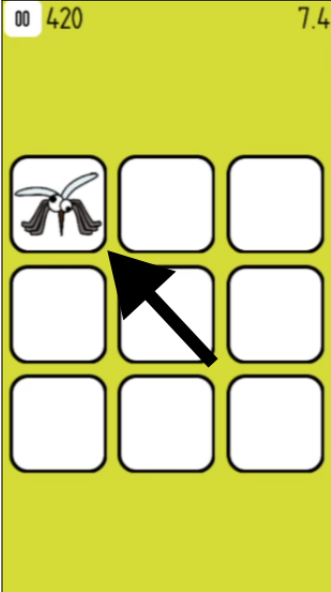
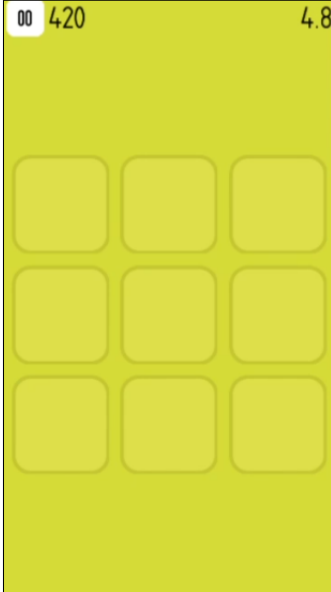



Flying frog




Follow the owl

<p>Welcome to the game Follow the Owl</p>	<p>You have to memorize all places, where the owl has appeared</p>	<p>You have to memorize all places, where the owl has appeared</p>
<p>You have to memorize all places, where the owl has appeared</p>	<p>And then repeat the sequence by tapping them in the correct order</p>	<p>Now you know the basics. Good Luck!</p>

Hiding mosquito

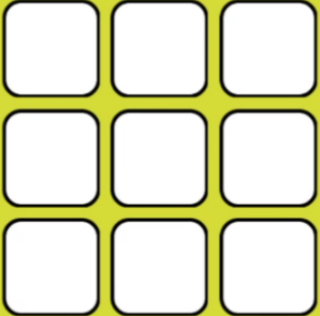
 <p>Welcome to the game Mosquito</p>	 <p>The first goal is to remember where the mosquito appears</p>	 <p>After hiding, watch the directions that shows you where the mosquito goes</p>
 <p>And remember all of them!</p>	 <p>And remember all of them!</p>	 <p>And remember all of them!</p>

00 420 0.0




And remember all of them!

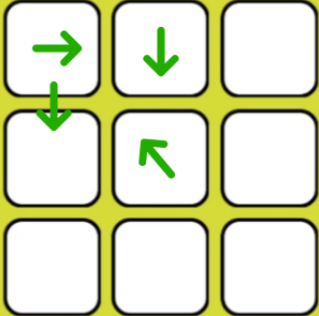
00 420 0.0



From the memory,
I remember those positions




00 420 0.0



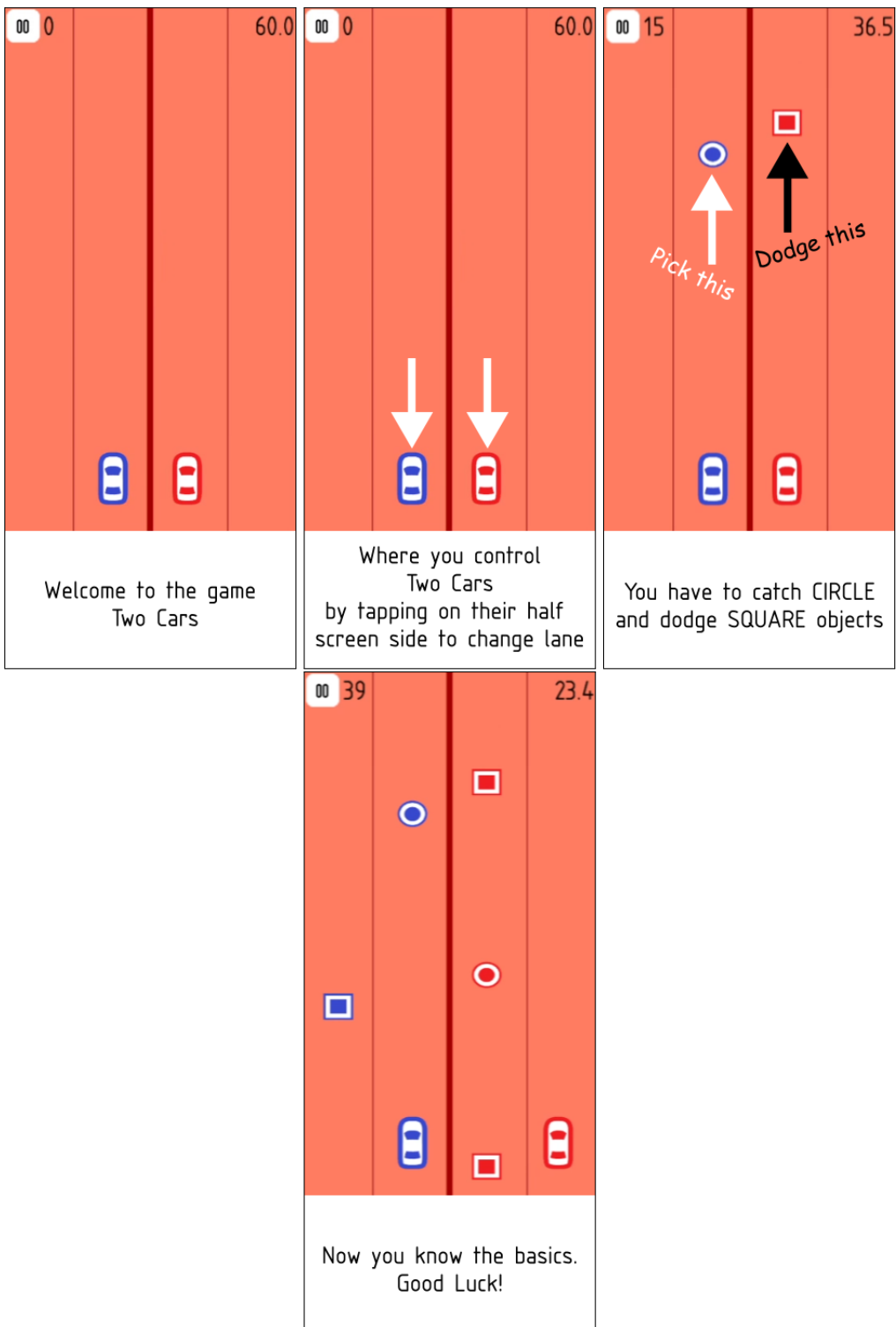
Apply them imaginary to
the grid and tap the
mosquito landing position

00 650 0.0



You smashed it!
Now you know the basics.
Good Luck!

Two cars



C Contents of attached DVD

The attached DVD contains source files, assets used in the app, graphic programs save for handmade images, sounds, poster files, and LaTeX source files. Furthermore, the app's source code is also available in the online GitLab repository ¹. DVD hierarchy is as follows:

```
|_ Blik.apk ..... Android .apk file of the application
|_ documentation/ ..... All LaTeX files packaged into ZIP file
|_ poster/ ..... PDF and PUB poster files
|_ sources/ ..... Project source files
  |_ app/ ..... Application source code files
  |_ assets/ ..... All assets used in the application
    |_ fonts/ ..... Font files
    |_ graphics/ ..... Graphical files
    |_ locales/ ..... Locale files
    |_ sounds/ ..... Sound files
  |_ graphics_saves/ ..... All graphical programs saves
```

¹<https://gitlab.com/Lukado/blik>