[[1]](#footnote-1)

A Technical Overview of “Calculot”

Kenneth Zheng, Levent Kylych, Ryan Peterson, Jean Pierre Acebedo  
Simon Fraser University  
British Columbia, Canada  
[kza21, lkylych, repeters, jacebedo] @sfu.ca

*Abstract*— This document focuses on the software development of Group 14’s application, “Calculot.” In retrospective, this document will focus on the techniques, software packages used, constraints and limitations of the development of Calculot. As this is a simple application consisting of a game with simple learning topics, the technicality involved in the document will be considered as a document that concerns a high-level understanding of software development and implementation techniques.

# INTRODUCTION

“Calculot” is an application designed for helping students in understanding, learning and practicing different high school and university level topics. The idea was to create an application that will make learning math fun and enjoyable for everyone.

Main part of an application are the two games named “Defense of Calculot” and “Wizard Crystal Ball”.

In “Defense of Calculot” user required to defend against the forces of Calculot by solving derivatives and/or integrals. This game should help users to get familiar with different types of derivatives and integrals by solving more and more difficult ones. Fixed amount of health points will encourage students to choose answer carefully, while monster, approaching wizard's tower will make users to choose right answer as fast as possible.

In “Wizard Crystal Ball” user required to solve different questions about vectors, trigonometry and complex numbers to power up the Wizard's Crystal Ball. This game helps users not only to get better understanding of vectors, complex numbers and trigonometry, but also to develop a skill of solving questions on the given topics in limited amount of time. The game designed in the such way that more times you choose correct answer, the bigger is your Crystal Ball becomes . Each wrong answer will

decrease size of the Wizards' Crystal Ball. As result, the more experience user have, more time hi or she will be able to play without failing the game and get more and more fun.

Besides games, other features which should help user and make his or her experience more comfortable was implemented.

Sign up activity will help new users to create a profile. Users are required to provide their first and last name, and create a password. “Experienced” users are able to asses their profile using Login activity. In Login activity user are able to see profiles of people who already signed up. “Experienced” user can find their his or her profile and login using password, created during Sign up. Data about users and their profiles is saved in database.

To encourage users to play our game more and more such parameters as experience and level was added. Users are able to get experience by playing one of the two games on any of provided topics. More experience user get, higher his level becomes. Also user is able to get achievements for getting higher levels (for example achievement for reaching level 5, or achievement for reaching level 10). Achievements should also encourage users to play more, to get more experience and more achievements.

As “Calculot” is designed for people with different level of knowledge and understanding of math, not everyone will be able to start playing games immediately after signing up, because some level of knowledge is necessary to be able to answer questions in games. For such students Learning activity is designed. There users are able to find five major topics, and choose the one user is interested in. Each topic have a YouTube player with educational video on this topic and several questions about this topic and video, to test what knowledge user got. After completing all questions in a given topic, user should be ready to start playing the game on this topic. If user answered all question about the topic correctly, ho or she will be suggested to go back to main menu to play a game, learn some other topic, or check his progress on the profile page.

As result “Caculot” help student to learn something new, test their knowledge, and then get practice on the knowledge student got. Games, experience, and achievements make this process easy and fun even for people, who always were finding math to be a very difficult subject.

# background

## Topics Covered

As the target audience of "Calculot" are students in late high school to students in their first two years of university interested in or studying the discipline of mathematics, the topics covered include differential and integral calculus, as well as vectors, complex numbers, polar coordinates and their interrelationship with each other. In addition to the learning sections which introduce and test to an extent the understanding of key concepts, a game section serves to motivate the user to continue learning and mastering key principles in various subjects.

The game "Defence of Calculot," testing for the rapid calculation of derivatives and integrals, is aimed at driving the user to improve upon and eventually master their ability to think fast and accurately. The tangible benefits of this being of course, arguably higher test scores on academic situations as the users are able to spend less time with more accuracy on the routine calculations of simple derivatives and integrals while being less susceptible to confusion in the differences between the two.

In contrast, the second game "Wizard's Crystal Ball" aims to provide a deeper understanding between vectors, complex numbers, and polar coordinates and their highly integrated nature. The intent of the game is to not only improve the user’s intuitive understanding of the aforementioned topics but also to help the user gain an appreciation for an attention to detail in time pressure situations.

## Theme

“Calculot” is based around a fantasy world with an aesthetic basis in the medieval style. The app name “Calculot” is derived from the fictional realm of “Camelot” which is associated with the legendary King Arthur. The game “Defence of Calculot” engages the user by immersing them into the world of “Calculot” where the user plays as a wizard above a stone tower and must defend it from incoming enemies. The “Wizard’s Crystal Ball” recaptures some elements of that high-pressure environment by presenting the user with the task of generating the various shell layers which compose a crystal ball by answering timed questions to progress to higher levels. The visual presentation of the crystal ball also adds an element of magic to the medieval style artwork.

## Random Question Generation in “Magician’s Crystal Ball”

Both games “Defence of Calculot” and “Wizard’s Crystal Ball” utilize a multiple choice random question generator to generate their respective questions and answers along with a minimally interactive visual component to tie together the technical aspect with the theme. This section focuses on the generation of questions in “Magician’s Crystal Ball.”

Upon generation of a question, a series of actions occur before the possible answers are presented. First, the VectorQuestionGenerator class must randomly generate a vector. This includes the x as well as y component value. The values for the angle theta and norm are then generated based on the generated x and y values to keep the values consistent.

After generation of the question vector, there are many forms in which a randomly generated question may have. They are presented as follows:

Where: x, y are the randomly generated components of the vector: v = (x,y)

Re,Im corresponds to the real and imaginary components of the complex number c = Re + Im which describes the vector v.

Theta, radius corresponds to the polar form (radius, theta) of the terminal point of the vector v.

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Find | Given Components | Answer form (shown using vector components only): |
| 0 | radius/norm/mod-ulus | Re/x, Im/y | sqrt( x^2 + y^2 ) |
| 1 | theta | Re/x, Im/y | arctan( y / x ) |
| 2 | Im/y | modulus/norm, Re/x | sqrt( norm^2 - x^2 ) |
| 3 | theta | modulus/norm, Re/x | arccos( x / norm ) |
| 4 | Re/x | modulus/norm, Im/y | sqrt( norm^2 - y^2 ) |
| 5 | theta | modulus/norm, Im/y | arcsin( y / norm ) |
| 6 | Re/x | theta, radius | norm \* cos( theta ) |
| 7 | Im/y | theta, radius | norm \* sin( theta ) |
| 8 | radius/norm/mod-ulus | theta, Im/y | y / sin( theta ) |
| 9 | Re/x | theta, Im/y | y / tan( theta ) |
| 10 | radius/norm/mod-ulus | theta, Re/x | x / cos( theta ) |
| 11 | Im/y | theta, Re/x | x \* tan( theta ) |

In total this gives us 44 unique question forms with each form having randomly generated components.  
Although the answer forms are technically limited to twelve, with an answer form being a certain combination of operations on various components such as a trigonometric function, each answer form is generated randomly in such a way that it’s presented structure varies greatly.

## Random Answer Generation in “Magician’s Crystal Ball”

The method “generateMediumAnswerArray” is called which is responsible for generating the random answers and checking for the duplication, albeit unlikely, of random answers. An array is instantiated with the array size representing the total number of answers generated including the correct answer. The size of the array, and with it the number of choices as answers, varies in relation to difficulty tracked by the level number. Upon the call of this method, a random integer between zero and the size of the array minus one is generated to select the index of the correct answer. A for loop is then triggered to cycle through each empty address in the array and populate it with either a correct answer or an incorrect answer, with both having randomly generated structures. When cycling through the loop, the index of the loop is checked, if it is the selected location of the correct answer, another method, “generateMediumRightAnswer” is called to generate such an answer. If the address is not the selected location of the correct answer then a do while loop is called. In the do while loop, a temporary wrong answer is generated by calling another method “generateMediumWrongAnswer”. This wrong answer is checked with the rest of the array to see that it is unique. If so, then it is put into the next empty location in the array, otherwise the method is called until a unique answer is found.

Random generation of answers is achieved through the use of interdependent modular methods.

Typically, the answers can be broken up into two base strings. The base strings are set with the string representation of the given components. For example, in case 1 (Vector components given) where the user is asked to find Theta given the X and Y component values. The answer form to be generated must be equivalent to arctan(y/x). The base strings ComponentA and ComponentB would be set to the string representation of Y and X respectively. Now a series of applications of modular methods are applied to the string to generate the final version of the random correct answer. In this particular case, the modular methods used include applyPower, commutativeOperation, and applyTrig as well as decide complexity. First, the applyPower method is used to return a value for ComponentA. This introduces variability in the representation of the given component values. This variability means that the component may be raised to a fraction or decimal representation of the intended power. In this case it is one. The fraction may be further scaled by any low scalar from one to four or multiplies of ten to keep it relatively simple. Note that all calls of the applyPower method also uses the decideComplexity boolean method. This effectively controls the amount of variability in answer form which takes into account level and how many components are already going to be randomized. The higher the level, the more likely it is to reform the correct answer. However, the more components to be randomized, the less likely it is to reform the components as well. A similar action is applied to ComponentB. Now we may take the two components and apply the commutativeOperation method. This method takes an operation character as an input. In this case, it is the division character ‘/’. Once it is called, the two components are joined together with the operation in mind. In this particular case of division, there are four different ways used to signify division. All of which take advantage of the commutative nature of division. Thus we may simply divide ComponentA by ComponentB, multiply ComponentA by the reciprocal of ComponentB using brackets or the multiplication symbol ‘\*’. The final way to represent division is to make a recursive call to commutativeOperation again but using the multiplication character ‘\*’ as input. But first applyPower is used on ComponentB to raise it to a power equivalent to negative one. This serves to further vary the representation of division. Finally applyTrig is used to encase the returned string inside a trigonometric function, which is arctan in this case.

Wrong answers are generated in a similar fashion to the process shown above, however the modular methods used may now vary the components into forms no longer equivalent to the given ones. For example, applyPower may now take the intended power as a negative, a reciprocal, a negative reciprocal, a decimal, or a scaled fraction numerator/denominator/both by 2 or 10. This allows the fraction to potentially appear more correct.  
Functions such as commutativeOperation and applyTrig will choose a random operation or trigonometric function instead. At least one component is always guaranteed to be reformed to an inequivalent variant.

## Increasing Difficulty

Time may be reduced, the number of shells required to pass a level increased, the number of options presented as well as the similarity and complexity of the questions increased in order to scale up the difficulty of the game based on user level.

As the player progresses through the levels they have acquainted themselves to the answer forms, so the game focuses more of testing the user on their ability to spot small mistakes.  
To closely match an answer when it involves a trigonometric function, we present more and more of the same trigonometric functions depending on the user level and randomly vary all other components. Note that the number of wrong components changes the difficulty. Less wrong components means it is harder for the user to figure out the right or wrong answer. Thus, we track and randomize the number of wrong components to a degree based on the player level.

# Techniques Used

## Databases

Storing information after the application, “Calculot”, closes was deemed an important part of the application when deciding functional requirements. The user must be able to store information about their progress after the application has been closed. More so, in the case that one would choose to share the application between multiple users on the same device, the application must be able to support multiple users and can keep track of each individual user’s progress separately.

SQLite was chosen to be used as the database manager for “Calculot” for various reasons. The development team for the application was already familiar with the implementation of SQLite in an Android application from past experiences. Moreover, the manager is natively supported by Android [6], and features a relatively simple interface that allows its implementation to be done without problems.

After beginning the application and not already logged in, the user is asked to sign up within the app. This sign-up activity extracts the inputted user information and stores it in the SQLite database, creating the user’s account. Information asked for by the sign-up activity includes: chosen username, user’s first name, and the user’s chosen password. The password field must be entered twice to ensure that it is correct.

The user also can add multiple users, one at a time. The same fields are required to be filled, and the user information is stored in the next row of the database.

Throughout the application the user gains experience points (“XP”), based on their progress through the learning activities and game activities. These are also stored in the database respectively as “Learning XP”, and “Practice XP”.

The information described makes up the user’s profile within the app and is called upon by the application to show their progress through the application.

## SharedPreferences

During the first sprint of production, the development team decided the application must keep the user logged in until they decide to log out. The application must also keep the user logged in when the application is closed.

The best approach to this problem was decided to be using SharedPreferences. SharedPreferences is a common method used to store data in an Android application even after the application ceases. SharedPreferences has been include in Android since API level 1. It allows the developers to store a piece of data with a string (“key”) that is used to retrieve it [7].

After the user selects their username from the list of registered users, and then enters their password correctly, they are then considered logged in. The application implements this by storing that user’s username in SharedPreferences under the key “username”.

Upon starting the application, SharedPreferences is checked to determine if a use is already logged in. If the string pulled from the SharedPreferences is null, then the application’s main menu will ask the user if they would like to sign up or sign in. If the string pulled from the SharedPreferences is not null, then the application’s main menu will instead show a different set of options.

SharedPreferences are used throughout the app to determine information about the currently logged in user. If the development team required any user information besides the username, they are required to call from the user database for the rest of the information. The implemented database returns a custom “User” class that holds all the user’s information. From here the developer can use any information about the user wherever that User instance carries on.

Because the user is kept logged in to their account even when closed, a way is needed to allow users to log out of their account and switch users if needed. When the user is logged in, the main menu displays a button to log out. Upon pressing the button the “username” field in SharedPreferences is set to null and the menu is set to the menu that appears when not logged in, thus logging the user out.

# Constraints and Limitations

## Preface

Due to the simplicity of the application and the unfamiliarity of android and java development of the team, “Calculot” as an example of a standardized application proved to be constrained and limited to simple design and implementation techniques. The issue can be generalized as an issue of understandability vs. performance gains of the games. In short, the games had to be as understandable as possible while also providing a playable experience to the user. As such, the following points of the development of “Calculot” was constrained:

1. The limited range generated by the question generation function
2. The notation provided in “Calculot” was done using a simpler, but more effective alternative,
3. Varying systems can differ in entertainment experience and difficulty.
4. Simplicity in design prevents complexity in development.

## The random question generation

Since “Calculot” is based on low-level university courses, the topics must be simple enough to provide entertainment to the user along with providing an effective learning schema to help the user understand the topics involved. Because of this, the question generation for the games implemented in “Calculot” had to be limited in design to prevent unreasonable questions and to create a fair experience for the user. The generation of the questions limited in the format and the range of constants to provide a solution to this issue.

Table I

Formatting of differentiation question set of “Calculot” in easy difficulty

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question | Answer 1 | Answer 2 | Answer 3 | Answer 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

As shown in Table 1, the formatting of “Calculot” had to be limited to simpler questions based on the topics. However, the answer set provided extra difficulty through making the questions somewhat similar to the solution string. This formatting is designed to punish the user from being unaware and unfocused on the question while also providing a fast-paced and interactive experience.

Similarly, the range of the constants and exponents in “Calculot” were limited reasonable integer values. This is to allow a reasonable pace on integrating or differentiating the functions. Allowing for unreasonable multiplication can result in a detrimental learning experience for the user as the objective of the game from a motivational standpoint is to allow the users a platform to display and/or practice their skills in advanced mathematics. Struggling with mental arithmetic should not be the focus of “Calculot” and should be treated as a trivial step to the solution.

## Notation

The notation in “Calculot” is an issue of android compatibility. Mathematical notation techniques have not yet been implemented and is currently a brute-force approach for mathematical applications. A workaround this issue would be to incorporate the android WebView [1] and HTML math notation using MathJax or KateX to the application. However, the drawback is a significant decrease in responsiveness and performance in the “Calculot” games. As such, the implementation of mathematical notation is to use a classical notation, while using constants. The implementation of “Calculot” used common ASCII symbols, such as “/” and “^” to incorporate the mathematical functions, such as

= x^n

= (k/l)

The decision to use a classical notation is motivated by prioritizing performance over understandability. While the understandability of the mathematical functions is severely affected by the notation, the performance drawback caused by WebViews caused both games in “Calculot” to be near-unplayable.

## Difference in platforms

The smartphone advancements in the past decade have advanced substantially. With new graphics and computer hardware becoming faster and faster, the experience of different applications can change dramatically, when comparing two extremes. The same reasoning can also apply to “Calculot.” The way that the integrated android Canvas class [4] operates is that the class calls the method onDraw() as much as it can, which allows the class to draw each object to the screen. The rate at which onDraw() is called can vary based on the device specifications; some devices will call onDraw() much faster than other devices. This can result in the game providing slight differences in the form of the enemy’s velocity as it progresses toward the tower. A possible solution to this issue is to limit the calls of onDraw() to a minimum. Equivalently, this will lock the frame rate of the graphics for “Calculot” However, this reduces efficiency further and older devices will be impacted more heavily than the benefit for newer devices. The development design focuses on simpler game design and the development team decided to broaden the range of compatible devices that “Calculot” can support.

Similarly, the screen resolution of different devices proved to be an issue for “Calculot.” More specifically, the sizes of each game object differed in size for different screen resolutions and the velocity of the monster was also impacted by screen resolution for “the Defense of Calculot.” Both of these issues were solved through scaling bitmaps and using mathematics to place and move each object based on the screen resolution. An example of a mathematical functions that were used were using this format:

Where the dimension of the object was the height and width, separately. The reasonable % of screen was achieved through trial and error, where the graphics were drawn on one mobile device and tested for a plethora of other devices.

In conclusion, the experience of “Calculot” can vary depending on each device, as varying devices have different screen resolution, CPU speed and other hardware. The development team behind “Calculot” attempted to minimize the impact of the differences in media through using workaround methods in the application and as a result, provided somewhat comparable results across a range of devices.

## Simplicity in Design and how it impacts current and future development

The way that “Calculot” was implemented and designed is an straightforward design compared to other applications of the same domain. This can prove to be an issue as some users can find “Calculot” to be too simple in nature, and not have an enjoyable experience as other might have. However, this opinion can also vary as some prefer the simple entertainment method to also help them learn mathematics.

There are also other limitations that is a result of this straightforward design. As mentioned previously, the issue with compatibility of a range of devices can be attributed to the design of “Calculot.” The implementation of the workarounds proves to be a difficult issue due to this. More specifically, the frame rate lock for “the Defense of Calculot” becomes a trivial issue in terms of implementation as the experience gains for this came is minimal compared to the impact it will have on older devices.

Another example is the decision to use the SQLite Database [5] While SQLite provides a very effective method of storing organized data, this only stores the data locally, and can provide issues later: the size of the database might grow to affect the user’s storage as it grows to a large size, or the issue that the user has no way to back up their user data. Due to the decision to use SQLite for the simplicity and effectiveness of a local database, the modern standard for entertainment-based applications is a cloud-based online webserver, which the user can back-up and maintain through the web.

Lastly, the incorporated Canvas class that “Calculot” uses operates in a simple manner. When the onDraw() is called, it simply displays all of the objects at given coordinates and sizes. A much more effective way of providing graphics for “Calculot” is to use an integrated game engine, such as unity. However, due to the unfamiliarity with these alternative methods, “Calculot” had to be programmed using the OpenGL Canvas class for its simplicity.

Though there are many constraints to the simplicity in design, it also provides many benefits. It helps to gauge all of the advantages and disadvantages to decide which method will deliver the best experience to a broad range of users.

# < insert segment here >

# < insert segment here >

References

[1] <https://developer.android.com/reference/android/webkit/WebView.html>

[2] <https://www.mathjax.org/>

[3] <https://github.com/Khan/KaTeX>

[4] <https://developer.android.com/reference/android/graphics/Canvas.html>

[5] <https://www.sqlite.org/>

[6] <https://developer.android.com/training/basics/data-storage/databases.html>

[7] https://developer.android.com/reference/android/content/SharedPreferences.html

1. [↑](#footnote-ref-1)