GSoC 2020 Proposal

Introduction

The GCompris Project is an educational software suite, aimed at children between the ages 2 to 10. Currently, the level of difficulty of activities are restricted by their dataset. This reduces the options of activities that the child of a given age can play.
In order to better cater to the different ages, the idea of adding the multiple datasets for different difficulty levels has been proposed.

Currently, a multiple dataset branch exists with the datasets for 25 activities. This has to be further extended to more activities.

The task is to add multiple datasets for different difficulty levels, for children aged 2 - 10. This will help better selection of competences that they want to work upon, by the children, guardians or teachers.The project will incorporate the datasets into the suite, with a global setting enabling to choose the level of difficulty the user wishes to pursue.

For the work of implementing multiple datasets, a phabricator task had been created. This mentions the different tasks as well as the dataset that needs to be implemented for some of the tasks: https://phabricator.kde.org/T12428

Project goals

As per the details of the project, I haved planned to submit multiple datasets and incorporate them in different activities in the GCompris project.
Following is the list of different activities and their datasets that I plan to work upon during the GSoC period:

- Sudoku
- Enumeration memory activity
- Addition memory activity against Tux
- Subtraction memory activity against Tux
- Addition and Subtraction memory activity against Tux
- Multiplication memory activity against Tux
- Division memory activity against Tux
- Multiplication and Division memory activity
- All operations memory activity against Tux
- Mirror the given Image
- Share pieces of candy

- Categorization
- Build the same model
- Hangman activity
- Gnumch Equality
- Gnumch Inequality

I will also plan to rework and improve on some activities:
- Sharing Candy
- Categorization
- Gnumch Equality
- Gnumch Inequality

For future developments, I would like to take part in discussions and contribute to activities for which it is yet to be decided if having multiple datasets is a viable option or not.

Implementation

For the work of implementing multiple datasets, a phabricator task had been created. This mentions the different tasks as well as the dataset that needs to be implemented for some of the tasks: https://phabricator.kde.org/T12428

- Sudoku
    - In the activity, there are symbols in lower levels and numerals in bigger levels. I can extend symbols to the higher levels and numerals to lower levels, so that levels involving symbols are present in higher levels of difficulty and those involving numbers are present in easier levels of difficulty.
    - This will help target a smaller age group that doesn't know numbers yet as well the older half of the age group that will prefer practice with numbers for such an exercise.
    - The current activity contains a total of 13 levels.
    - A proposed idea of such a multidataset would be:
      MultipleData1: 2 stars
      Level 1: 3X3 grid, with shapes (simple shapes like square, circle, triangle)
      Level 2: 3X3 grid, with shapes (shapes like star, hexagon)
      Level 3: 3X3 grid, with colored numbers (1,2,3)
      MultipleData2: 3 stars
      Level 1: 4X4 grid, with shapes (simple shapes like square, circle, triangle)
      Level 2: 4X4 grid, with shapes (shapes like star, hexagon)
      Level 3: 4X4 grid, with colored numbers (1,2,3,4)
      MultipleData3: 4 stars
      Level 1: 5X5 grid, with shapes (similar repeated shapes like star, hexagon ,plus etc.)

Level 2: 5X5 grid, with characters such as alphabets
Level 3: 5X5 grid, with colored numbers (1-5)
MultipleData4: 5 stars
Level 1: 9X9 grid, with shapes (triangle, square, circle, hexagon, plus etc.)
Level 2: 9X9 grid, with characters such as alphabets
Level 3: 9X9 grid, with colored numbers (1-9)
Level 4: 9X9 grid, with colored numbers (1-9)

- The levels are already defined in the sudoku.js file within the sudoku activity file in the form of arrays. This can be added to the Data.qml file within the Dataset object as levels property
- Different difficulty levels of the activity are put into different Data.qml files, below is an example of what a dataset for the MultipleData1 would look like:

```
Dataset {
        objective: qsTr("Arrange in a gridsize up to 3X3")
        // 2 star difficulty
        difficulty: 2
        data: [
                {
        "objective" : qsTr("Arrange the simple shapes: square, circle, triangle"),
                "levels" : [
                //  8 puzzles, characters A, B, C to be replaced with simple shapes
                        [
                        ['.','C','B'],
                        ['.','B','A'],
                        ['.','A','C']
                        ],
                        .
                        .
                        .
                        [
                        ['A','B','C'],
                        ['.','C','A'],
                        ['.','A','.']
                        ],
                ]
                },
                {
        "objective" : qsTr("Arrange the more complex shapes: star, hexagon"),
                "levels" : [
        //  5 puzzles, characters D,E,F denoting complex shapes like star, hexagon
                        [
                        ['D','.','.'],
                        ['F','.','.'],
                        ['E','F','.'],
                        ],
                        .
                        .
                        .
                        [
                        ['.','D','E'],
                        ['E','.','.'],
                        ['D','.','F'],
                        ],
                ]
                },
        "objective" : qsTr("Arrange the numbers 1-3"),
                "sublevels" : 5,
                "levels" : [
                        [
                        ['1','.','.'],
                    ['3','.','.'],
                    ['2','3','.'],
                    ],
                    .
                    .
                    .
                    [
                    ['.','1','2'],
                    ['2','.','.'],
                    ['1','.','3'],
                    ],
                ]
                },
            ]
        }
```
- For images/symbols needed at the higher levels, I will reuse the images such as the GCompris logo, Tux, warning logo etc. I am proficient in Gimp and can make more images if required.

● Enumeration Memory Activity
  - In the activity, ability to count is a prerequisite. The range of these numbers can be changed in the datasets for varying levels of difficulty. The enumeration memory activity dataset implementation will be as is described in the task link.

● Addition memory activity against Tux
  - In this activity, cards with the sum value and the addition operation need to be matched. There are currently 10 levels, with the activity of the level being 3 stars.
  - The difficulty can be changed by the numbers that are being added and whether the sum involves a carry operation or not.
  - For the dataset, I should create levels , the easier ones with one digit sum and the ones at higher difficulty with 2,3,4-digit sum and carry operations (such as 9 + 2)
  - The Multidataset that I propose for this activity is as follows:
    MultipleData1: 3 stars
    Level 1: Addition of 1-digit numbers with one-digit sum
    Level 2: Addition of 1-digit and 2-digit numbers with one and two-digit sum
    Level 3: Addition of 2-digit numbers with two-digit sum and carry
    MultipleData2: 4 stars
    Level 1: Addition of 2-digit numbers with two-digit sum with carry
    Level 2: Addition of 3-digit numbers with three-digit sum
    Level 3: Addition of 3-digit numbers with three-digit sum and carry
    MultipleData3: 5 stars
    Level 1: Addition of 3-digit numbers with three-digit sum with carry
    Level 2: Addition of 4-digit numbers with four-digit sum
    Level 3: Addition of 4-digit numbers with four-digit sum and carry

  - When the number of digits increases, it may become difficult to fit the arithmetic expression properly in one line, such as in addition to 4 digit numbers. In such cases, I could change the format to write the text in 3 lines: 1st operand number, operation character, 2nd operand number.

- The same dataset can also be reused in Addition memory activity without Tux

- Subtraction memory activity against Tux
    - In this activity, cards with the difference value and the subtraction operation need to be matched. There are currently 10 levels, so I plan to add the higher difficulty datasets on top of it.
    - The difficulty can be changed by the numbers that are being added and whether the sum involves a borrow operation or not.
    - For the dataset, I can create levels , the easier ones with one digit difference and the ones at higher difficulty with 2,3,4-digit difference and borrow operations.
    - The Multidataset that I propose for this activity is as follows:
    MultipleData1: 3 stars
    Level 1: Subtraction of number table zero (1-0, . . . 9-0)
    Level 1: Subtraction with number table one (1-0, . . . ,5-1)
    Level 2: Subtraction with the number table one and two (2-0, 2-1 . . . 5-2)
    Level 3: Subtraction with the number table two and three (3-0. . .5-3)
    MultipleData2: 4 stars
    Level 1: Subtraction with number table three and four (4-0, ... 10-4)
    Level 2: Subtraction with number table four and five (5-0. . .10-5)
    Level 3: Subtraction with number table five and six (6-0. . .10-6)
    Level 4: Subtraction with number table six and seven (7-0. . .10-7)
    Level 5: Subtraction with number table seven and eight (8-0. . .10-8)
    Level 6: Subtraction with number table eight and nine (9-0. . .10-1)

    - The same dataset can also be reused in Subtraction memory activity without Tux

    - When the number of digits increases, it may become difficult to fit the arithmetic expression properly in one line, such as in addition to 4 digit numbers. In such cases, I can change the format to write the text in 3 lines: 1st operand number, operation character, 2nd operand number.

- Addition and Subtraction memory activity against Tux
    - In this activity, cards with the value and the addition/subtraction operation need to be matched.
    - I plan to reuse the datasets that I implement in Addition and Subtraction activities. In this case, the Addition and Subtraction datasets with the same level and difficulty will be present together.

- Multiplication memory activity against Tux
  - In this activity, cards with the product value and the multiplication operation need to be matched. The difficulty can be changed by the numbers that are being multiplied, with small numbers for easy levels and bigger numbers for difficult levels.
  - The Multidataset that I propose for this activity is as follows:
    MultipleData2: 4 stars
    Level 1: Multiplication of number table zero (0x1, . . . 0x10)
    Level 1: Multiplication with number table one (1x1 . . . ,1x10)
    Level 2: Multiplication with the number table one and two (2x1 . . . ,2x10)
    Level 3: Multiplication with the number table two and three (3x1 . . . ,3x10)
    Level 1: Multiplication with number table three and four (4x1 . . . ,4x10)
    Level 2: Multiplication with number table four and five (5x1 . . . ,5x10)
    Level 3: Multiplication with number table five and six (6x1 . . . ,6x10)
    Level 4: Multiplication with number table six and seven (7x1 . . . ,7x10)
    Level 5: Multiplication with number table seven and eight (8x1 . . . ,8x10)
    Level 6: Multiplication with number table eight and nine (9x1 . . . ,9x10)

  - The same dataset can also be reused in Multiplication memory activity without Tux

  - When the number of digits increases, it may become difficult to fit the arithmetic expression properly in one line, such as in addition to 4 digit numbers. In such cases, I can change the format to write the text in 3 lines: 1st operand number, operation character, 2nd operand number.

- Division memory activity against Tux
  - In this activity, cards with the quotient value and the division operation need to be matched. The difficulty can be changed by the numbers that are being divided, with small numbers for easy levels and bigger numbers for difficult levels.
  - The Multidataset that I propose for this activity is as follows:

    MultipleData2: 4 stars
    Level 1: Division of number table zero (0/1, . . . 0/10)
    Level 1: Division with number table one (1/1 . . . ,10/1)
    Level 2: Division with the number table one and two (2/1 . . . ,10/2)
    Level 3: Division with the number table two and three (3/1 . . . ,10/3)
    Level 4: Division with number table three and four (4/1 . . . ,10/4)
    Level 5: Division with number table four and five (5/1 . . . ,10/5)
    Level 6: Division with number table five and six (6/1 . . . ,10/6)
    Level 7: Multiplication with number table six and seven (7/1 . . . ,10/7)
    Level 8: Multiplication with number table seven and eight (8/1 . . . ,10/8)

Level 9: Multiplication with number table eight and nine (9/1 . . . ,10/9)

- The same dataset can also be reused in Division memory activity without Tux

- When the number of digits increases, it may become difficult to fit the arithmetic expression properly in one line, such as in addition to 4 digit numbers. In such cases, I can change the format to write the text in 3 lines: 1st operand number, operation character, 2nd operand number.

- Multiplication and Division memory activity
    - In this activity, cards with the value and the multiplication/division operation need to be matched.
    - I plan to reuse the datasets that I implement in Multiplication and Division activities.

- All operations memory activity against Tux
    - In this activity, cards with the value and the multiplication/division operation need to be matched.
    - The datasets used for addition, subtraction, multiplication and division activities can be reused here.For higher levels of difficulty, I can incorporate more than one operation at a time in the card. This can be done by adding the operators and operands in different lines.

- Mirror the given Image
    - In this activity, the left side needs to be coloured to as a mirror image to the right side. I plan to implement the dataset described in the task itself.

- Share pieces of candy
    - In this activity, candies need to be divided equally among friends and keep the rest in a jar. For the lower levels, I can keep the division such that there are no rests and include rest in higher levels.

    - In the activity, the levels are present in the board1.qml to board7.qml files, where each board is for level 1 to 7.

    - I can directly use the levels property in QtObject in the Dataset object present in the Data.qml files.

    - In this case, the Dataset present in the Data.qml file will be in the following format:

```
Dataset {
        objective: qsTr("Share candy between friends")
        difficulty: 4
        data: [
        {
        "objective" : qsTr("Select the numbers 1 and 2"),
        "levels" : [
        { // 3 sublevels
                "instruction": qsTr("George wants to equally share 2 pieces of
candy between 2 of his friends . . . drag the pieces of candy to each of them."),
                "totalBoys": 1,
                "totalGirls": 1,
                "totalCandies": 2,
                "showCount": true,
                "forceShowBasket": false,
                "placedInGirls": 0,
                "placedInBoys": 0
        },
        ....
         {
        "instruction": qsTr("John wants to equally share 9 pieces of candy between
3 of his friends . . . then drag the pieces of candy to each of them."),
                "totalBoys": 1,
                "totalGirls": 2,
                "totalCandies": 9,
                "showCount": true,
                "forceShowBasket": false,
                "placedInGirls": 0,
                "placedInBoys": 0
                }
        ]
        }
    ]
}
```

- For the lower level, we need an implementation without any rest, so I will set forceShowBasket to false, the value of totalBoys and totalGirls can be interchanged to create more sublevels if required. The value of totalCandies can also be increased and forceShowBasket can be set to false for higher levels.

- Categorization
    - In this activity, different pictures and objects need to be categorized according to the given instructions by placing them on different parts of the screen. I plan to implement the dataset described in the task itself.
    - Also, I think we can add a hint system, or make the hint persistent as the user cannot refer to it again to confirm the aim of that level.

- Find the details
    - In this activity, regions matching the picture need to be placed in the correct spots. I plan to implement the dataset described in the task itself.

- Build the same Model

    - In this activity, pieces need to be arranged as made on the right side of screen by selecting and navigating the pieces to their correct position. I plan to implement the dataset described in the task itself.

- Hangman activity
    - In this activity, we need to guess the word and there's a limit to the wrong letters that we can guess.
    - For the difficulty level, I think we need to decide on more commonly occuring words rather than longer and short words. But as this will not remain fixed when translated to other languages, we will first set the words, that are commonly smaller in length when translated, for difficult levels and more common words that are generally long for easy levels

- Gnumch Equality
    - In this activity, the number muncher needs to be set on blocks with arithmetic expressions that give a value mentioned near the bottom of the screen.
    - For this activity, the use of different operators for varying difficulty levels has been decided on IRC. So, addition and subtraction operators in the easy levels and multiplication and division in the difficult levels will be incorporated.
    - For this activity, it was decided to separate the levels based on operations into 4 different levels. The activity is aimed at an age group that wants to practice the different operations.
    - I feel that we can also make it suitable for higher age groups by adding the concept of negative numbers in these operations.

      The Multidataset that I propose for this activity is as follows:
      MultipleData1: 2 stars
      Level 1: Practice Addition Equality with one-digit sum
      Level 2: Practice Addition Equality with two-digit sum
      Level 3: Practice Subtraction Equality with one-digit difference
      Level 4: Practice Subtraction Equality with two-digit difference
      MultipleData2: 3 stars
      Level 1: Practice Addition Equality with one-digit numbers with carry
      Level 2: Practice Addition Equality with two-digit numbers with carry
      Level 3: Practice Subtraction Equality with one-digit numbers with borrow
      Level 4: Practice Subtraction Equality with two-digit numbers with borrow
      MultipleData3: 4 stars
      Level 1: Practice Multiplication Equality with one-digit numbers
      Level 2: Practice Multiplication Equality with two-digit numbers
      Level 3: Practice Division Equality with one-digit numbers
      Level 4: Practice Division Equality with two-digit numbers
      MultipleData4: 5 stars
      Level 1: Practice Multiplication Equality with one-digit numbers with carry

Level 2: Practice Multiplication Equality with two-digit numbers with carry
Level 3: Practice Division Equality with one-digit numbers with borrow
Level 4: Practice Division Equality with two-digit numbers with borrow
MultipleData5: 6 stars
Level 1: Practice Addition Equality with one-digit numbers with negative numbers
Level 2: Practice Subtraction Equality with one-digit numbers with negative numbers
Level 3: Practice Multiplication Equality with one-digit numbers with negative numbers
Level 4: Practice Division Equality with one-digit numbers with negative numbers

- Also, I think that activity requires rework in the following areas: adding the option to switch between levels as we have to work on the activity from the beginning.
- Resetting the troggle position that follows the gnumch , after we get a second chance as sometimes having it near the reset position of gnumch leads to instant destruction of gnumch. For adding the levels, I will reuse the code from other applications that involve levels (almost all activities involve levels that can be switched)

- Gnumch Inequality
    - In this activity, the number muncher needs to be set on blocks with arithmetic expressions that do not give a value mentioned near the bottom of the screen.
    - Similar to Gnumch Inequality, the different operators in the expression will help create varying levels of difficulty within the activity.
    - I plan to use the same MultiDataset plan that I propose for the Gnumch Equality activity.
    - This activity will also require rework similar to the Gnumch Equality activity.

To incorporate the different datasets I will do the following:

- Add levels property in the ActivityInfo.qml

- After that, we will add another field to the EnumBarContent for the option of changing the dataset. For maintaining conventions in naming, it will be named ActivityConfig. This will point the code in ActivityConfig.qml file

- Create the ActivityConfig.qml file. Within that file, we'll create the Item object containing the required aliases (such as speed slider and speed settings, if the activity involves object movement from the activity side).

- The Item will further contain the GCComboBox, GCCheckbox, GCSlider objects along with GCText that will tell about the settings and allow the user to change the required settings.

- Furthermore, JS functions for setDefaultValues() and saveValues() will be added within the ActivityConfig.qml file. Activities with multidatasets implemented will be used as reference for writing both of these functions. These functions will be used to set the default values for the checkbox and slider mentioned as well as to save locally any changes made by the user.

- I will then add the ActivityConfig onClicked functionality, which forms the display dialog by calling the displayDialog() function.

- When the user will click on the ActivityConfig button, it will display a dialog whose code will be present in the ActivityConfig.qml.

- Within the .js file, I will create a variable currentLevel, which will start from 0 and iterate through the levels array that I defined within the ActivityConfig to fetch the level.

- Using levels.length property, I will assign the value I get to variable numberOfLevels

- Then, the JS code will be modified to extract the number of levels, sublevels and value-ranges from the dataset.

- To implement the addition of the datasets, I will first define the dataset in the Data.qml file for each activity.

- This dataset will be present in the dataset.qml file within Dataset (object) defined within the data property in JSON format, containing the level number, total sublevels, value ranges and the objective for that level. Any instructions or objectives mentioned within the dataset will be enclosed within qsTr() to help translate these instructions based on localization.

- I will write each of the datasets separately for each level in a different Data.qml file.

For an activity such as sudoku, a Dataset I could add the different fields objective, difficulty, and data array which will contain details specific to each level that has the mentioned level of difficulty. Such an example could be coded as follows:

```
Dataset {
       objective: qsTr("Select a gridsize up to 3X3")
       difficulty: 1
       data: [
        {
         "objective" : qsTr("Select the simple shapes: square, circle, triangle"),
         "levels" : [
              //  8 puzzles, characters A, B, C to be replaced with simple shapes
              [
              ['.','C','B'],
              ['.','B','A'],
              ['.','A','C']
```

```
                ],
                .
                .
                .
                [
                ['A','B','C'],
                ['.','C','A'],
                ['.','A','.']
                ],
            ]
        },
        {
         "objective" : qsTr("Select the more complex shapes: star, hexagon"),
         "levels" : [
                //  5 puzzles, characters D,E,F denoting complex shapes like star, hexagon
                [
                ['D','.','.'],
                ['F','.','.'],
                ['E','F','.'],
                ],
                .
                .
                .
                [
                ['.','D','E'],
                ['E','.','.'],
                ['D','.','F'],
                ],
              ]
            },
       ... ]
}
```

As the majority part of the code is written in QML, for which a standard testing framework has not been decided yet by the team, there will not be any unit tests on the QML code.

For documentation, there is no specific code documentation requirement from the team. However, proper comments will be written while writing code , especially above each of the user instructions to specify its purpose to the translators. In terms of user documentation, proper instructions regarding the selection of multiple datasets will be added to aid the user.

Timeline

Community Bonding Period (May 4, 2020 - June 1, 2020)

- Bond with the organization members
- Discuss and finalize dataset preparation methods and workflow for all the activities.
- Finalize any images to be added in the activity.
- Decide upon communication methods and times.
- Add the blog to KDE Planet

Phase 1 (June 1-29, 2020)

- Week 1 (June 1 - June 7)
    - Write a blog about the community bonding period

  - Integrate the dataset to the Sudoku activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs , comments reviewed and merged
- Week 2 (June 8 - June 14)
  - Write a blog about the developments in previous week
  - Integrate datasets to the Share pieces of candy activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Add hint system to Share Candy activity
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs , comments reviewed and merged
- Week 3 (June 15 - June 21)
  - Write a blog about the developments in previous week
  - Integrate the datasets to the Addition activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs , comments reviewed and merged
- Week 4 (June 22 - June 28)
  - Write a blog about the developments in previous week
  - Integrate the dataset plan for Subtraction activity, Multiplication activity
  - Perform user testing of the updated activities
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs , comments reviewed and merged
  - Buffer period to complete any backlog of previous weeks.

Phase 1 Evaluation

Phase 2 (June 29 - July 27, 2020)

- Week 1 (June 29 - July 5)
  - Write a blog on the work done in phase 1 and send an email to the mailing list for the same
  - Integrate the dataset to the Division activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs , comments reviewed and merged
  - Fix any bugs that are encountered
- Week 2 (July 6 - July 12)
  - Write a blog about the developments in previous week

  - Integrate the dataset to the Addition and Subtraction, Multiplication and Division activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Get the diffs, comments reviewed and merged
  - Fix any bugs that are encountered
- Week 3 (July 13 - July 19)
  - Write a blog about the developments in previous week
  - Integrate the dataset to the Enumeration activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Integrate the datasets to the All operations memory activity against Tux activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs, comments reviewed and merged
- Week 4 (July 20 - July 26)
  - Write a blog about the developments in previous week
  - Integrate the datasets to the Categorization activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs , comments reviewed and merged
  - Buffer period to complete any backlog of previous weeks.

Phase 2 Evaluations

Phase 3 (July 27 - August 24, 2020)

- Week 1 (July 27 - August 4)
  - Write a blog on the work done in phase 2 and send an email to the mailing list for the same
  - Integrate the dataset to the Build the same model, Mirror Image activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Around 2 days to get the diffs, comments reviewed and merged
- Week 2 (August 3 - August 9)
  - Write a blog about the developments in previous week
  - Integrate the datasets to the Hangman memory activity along with comments and user instructions
  - Perform user testing of the updated activity
  - Fix any bugs that are encountered
  - Improve hint system of Categorization activity

    - Perform user testing of the updated activity
    - Fix any bugs that are encountered
    - Around 2 days to get the diffs , comments reviewed and merged
- Week 3 (August 10 - August 16)
    - Write a blog about the developments in previous week
    - Add the Level System to all the Gnumch activities
    - Perform user testing of the updated activities
    - Fix any bugs that are encountered
    - Get the diffs, comments reviewed and merged
- Week 4 (August 17 - August 23)
    - Write a blog about the developments in previous week
    - Implement dataset plan for Gnumch Equality, Gnumch Inequality activity
    - Integrate the datasets to the activities along with comments and user instructions
    - Perform user testing of the updated activity
    - Fix any bugs that are encountered
    - Get the diffs, comments reviewed and merged
    - Buffer period to complete any backlog of previous weeks

Final Evaluation

To dedicate fully for GSoC, I am not taking any other commitments. I am ready to work up to 40-45 hours per week to maintain a proper work-life balance.
I will not be taking any internships or other work for this time so that I can reserve this time to contribute fully to the project and GSOC work.
I may have my exams in a very small patch of the coding period for which I am prepared to work overtime during weekends to maintain my progress during that time.

I plan to coordinate with my mentors regularly and have previously done so. As most of the developers are near the EU time zones, I have already shifted my daily workflow so that I can easily communicate regularly with them.
I have a blog hosted on Github pages and can initiate a weekly blog post on the developments, sharing it with the rest of the KDE community. I am also planning to communicate any major changes/developments or change in ideas in the team mailing list to which I have already subscribed.

About me

Name: Abhay Kaushik
IRC Nick: Matorix
Telegram Username: Matorix
Email Address: abhay.gyanbharati@gmail.com
Website: https://abhaykaushik.github.io

I am a 3rd year undergraduate student pursuing B.Tech with a major in Computer Science from Bharati Vidyapeeth's College of Engineering, New Delhi.
I am comfortable writing code in C++, QML. I have prior experience in building 2D activities in Unity and have built an infinite runner(https://github.com/AbhayKaushik/FlappyBird-Sidescroll) and 3-match puzzler(https://github.com/AbhayKaushik/Match-3-Puzzle), which I believe can help me provide valuable contributions to the GCompris project.
Also, I am comfortable using git and Linux operating system and using KDE Neon specifically for development purposes. I have also contributed to the repository within Phabricator and am aware of the workflow. I also have a blog setup and can communicate regularly to the community about my work.

I am passionate about education and sharing knowledge. I am currently a part of the Developer Students Club initiative in our campus and have taken workshops on Game Development, Web-based AR, Voice Assistants and Google Cloud, helping around 600 juniors in a span of almost 2 years.
In terms of non-coding open source contributions, I have contributed in terms of string translations in Hindi language for Calamares(https://www.transifex.com/user/profile/Matorix) and SecureDrop(https://weblate.securedrop.org/user/AbhayKaushik).

Within KDE, I have contributed to the Kaidan website:
https://invent.kde.org/websites/kaidan-im/-/merge_requests/15
Within the GCompris project, I have participated in the newcomers contribution activity:
https://github.com/gcompris/GCompris-qt/pull/255
Contributed to improvement through these merge requests: https://phabricator.kde.org/D28237
[Improving Score Counter in GLetters]
Also suggested a minor change in GCompris MediaWiki page

I have been on the KDE telegram channels, KDE Identity account holder and community mailing lists since the Season of KDE 2020 and attended the KDE India Conference 2020. I am fond of this community and decided to contribute to this organization only. I have not applied to any other GSoC organization.