Finishing activities for GCompris in Qt-Quick

Rudra Nil Basu

***Email ID**: rudra.nil.basu.1996@gmail.com*

***Freenode IRC Nick**: rudra*

***Location**: Kolkata, West Bengal, India UTC+5.30*

---

1. Motivation

GCompris is a high-quality educational suite which aims at making learning easier for children aged 2 to 10. GCompris currently has 137 activities on various topics such as science, maths, games with which it has successfully created a great learning environment for children. However, there are few activities which were started previously but is still not complete. I strongly believe in what GCompris stands for and in this project, I aim at taking GCompris one step forward by finishing three started activities: *Pilot a Submarine*, *Family* and *Digital Electronics*

2. Project Goals

By the end of the Google Summer of Code's time period, I will be completing the following activities :

- **Pilot a Submarine**: It is a port to the Qt version of a strategic activity originally present in the Gtk+ version aimed to teach how a submarine works. It was started in this branch:

  https://cgit.kde.org/gcompris.git/log/?h=gsoc-submarine

- **Family**: It is an activity to help children understand how they are related to their relatives. It was started in the branch:

  https://cgit.kde.org/gcompris.git/log/?h=GSoC-family

- **Digital Electronics**: This activity is aimed at providing a real time simulation of an electric circuit. It was started in the branch:

  https://cgit.kde.org/gcompris.git/log/?h=gsoc-pulkit-digital-electricity

Additional Task:

If time permits me, I will be working on one more activity which has not been started yet, but discussions were ongoing for the implementation of the activity:

- **Object Arrangement**: This activity is aimed at teaching arranging objects in a given order based on factors like height and width.

  The activities for arranging numbers and alphabets in either ascending or descending order is already in review and this will be a generic one, adding all kinds of ordering activities to the project.

  Discussions:

  https://phabricator.kde.org/T1956

3. Implementation Details

3.1. Pilot a Submarine

The *"Pilot a Submarine"* is to learn how a submarine works, explaining the usage of elements such as engine, rudders and air tanks, in order to navigate a submarine to a required depth.

- Since this activity was already present in the Gtk+ version, I will be using the svg and the audio files from the resources used in the Gtk+ submarines activity. This will allow me to dive into the coding part directly.
- I will be using box2d for the simulation and handling physics and collisions
- There will be a tutorial at the start of the activity, which will give a brief description about the different elements (engine, rudders and air tanks) and it's functions.
- Firstly I will be implementing the submarine and the mechanics of it's elements, namely the engine, rudders and the air tanks. Once that is in place, I will start creating various levels and it's variations.

- The aim is to make the activity easy for children to understand. For achieving this, the levels should start from the very basics. The implementation is divided into two parts:
  - the initial levels, which are aimed at giving a basic understanding of the use of different elements required to control a submarine. For example, the first level will explain the use of engines of a submarine and will ask the user to move from one point to the other by using the engine only. Similarly, the next levels will focus on using the ballast tanks, rudders and other elements of a submarine
  - the latter levels will use the concepts from the initial levels in various forms to check whether the child has fully understood the mechanism of controlling the submarine or not
- The activity will contain pickups in the form of jewels, as it was present in the Gtk+ version
- Besides the regular pickups in the Gtk+ version, there will be additional threats in the form of rocks and caves, in order to maintain an increasing difficulty curve, while still keeping it doable for children within the prescribed age limit.
- In order to enhance the experience, the overall activity and the movement of the submarine and the animations will be smoother compared to the Gtk+ activity.

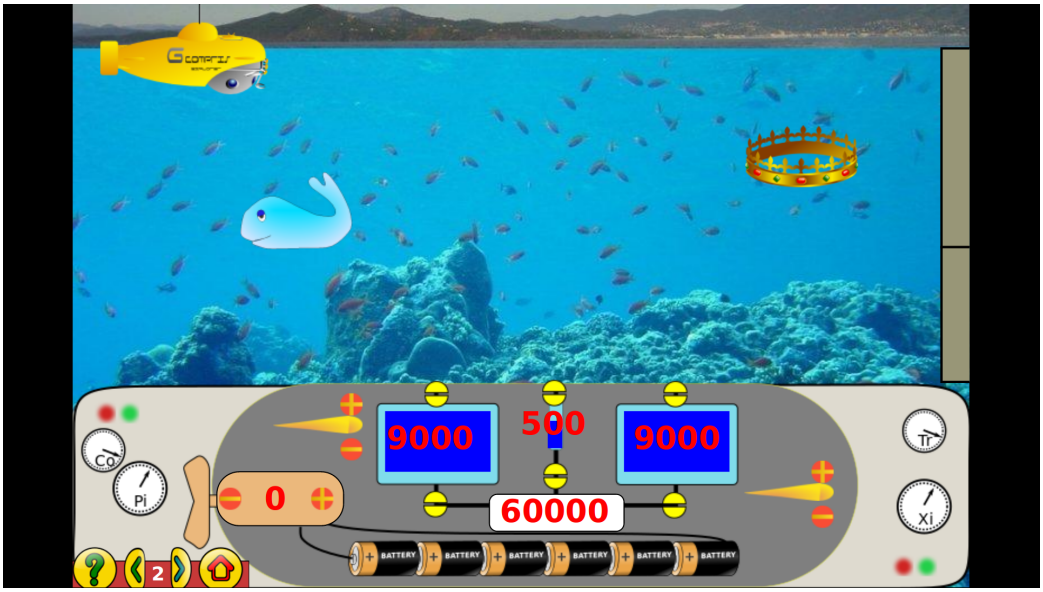

Figure 1: Submarine Activity

3.2. Family

The core implementation of the family activity is finished. The layout of the activity needs to be improved.

- The layout of the activity will be changed to a tree like representation
- It will have one extra sub-activity besides the existing activity
- In the sub-activity, a relation will be provided and the child will have to click on the correct pair which demonstrates the given relation.

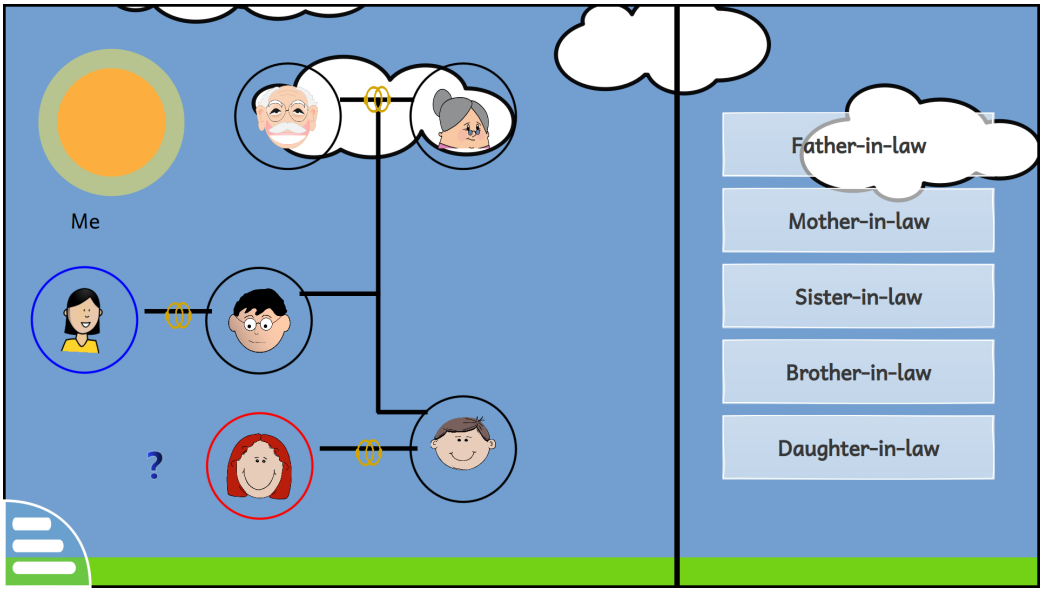

Figure 2: Current layout of the Family Activity

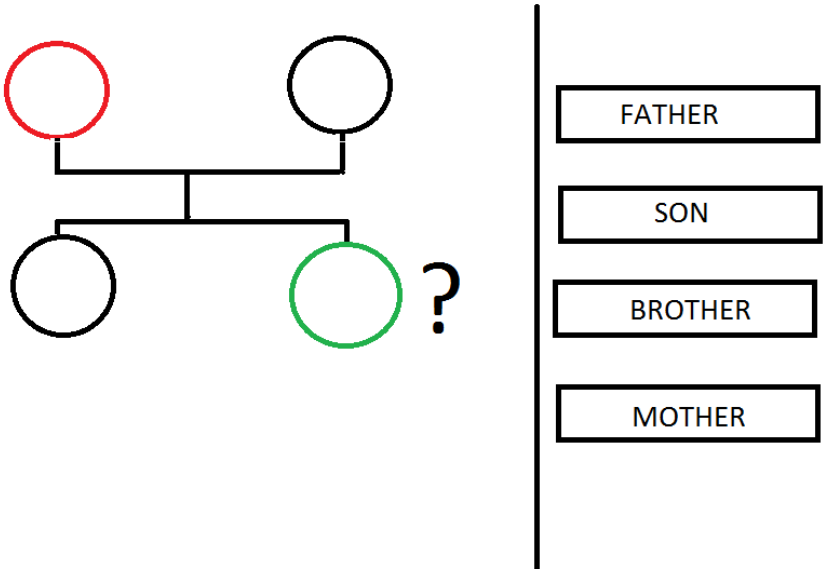

Figure 3: Sketch of the Final layout of the Family Activity

3.3. Digital Electricity

The core simulation of electricity is done and the circuits are working properly. But to make sure it is easier for the children to understand, we have to illustrate how each of the components work in a circuit. For this project, I propose:

- Whenever there is a level where a specific item is used for the first time, there will be an instruction text to inform the children on what the specific component does and how to operate it
- There will be levels to teach the children on how to use the components. These levels are broadly divided into three parts:
  - The first part will be a very basic one: the aim will just be to learn how to use the component. For example, if the component is the *Or Gate*, the child will be asked to light the bulb once using the "1" and "0" input, or with two "1" inputs.
  - The second part will take the previous concepts a bit further and will ask the child to solve a simple problem using the component, maybe reusing it more than once. For example, for the *Or Gate*, the child will be asked to light the bulb from three power sources, such that the bulb should be on if at least one of the power sources is turned on
  - The third part will make use of the component and one or more previously learned components and will ask the child to solve a given problem with a limited number of components
- There will be an open arena, as it is present in the current progress of the activity, where the child can try out any combination of circuits and test it
- Since the basic simulation is already done in the current progress of the activity, I can directly move on to the coding part, implementing the different levels for the activity

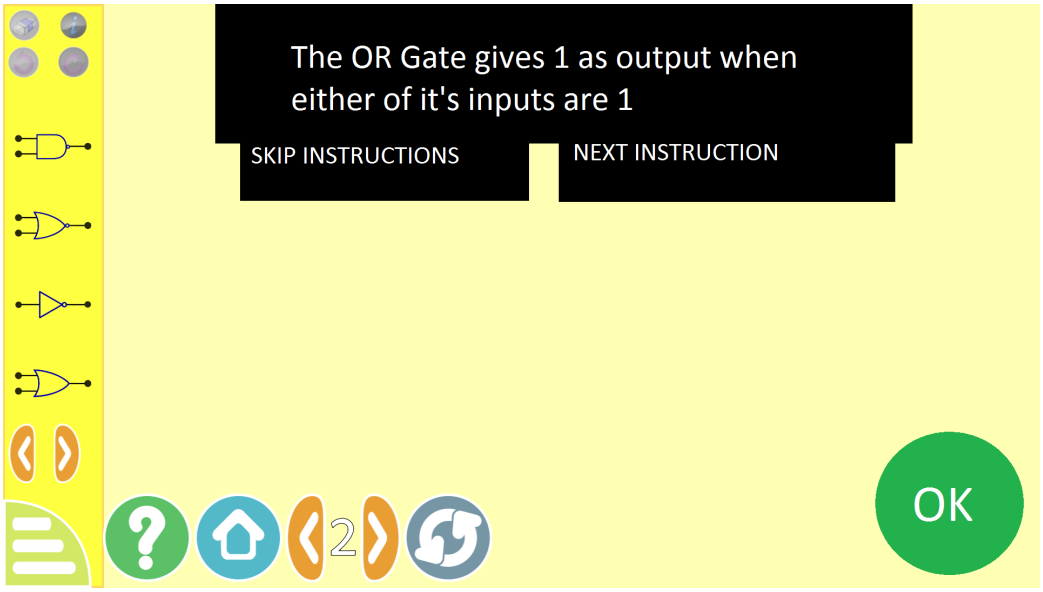

Figure 4: Digital Electricity: Instructions

3.4. ***Additional Task****: Object Arrangement*

This activity will be created from scratch, in order to expand the current ordering activity to a more generic version of it. It will have the following features:

- I will be using openclipart.org and Inkscape for the images
- Different types of objects will be provided at the start of each levels and the user will be asked to arrange them in a specific order (ascending or descending) based on a given parameter (height or weight)
- A toolbox will be provided to the user with the help of which the user can measure the required property of an item
- The measured properties will be marked on the object. If a property of an object is not yet measured, it will be marked as unknown.
- With the proper measurements, the user will then arrange the items in the correct order of the parameter

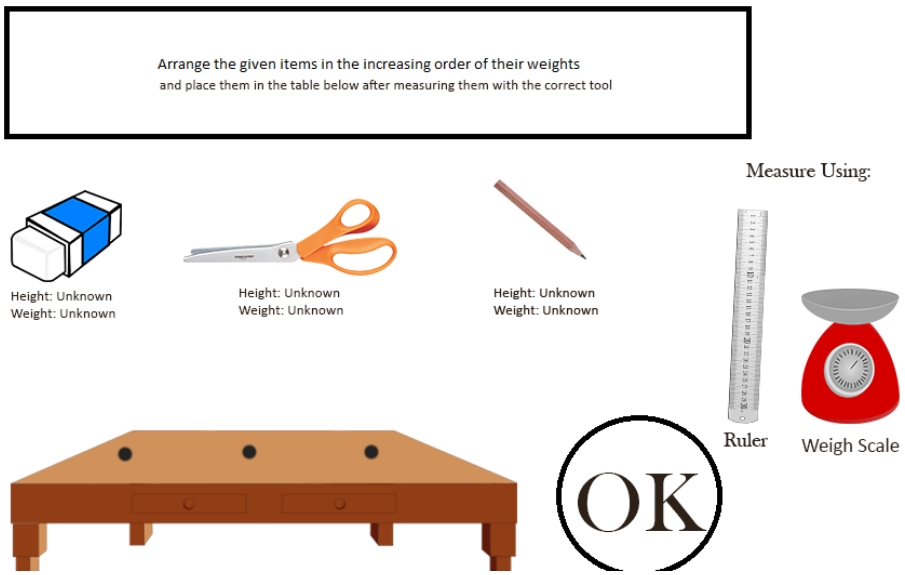

Figure 5: Object Arrangement

4. Timeline

May 5, 2017 - May 30, 2017: Community Bonding Period

- I will interact and consult with my mentors.
- I will take an in-depth look at the implementation of the activities in my proposal, discussing about the same with my mentors, so as to device an optimum method to implement the proposed tasks.
- I will look up the documentations and learn about box2d, the physics engine I will be using for the submarine activity
- I will look for the necessary resources (images and audio clips) required for my proposed projects.

May 31, 2017 - July 10, 2017: *Pilot a Submarine*

- **May 31 - June 5:** ***Basic Structure, Instructions***

  - Implement the basic structure of the activity, which includes the submarine itself, the surrounding environment, the gates, the colliders such as the rocks and ships. Also, include the basic components of the submarine: the engine, rudders and air tanks.
  - Create instructions for the activity: explaining the use of the components of the submarine (engine, rudders and the air tanks) and the goal of the activity.
  - Review the layout and the instructions reviewed by the mentors before proceeding and make necessary changes

- **June 6 - June 20:** ***Movement, Physics, Collisions and animations***

  - Implement basic movement of the submarine, physics and collision handling of the submarine using box2d
  - Detect pickups, game over and succesful level completion
  - Implement animations of submarine
  - Get the implementation reviewed by my mentors and make necessary changes

- **June 21 - June 30:** ***Implement levels with difficulty***

  - Implement different levels while maintaining a difficulty curve
  - Get the levels reviewed by my mentors and make necessary changes

- **July 1 - July 10:** ***Testing and bug fixing***

  - Test the activity on various platforms and screen sizes
  - Get reviews for the overall activity from my mentors
  - Make necessary changes based on the reviews from my mentors

July 11, 2017 - July 25, 2017: ***Family***

- **July 11 - July 18:** ***Improve layouts***

  - Change the layout of the activity to make sure that the implementation is clean and doesn't override other components of the activity

  - Get reviews from my mentor and make necessary changes

- **July 19 - July 25:** ***Implement sub-levels***
  - Implement sub-levels: given a relation, the user will have to click on a pair which satisfies the given relation
  - Review the new sub-levels from my mentors and make necessary changes

July 26, 2017 - August 21, 2017: ***Digital Electricity***

- **July 26 - July 31:** ***Initial levels, Instructions***
  - Create the initial levels which are aimed at teaching the basic workings of the components
  - Display the instructions at the beginning of the levels where a new component is used for the first time
  - Review the layout and the instructions from my mentor and make necessary changes

- **August 1 - August 15:** ***Implement levels with difficulty***
  - Provide a specific problem to the user and ask them to solve it using only specific number of components
  - Implement restricting the number of components that can be used per level to that specified in the level
  - Checking the resultant circuit arrangement after the *"ok"* button is clicked, and display the result to the user.
  - Get the working of the activity reviewed by my mentors and make necessary changes

- **August 16 - August 21:** ***Testing***
  - Testing the activity on various platforms and screen sizes
  - Get more reviews from my mentors on the overall activity
  - Make necessary changes based on the reviews

August 22, 2017 - August 29, 2017: *Final Week*

- Rigorous testing of the activities on various platforms
- Fixing the reported bugs
- Submit the code for final evaluation

Other commitments

- I will be having exams from June 1st to June 15th. During that timeline, I will be able to work 3 hours per day
- I might be out of station from July 10th to 16th. I will be able to work in my free time, giving around 6 hours per day
- During the rest of the period, I will be able to work 8-10 hours per day

5. About Me

I am a third year undergraduate engineering student from Maulana Abul Kalam Azad University of Technology (formerly known as West Bengal University of Technology), pursuing B.Tech in Computer Science and Engineering. I have been contributing to KDE for the past few months on the Qt version of GCompris, which helped me to get familiar with Qt and the overall GCompris codebase.

My contributions on GCompris include:

- Display the characters attempted by the user in the Hangman activity

  https://github.com/gcompris/GCompris-qt/commit/8ab75acf49431c685021f3cd0e58cf31f3fa4568

- Adding a Directory class in Core to directly get a list of all files in a given directory

  https://github.com/gcompris/GCompris-qt/commit/955462b943c34fc130d1a68fcfb0e1ec6393a3f0

- Improve the algorithm to add new levels in the categorization activity, so that we do not need to change the code while adding new levels. Also, added odd-even category in the categorization activity

  https://github.com/gcompris/GCompris-qt/commit/db7a4a9b743a3521c7c68f0b2b54719cbc9582db

- Display a black point in drawletters and drawnumbers activity whenever a line cannot be drawn for a given input

  https://github.com/gcompris/GCompris-qt/commit/eaf8dd326dd3f5581995155739be5c72af744f92

- **Under review**: Ordering activity, which aims at arranging numbers and alphabets in it's increasing or decreasing order

  https://github.com/gcompris/GCompris-qt/pull/172

I am also the head of our college's programming club, where we create awareness about open source and encourage newcomers to take part in open source contribution.

Some of my other open source projects include contributing to Algorithm Visualizer (github.com/parkjs814/AlgorithmVisualizer), Battery Manager (github.com/BytesClub/Battery-Manager) and few video game projects which I build during my free time, namely: Followed (github.com/RudraNilBasu/AsylumJam2016), Fortior (github.com/RudraNilBasu/Fortior) and a Quiz game (github.com/RudraNilBasu/Quiz-Game) to name a few.

All of my open source projects can be found on my Github profile: github.com/RudraNilBasu

Besides this, I have actively taken part in online and offline programming competitions, hackathons and have been involved in video game development, taking part in numerous online and offline game jams.

Blog: rudranilbasu.me/blog

Contact Information:
Email ID: rudra.nil.basu.1996@gmail.com
Freenode IRC nick: rudra