AALBORG UNIVERSITY
STUDENT REPORT

Developing Complex Software Systems

GIRAF - Piktotegner

Semester Project by sw608f14

from 3-2-2014 to 28-05-2014

AALBORG UNIVERSITY
STUDENT REPORT

third study year
Computer Science and Software
Selma Lagerlöfsvej 300

Title: Developing Complex Software Systems

Theme: GIRAF - Piktotegner

Project period: Spring semester 2014

Project group: sw608f14

Participants:

Daníel Steínar Friðjónsson
Lars Andersen
Mathias Winde Pedersen
Søren Skibsted Als

Supervisor:

Jinling Jiang

Copies: 6

Total Pages: 68

Appendix: 9

Completed: 28-05-2014

Abstract:

We develop a pictogram application, *Piktotegner*, which is a part of a larger software package called *GIRAF*. *GIRAF* is an autism communication software package developed for android devices. The pictograms made in *Piktotegner* can be saved and used in the other applications of *GIRAF*.
We determine how to rotate and resize drawn objects as well as the thought process of designing an intuitive user interface.
Finally, a description of collaboration with other teams of the *GIRAF* project is given, as well as the outcome and the experience gained thereof. This description includes working in a large system setting and collaborating with other teams to develop a shared library.

Foreword

This report was made at Aalborg University in the sixth semester of the Software study by the group

Daníel Steínar Friðjónsson _____

Lars Andersen _____

Mathias Winde Pedersen _____

Søren Skibsted Als _____

Contents

1. Introduction

Disabled people need help and this help is nowadays given through the use of technology. However, there are still some disabled people who are using non-technological solutions to aid their needs. Among these people, are people suffering from the mental disability known as autism. There are many different forms of autism though, ranging from super intelligent to be unable to speak. A shared trait though is the inability to put themselves into the mind of others. The result of this is that they cannot interpret emotions, causing them to have difficulties when interacting with other people in social contexts. The main focus is to help people that are having the strongest negative effect of autism, which is primarily the ones who have trouble with speech. Currently, folders with pictures are being used to communicate with the guardians. These folders are very cumbersome since they can grow very large in size. Furthermore, it can be very difficult to find the specific picture you are looking for. This is sought to improve through technology by creating an application on a mobile device that fulfils the same functionality as the folders. By creating this application, it is possible to have the pictures located at one place without having it grow in size. In addition to this, it could be easier for the people to find the specific picture by creating a search function in the application.

Currently this project is an ongoing development, which is built upon the work of previous generations of students. Therefore, the project will primarily focus on getting the current features work as intended, with potential for new functionality if there are no further improvements available to the current features. The development is done in four SCRUM sprints detailing the process of each.

2. Role in the Multi Project

Our role in the multi project environment is to improve on the foundations created by the previous developers. These improvements can be fixing bugs in the existing code, integration with other projects in the multi project, or adding new functionality to match the customers request and expectation. This goal is the reason that the focus is on developing working code, which can be released to the customers, rather than optimal code that is not ready for the customers.

2.1 Analysis of Organisational Context

In the multi project, there is around 60 people working together. To make this teamwork manageable, some organising is needed. Organising is done by utilising the SCRUM methodology, and since there are so many groups, a SCRUM of SCRUMS is used. This means that there are daily SCRUM meetings, weekly meetings between the SCRUM teams, and meeting with the customers at the end of each sprint. The goal is to create an application that aids autistic people and their guardians in communication. Each group gets a part of the application and further develops on it. The decisions involving parts of the application are put in the hands of the group responsible for said part, while decisions involving the overall application are discussed at the weekly meetings.

At the start of the development process, we had several meetings, with everyone present, where we did a transition with the previous development group to the current one. During these early meetings we discussed how to organise the multi project and did decision based on our own knowledge and the previous development group's experience.

Tools for managing the development were also decided at these meetings. GitHub was chosen as our configuration management tool, as this is easily available, a lot of developers have experience with it, and the previous group also worked with it, which made the transition easier.

Redmine was chosen as the main information sharing tool in regards to guides on how to use various parts, different standards such as the ones used for coding and meetings. Redmine was also chosen because it had a great issue tracker, allowing developers to easily check what was being developed on the other parts and to check their progress.

Another tool we used was Jenkins, which automatically build the project and alerts certain people when the build fails. Jenkins was also used to get the newest version of a part of the project, without the need to download the repository of the project and compile it.

The chosen IDE was Android Studio, which was suggested by the previous development group. Android Studio is an IDE designed specifically for Android devices making it a suitable choice.

Personnel from the development group were chosen as being in charge of the tools. It was also the responsibility of these people to be knowledgeable of their tool.

As mentioned before, we had planned meetings with the customers at the end of each sprint, where all the parts of the application were presented with focus on the newly developed features. At the start of the sprint we did another meeting with all the developers present, where decisions were made regarding what parts each development group was to develop in the coming sprint.

3. Sprint 1

The first sprint of this project was decided to run from 20th of February to the 19th of March. The primarily focus of this sprint for all groups was to get the existing code to work. For this project the code chosen was the *Croc* application developed in spring 2013, which is responsible for creating pictograms.

3.1 Croc

Croc is a part of the *GIRAF* software package. It is meant to support manual creation of pictograms. In that regard, to create a pictogram, the user can take pictures, record audio, and draw their own pictures. These options can then be used in any combination as the user sees fit for a given pictogram.

The target audience are guardians and parents, contrary to other parts of *GIRAF*. For that reason, the design focus is on usability for guardians as well as to make suitable pictograms, rather than educational or entertainment purposes.

Croc is developed as an application that may be launched from other applications in the *GIRAF* project. When a pictogram has been created in *Croc* it is meant to be stored in a database to make it accessible from other applications.

In order to achieve these goals of *Croc*, features has been implemented although not without issues.

Features

A lot of features of *Croc* are tied to the user interface, and for that reason it makes sense to present the *Croc* UI. In order to get an idea of how the *Croc* UI looks like, see Figure 3.1.

The UI is divided into two parts, the upper serving as a menu-bar allowing access to the camera, audio recorder, help view, closing the application, or saving the pictogram.

The lower part changes depending on the mode chosen. In the figure, the lower part shows the UI for the following features.

- Selection
- Freehand, rectangle, line, and circle drawing
- Colour changing
- Load from Camera
- Colour preview
- Stroke width

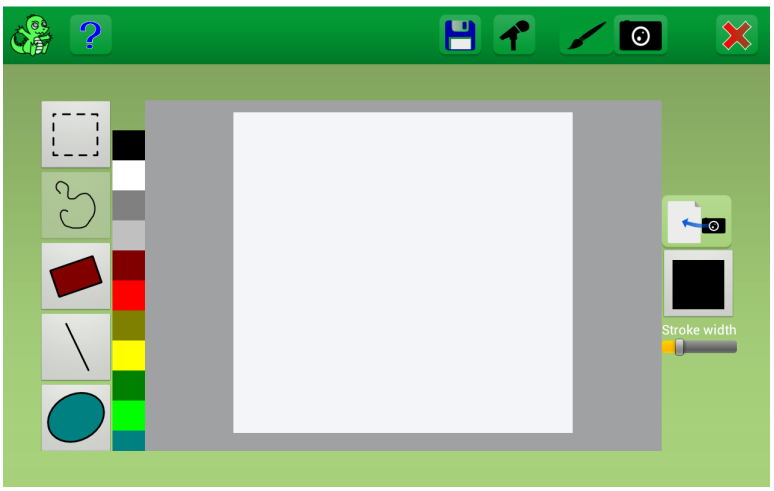


Figure 3.1: Screenshot of canvas UI.

Other features exists for the camera and audio recording, which include switching between black/white and colour pictures, take picture, start and end recording. For screenshots of the camera and audio recorder, see Appendix A.

3.2 Sprint Backlog

As mentioned before, in this sprint we will add missing features and resolve bugs in the application we have been assigned to. Some of these issues are found by the group working on the program last year, and the rest have been found by testing the functionality of the application. The different issues found are prioritised and estimated through poker planning. By this estimation and prioritisation, the following sprint backlog was constructed:

Change stroke width
Be able to change the stroke width for freehand drawing.

Loading pictures from camera in correct size
Pictures loaded into the canvas are zoomed in too much, as the picture is larger than the canvas.

Colour swap
The colour swap button does nothing at the moment. The fill and stroke colour should be able to be swapped by a click on the colour swap button.

Straight line colour
Straight lines can only be drawn as black, it should be able to be drawn with ones selected colour.

Icon changes
In general several icons do not represent their functionality, therefore they need to be changed.

Freehand Drawing Colour
An unexpected swap in the colour of the last freehand drawing when

swapping colours with the colour swap button occurred and should be fixed.

Clear button
Implement a button in canvas to clear the canvas.

Collision detection for rotated objects
The collision detection only works when objects are not rotated. It needs to be changed to also work for rotated objects.

Save canvas state
Find a way to save the drawStack between uses. That is when switching to camera and back, or between app uses.

Record Dialogue
Change dialogue box for recording, as the visuals does not match the expectations of how a sound recorder works.

Scale camera pictures
Pictures taken with the camera should be scalable in canvas.

3.3 Implementation

With issues assigned to the sprint backlog of sprint one, a description of the issues and their corresponding solutions follows hereafter.

Change Stroke Width

Changing the stroke width was not possible for a freehand line. The other entities already had the feature implemented in its *updateBuffer* function, which was called when the entity was created and assigned the stroke width to be that of the current handlers stroke width. To resolve this issue, we updated the stroke width in the *updateBuffer* function.

Loading Pictures from Camera in Correct Size

In order to fix this issue, it was necessary to look at the function that loads the pictures into canvas. Previously, the function took the picture as it was and loaded it into canvas. Therefore it was imported in a too large scale, which could not fit into canvas. The issue was fixed by implementing a new constructor when creating a picture. The new constructor, seen in List 3.1, was similar to the old constructor but with the option of including a scale factor as input that is used to change the width and height of the picture by this factor.

Colour Swap

To resolve this issue, the class diagram was used to identify a connection between the preview button and the draw view. Two connections are found between *DrawView* and *PreviewButton*, which is *ColorButton* and *DrawFragment*. *ColorButton* is used when picking a new colour, and assigns the new colour to *PreviewButton*. However, when pressing the *PreviewButton*, no connection existed. In order to correct this issue, the event for touching *PreviewButton* could be accessed from *DrawFragment* and was already implemented for the other buttons.

FiXme Note: Overvej om denne del skal skæres fra, eller om den giver værdi i form af vores analyse af problemet.

```
1 public BitmapEntity(Bitmap src, int size ) {
2   internalBitmap =
        Bitmap.createScaledBitmap(src,src.getWidth() * size / 100,
3       src.getHeight() * size / 100, true);
4
5   setHeight(internalBitmap.getHeight());
6   setWidth(internalBitmap.getWidth());
7 }
```

Listing 3.1: New Constructor for *BitmapEntity*

In order to correct this issue, the *onTouch* event was moved from *PreviewButton* to *DrawFragment*, since *DrawFragment* had access to both *DrawView* and *PreviewButton*. *DrawFragment* was already subscribed to the other buttons, thus it was a logical change to move the touch event of *PreviewButton* to *DrawFragment*. The event implemented in *DrawFragment* can be seen in List 3.2.

```
1 private final OnClickListener onPreviewButtonClick = new
    OnClickListener() {
2     @Override
3     public void onClick(View v) {
4         previewButton.swapColors();
5         drawView.setFillColor(previewButton.getFillColor());
6         drawView.setStrokeColor(previewButton.getStrokeColor());
7     }
8 };
```

Listing 3.2: onPreviewButtonClick event

Straight Line Colour

The straight line colour was changed to be that of the fill colour instead of the stroke colour. The problem was that when picking a colour and then drawing a line, the colour of the line would be that of the stroke colour, instead of the picked colour. This was deemed non intuitive, and thus the colour used to draw the line was change to the fill colour. The issue was fixed by changing a method call to take the fill colour instead of the stroke colour.

Icon changes

FiXme Note: En af de subsections vi kan fjerne hvis vi har for mange sider

The previous development group working on this application [1] performed usability tests and through these, found that some of the icons was not intuitive to the users and thus needed changing. When changing the icons, the focus was to make them easier to associate the functionality to the icons. The idea behind the new choice of freehand drawing icon is what people associate with this functionality and how it is represented in similar applications. In such similar applications the standard representation of a freehand drawing is a pencil. Due to this standard the icon in Figure 3.2b was replaced by Figure 3.2a.

The previous icon for the selection tool could give a misleading impression of the functionality. The icon, seen in Figure 3.3b, looks much like a tool that

(a) New freehand drawing icon (b) Old freehand drawing icon

Figure 3.2: Freehand drawing icons

marks an area, but the tool selects just one object. To make the icon represent the functionality of the feature a new icon was deemed necessary. The new icon was made to look like a hand, as this gives a clear representation of being able to select one object at a time, see Figure 3.3a.

(a) New selection icon (b) Old selection icon

Figure 3.3: Selection icons

Freehand Drawing Colour

FiXme Note: Optional subsection også

Before the colour swap functionality was implemented, it was not possible to change the colour of the freehand drawing. However, with that feature implemented a bug occurred when using the preview button to swap colours after drawing a freehand drawing. By using the button it resulted in swapping the colour of the latest freehand drawing. The bug happened since the latest drawn object was not released after being drawn. It was not released for the freehand line, since the freehand entity overrode that functionality from the superclass, and as of such it had to be added.

Clear Button

Before adding the clear button, to clear the canvas you had to either clear each entity individually or to restart the application. As this was highly undesired, we decided a clear button was necessary. It is decided that the icon of a trash bin is the most fitting, as it is our understanding that most people associate the trash can icon with throwing something away.

In order to clear the canvas it is a matter of clearing the *drawStack*, deselecting selected entities and redrawing the canvas through the *Invalidate()* method, which is performed when the clearing action is accepted.

An unintentional click of the clear button would be of great nuisance to ones work. Thus, a dialogue box is added, requesting final confirmation for the clear operation.

Collision Detection for Rotated Objects

The user should be able to select the object by clicking inside of the hitbox, even when the object is rotated. This gave an issue, as when rotating an object the hitbox did not follow and to click on the object you had to click inside the misplaced hitbox. Originally before we changed the hitbox, when rotating an object, the hitbox would move out of the canvas, and thus not targetable. As the previous implementation was out of the question, its code was scrapped and the hitbox location remained stationary.

The reason for this was that when the object was created the hitbox was set but never changed after rotating the object. Therefore, a change was made to the hitbox property such that the top left corner of the hitbox was set to the minimum x and y value of an object's coordinates.

To find the new points of the rotated object we used the rotation matrix to rotate the current corners of the object. As seen in List 3.3 the minimum x and y values of these corners are then found, and used to find the top left corner of the hitbox. Once this corner has been found, the width and height can be calculated by using the corner and the center of the object.

In Figure 3.4 the hitbox can be seen around the hitbox both when rotated and not rotated.

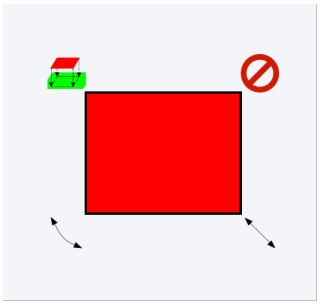
```
1  protected void changeHitbox(){
2          FloatPoint one = rotationMatrix( -(getWidth()/2),
               -(getHeight()/2));
3          FloatPoint two = rotationMatrix((getWidth()/2),
               -(getHeight()/2));
4          FloatPoint three = rotationMatrix((getWidth()/2),
               (getHeight()/2));
5          FloatPoint four = rotationMatrix( -(getWidth()/2),
               (getHeight()/2));
6
7          hitboxTopLeft = new FloatPoint(
8              findMin(one.x, two.x, three.x, four.x),
9              findMin(one.y, two.y, three.y, four.y));
10         hitboxWidth = (getCenter().x - hitboxTopLeft.x)*2;
11         hitboxHeigth = (getCenter().y - hitboxTopLeft.y)*2;
12 }
```

Listing 3.3: Method to change the hitbox

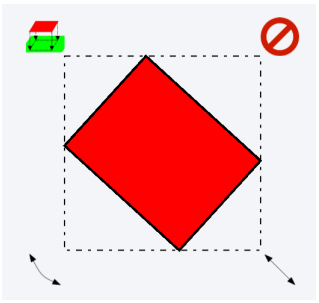
A new issue that emerged was that the collision detection for the object was now larger than the object. When an object was behind a rotated object, and the user clicked in the top left corner of the hitbox, the rotated object would be selected instead of the object behind. To resolve this issue, an implementation was done to give a precise collision detection on the object, which is described below.

Precise Collision Detection for Rotated Rectangles

To understand the mathematics behind the precise collision detection for rotated rectangles, an illustration of the problem can be seen in Figure 3.5.



(a) Hitbox for standard rectangle



(b) Hitbox for a rotated rectangle.

Figure 3.4: The different hitboxes.

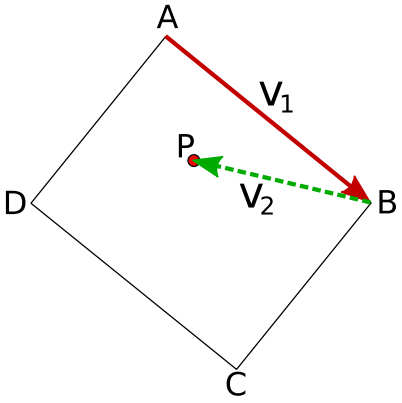


Figure 3.5: Vectors showing a right rotation.

The idea behind the mathematical formula is to see whether the vector to the point is a right rotation of all the rectangle's sides. To determine this we compare the slope of the vectors. The formula to determine a right rotation is:

$$\frac{B_x - A_x}{B_y - A_y} \geq \frac{P_x - B_x}{P_y - B_y} \tag{3.1}$$

The formula is changed to avoid divide by zero errors, and is therefore changed to the following formula:

$$(B_x - A_x) * (P_y - B_y) \geq (P_x - B_x) * (B_y - A_y) \tag{3.2}$$

Formula (3.2) is then used with all four vectors of neighbouring corners of the rectangle to determine whether a point is inside the rectangle or not.

Precise Collision Detection for Rotated Ellipses

To get a precise collision detection for rotated ellipses, we used the formula of an ellipse in standard form with the clicked point as input to check whether it is inside of the circle. If the clicked point is inside of the ellipse, the formula of an ellipse in standard form is smaller than or equal to 1 as seen in Formula (3.3).

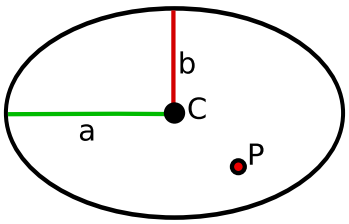


Figure 3.6: Vectors showing a point in the ellipse and the two radii.

$$\frac{P_x^2}{a^2} + \frac{P_y^2}{b^2} \leq 1 \tag{3.3}$$

To use Formula (3.3), we first have to accommodate for the rotation of the ellipse. If the ellipse is rotated, we rotate the clicked point around the centre of the ellipse and compare it with an unrotated version of the ellipse. The rotation of the clicked point is done by use of the rotation matrix, as seen in Formula (3.4). An ellipse has two radii which are used in the formula and illustrated as a and b, as seen in Figure 3.6.

$$\begin{aligned} P'_x &= P''_x * cos(\alpha) - P''_y * sin(\alpha) \\ P'_y &= P''_x * sin(\alpha) - P''_y * cos(\alpha) \end{aligned} \tag{3.4}$$

The variable P'' is the vector from the centre of the ellipse to the clicked point. This formula can now replace the coordinate in the standard formula, which results in Formula (3.5).

$$\begin{gathered} \frac{P_x^2}{a^2} + \frac{P_y^2}{b^2} \leq 1 \\ \Downarrow \\ \frac{P''_x * cos(\alpha) - P''_y * sin(\alpha)}{a^2} + \frac{P''_x * sin(\alpha) - P''_y * cos(\alpha)}{b^2} \leq 1 \end{gathered} \tag{3.5}$$

With this final formula derived, we can use it in the application to check if the user has clicked on an ellipse or not.

Precise Collision Detection for Rotated Lines

The last collision issue was a precise implementation to click on a line. We discussed how to click on a line and agreed that the click is allowed to be slightly off the line, since it otherwise was hard to click on. An idea was to calculate the distance from a point to the line and add a click range in order to determine if the click was meant to target the line.

To find this distance we used the formula for finding the area of a triangle, since the three points, clicked point, start point, and end point of the line, formed a triangle. The distance is determined by the height of the triangle, from the base to the clicked point as seen in Figure 3.7.

There are two equations for finding the area of a triangle, as seen in Equation (3.6) and Equation (3.7).

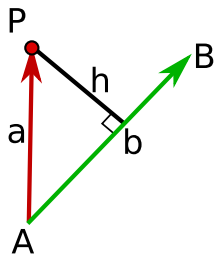


Figure 3.7: Drawing showing the idea behind the mathematical formula.

$$A = \frac{1}{2} * h * b \tag{3.6}$$

$$A = \frac{1}{2} * |\vec{a} \times \vec{b}| \tag{3.7}$$

We combine these two formulas since we are not interested in the area but the height h. Once combined, we isolate the height h and find a final equation for the height, as seen in Equation (3.8).

$$\begin{gathered} \frac{1}{2} * h * b = \frac{1}{2} * |\vec{a} \times \vec{b}| \\ \Downarrow \\ h = \frac{|\vec{a} \times \vec{b}|}{b} \end{gathered} \tag{3.8}$$

Another note to mention, is that the length of the vector $\vec{b}$ is the same as the base b.

$$|\vec{b}| = b$$

Since Equation (3.8) is based on given vectors and lengths, and the variables given in the application are coordinates, we have to change the formula to calculate the vectors and lengths from the coordinates. As can be seen in Equation (3.9), the vector $\vec{a}$ is equal to the difference in the clicked point P and the start point A.

$$|\vec{a} \times \vec{b}| = \begin{bmatrix} P_x - A_x \\ P_y - A_y \end{bmatrix} \times \begin{bmatrix} B_x - A_x \\ B_y - A_y \end{bmatrix} \tag{3.9}$$

From Equation (3.9), we can see how the cross product of the two vectors is determined, given the three coordinates. We can now replace that with the cross product in Equation (3.8), and add the equation for finding the length of the vector $\vec{b}$ in the denominator, as seen in Equation (3.10). By expanding the brackets in the numerator, we get a final equation for finding the distance from a point to a line that goes through two points.

$$
\begin{aligned}
h &= \frac{|\vec{a} \times \vec{b}|}{|\vec{b}|} \\
&= \frac{|(P_x - A_x)(B_y - A_y) - (P_y - A - y)(B_x - A_x)|}{\sqrt{b_x^2 + b_y^2}} \\
&= \frac{|P_x B_y - P_x A_y - A_x B_y - P_y B_x + P_y A_x + A_y B_x|}{\sqrt{b_x^2 + b_y^2}}
\end{aligned} \tag{3.10}
$$

Since the equation finds the distance from the clicked point and a line that goes through two known points, we have to limit the line so it does not pass the two points. To do this limitation, we use the hitbox of the line so that the click also has to be inside of the hitbox.

Save Canvas State

FiXme Note: Så kort at den er optional

To fix the issue, the *drawStack* was saved as a singleton, which made a single *drawStack* instance available across modes, such that even when the canvas was restarted, the *drawStack* could be reloaded. The canvas will still be cleared if the process is terminated, however, for the given issue the singleton will suffice.

Record Dialogue

The record dialogue contains two issues, the UI of the dialogue and the functionality to hear your recording.

For the UI, the main issue was an unintuitive icon for recording, discovered by usability tests performed by the previous development group on this application [1]. To fix this issue, we decide to make a red circle icon, similar to the look of other recording buttons. Furthermore, the record button changes to a darker colour when recording.

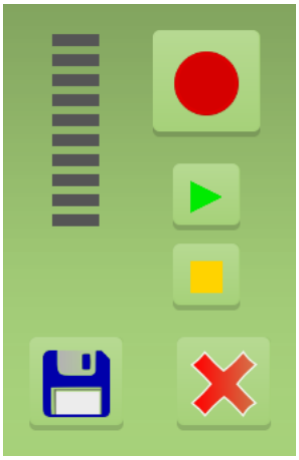


Figure 3.8: Record Dialogue window.

To get an idea of the dialogue box look, see Figure 3.8. As can be seen in the dialogue box, a play and stop button are also added to preview ones recording, which is a feature that was non existent in the application before this

sprint. The functionality of these buttons are implemented with the *SoundPool* android library, which allows for playing of audio.

Scale Camera Pictures

When a picture in the canvas is selected, it shows the resize button, and when the resize functionality is accessed, it resizes the hitbox, but it does not change the size of the picture. In order to make this resizing possible, a new method is created. This method takes a width and length as input, and recreates the picture based on this information. This method is then called four places in the code.

FiXme Warning: skriv specifikt hvor de her calls er.

3.4 Sprint Closure

To conclude, the sprint has been successful, as the issues for the sprint have been resolved. An additional issue was found. The stroke width of an entity was not taken into account when determining collision detection, resulting in squashed boxes with a large stroke not being selectable, and should be fixed for the next sprint. In addition to this, the application also shows a tendency to crash when a large amount of objects are drawn, as the heap runs out of space to store the objects. Furthermore, issues from the product backlog which was not in the sprint backlog should be considered for future sprints. Another thing to note, is that the UI changes that was made require usability tests in order to verify that the assumptions made regarding what is intuitive, actually correspond to the users' understanding.

FiXme Warning: skal der vaere en liste over product backlog her? med remaining is that is

4. Sprint 2

FiXme Warning: Vi er for meget på det redegørende niveua, fix it.

The second sprint of this project was decided to run from 20th of March to the 14th of April. The primary focus of this sprint was to resolve issues in the running application. *Croc*, which has been renamed to *Pictocreator* is still the application being worked on.

4.1 Sprint Backlog

For sprint 2, the sprint backlog consists of the following.

Load existing audiofile to the record fragment
After recording an audio file and exiting the record fragment, the recorded file can not be played when you enter the record fragment again, which it should.

playButton press before no recording is performed
The play button should be deactivated when no recording exists.

Change Pictocreator icon
Needs to be changed to a pencil drawing on a paper, as requested by the customer.

Record dialogue GUI change
Needs to be changed to one record button that switches icon to a stop icon when recording. In addition have a single toggle button for playing and stopping of audio. This was a request from the customer.

Colour settings from Launcher use in Pictocreator
Apply colour settings parsed from Launcher in *Pictocreator*.

Update GUI Components
Update GUI such that it uses *gComponents*, developed by another group in the *GIRAF* project.

Save pictogram
Save pictogram in database and its associated sound.

Load pictogram from database
Load a pictogram from the database into the canvas.

Flatten Button
Give flatten button the functionality to move selected entity to back of the canvas.

Pictogram title in text on pictogram
Customers have expressed interest in having the title of the pictogram written on the pictogram on either the top or the bottom.

Tags for save dialogue
Be able to add tags to ones pictogram when saving it and support this in the GUI.

Rotation of Freehand Entities
Freehand entities are rotating around their start point, but should rotate around the centre of the entity.

Collision detection for high stroke width objects
The current collision detection does not take the stroke width into account. This means that objects with a large stroke width are very hard to hit, if their center area is very small, even though one could swear they hit the area it covered.

Freehand collision detection
The collision detection should work so if you click the freehand drawing it should be marked.

Freehand hitbox
The hitbox have to follow the rotation, so the hitbox works as the line's hitbox.

Preview Pictogram
When trying to save a pictogram, the preview sometimes does not load.

4.2 Implementation

With the sprint backlog for sprint 2 specified, here follows a description of how the main issues were resolved.

Load Existing Audio File to the Record Fragment

A pictogram can have a single sound file associated. As of such it is reasonable to make the saved sound file path static.

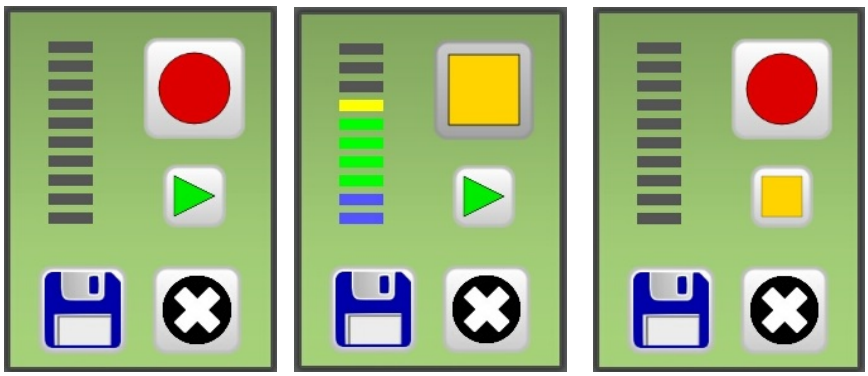
As the sound file path is made static, it just needs to get assigned a path the first time the record dialogue is added. Else you replace that file each time you save a new recording.

One issue occurred, which is associated with a play button press before no recording is performed. This issue is fixed by checking if a recording already exists at the static path. If a recording exists at the path the recording is played, if not, a toast is made telling the users that there is no recording.

Record Dialogue GUI Change

By satisfiability testing it is found that the Record Dialogue should be changed again.

The record dialogue GUI to be changed is the one seen in Figure 3.8. However, the customers desires fewer buttons, and point out that the play and stop button can be combined into one button, as they are mutual exclusive and are causing confusion.



(a) Idle mode (b) Recording mode (c) Playing mode

Figure 4.1: Record Dialogue GUI

To support this change, the audio player is changed to the Android *MediaPlayer* instead of *SoundPool*, as *MediaPlayer* supports an event for registering when a sound is finished playing. The new GUI can be seen in Figure 4.1. It was shown to the customers and they found it reflected their requirements.

Colour Settings from Launcher Use in Pictocreator

The colour settings originally parsed from the Launcher application can be applied to the layout with a gradient effect by use of the HSV colour encoding. Later, the Launcher group decided that this colour schema from the Launcher should not be applied after all, but the following description serves as a documentation of how it was implemented.

We make a gradient colour with use of the HSV encoding format with adjustment of the value parameter, to give a better depth to the background layout. If the application is not opened from Launcher, or the Launcher does not parse a colour value, a default colour scheme is applied.

Update GUI Components

This issue is a general issue for all the applications in the project. A group was assigned to create a GUI framework to the project so that the GUI has the same look in all applications.

To implement this framework, we have to add the path for the GUI-components such that they can be used in the XML-files that define the GUI. Each existing button is changed to a matching gComponent and its functionality changed to work with the new framework. This implementation resulted in deletion of several XML-files that were used to create the previous buttons, their functionality are now taken care of by the framework.

The clear dialogue changed more drastically than the rest of the interface, as the framework also contains a method to create simple confirmation windows. This change resulted in removal of the clear dialogue XML-file, as it is no longer needed.

Save Pictogram

A pictogram has some needed attributes that have to be saved together with the pictogram. These attributes can be assigned in the save dialogue fragment and include the publicity of the pictogram, public or private, the tags following the pictogram, the inline text, and the name of the pictogram. To save the pictogram, we first create a pictogram object and assign the needed attributes to it. The sound is saved in the cache of the tablet in another dialogue, but now it is assigned to the pictogram object so it can be saved in the database. Furthermore, the author's id is added as well as the bitmap from the canvas. Once the attributes have been assigned, the pictogram is stored in the database via the provided methods from the database. The database gives the pictogram an id and returns it back, so it can be used to create the relation between tags and pictogram.

Load Pictogram from Database

The application should be able to load existing pictograms, so they can be modified to accommodate desired changes. To do this we decided to use *Pictosearch*, which is an application that is also under development in the *GIRAF* project. *Pictosearch* is a search tool that can find pictograms stored in the database. The chosen pictogram is then sent to calling application, which in this case is *Pictocreator*.

The call to start *Pictosearch* also sends the guardian's ID, to allow access to their private pictograms, as well as the purpose of the call which is *single* meaning *Pictosearch* can return at most one pictogram, as seen in lines 5 – 9 in List 4.1. If *Pictosearch* is not installed, a message will inform the user about it. An event to receive the pictogram is create, called *onActivityResult*, and checks whether it return a results or not. If there is a result, another function is called which loads the pictogram bitmap from the database using the pictogram ID returned from *Pictosearch*, as seen in line 5 in List 4.2. The bitmap is then loaded into the canvas in the original scale.

```
1  private void callPictosearch(){
2      Intent intent = new Intent();
3
4      try{
5          intent.setComponent(new ComponentName(
               "dk.aau.cs.giraf.pictosearch",
               "dk.aau.cs.giraf.pictosearch.PictoAdminMain"));
6          intent.putExtra("currentGuardianID", author);
7          intent.putExtra("purpose", "single");
8
9          startActivityForResult(intent, RESULT_FIRST_USER);
10     } catch (Exception e){
11         Toast.makeText( this, "Pictosearch er ikke
               installeret.", Toast.LENGTH_LONG).show();
12     }
13 }
```

Listing 4.1: Method used to launch *Pictosearch*.

```
1  private void loadPictogram(Intent data){
2      try{
```

```
3           int pictogramID =
                data.getExtras().getIntArray("checkoutIds")[0];
4
5           Bitmap pictogram = (PictoFactory.getPictogram(this,
                pictogramID).getImageData());
6
7           drawFragment.drawView.loadFromBitmap(pictogram);
8       }catch(Exception e){
9           Log.e(TAG, e + ": No pictogram returned.");
10      }
11  }
```

Listing 4.2: Method to load a pictogram from Id received from *Pictosearch*

Flatten Button

The flatten button functionality is implemented as seen in List 4.3.

```
1  public void moveToBack(Entity entity){
2      int index = entities.indexOf(entity);
3      entities.remove(index);
4      entities.add(entities.size(), entity);
5  }
```

Listing 4.3: Flatten Button functionality

As the entities are already stored in a list, it is a matter of moving the entity to the end of the list. With the flatten button functionality implemented, the order in which a user draws entities can be disregarded, as the order can be changed.

Pictogram Title in Text on Pictogram

When talking with the customers, a need for adding text to pictograms was proposed. Different options for adding text was discussed, which included adding text to the pictograms themselves or in the dialogue in which the pictogram is saved. The customers agreed that adding text in the save dialogue made the most sense, and thus text on the pictogram itself was discarded in the first product version to be used by the customers.

The customers want the title of the pictograms to be in changeable positions. This issue is deemed as not being the responsibility of *Pictocreator*, as the text is merely stored together with the pictogram and can be placed wherever users of the pictogram want it.

Tags For Save Dialogue

Tags were partially added in the GUI but you could not add tags and they could not be stored in the database. To resolve this, tags are stored in an ArrayList and displayed in a ListView, such that when modifying the Arraylist the GUI is updated as well. Implementing the tags gives the opportunity of specifying keywords for the created pictogram, which can be searched for to find the pictogram later. Assigning the tags to a pictogram is done with use of a database helper provided in the developed Oasis database library.

Freehand Collision Detection

This issue was resolved by using the collision detection used for a line. However, the freehand drawing only contains points, so these points had to be combined into a line. All of these lines had to be compared to the collision detection from the line collision, see Section 3.3.

Preview Pictogram

Previously when loading the preview, it was handled in the draw view and was saved on the cache. This proved to be a problem, as it was unclear how the timing of saving to the cache worked, and using that storage was vulnerable to sudden cache clearings. In order to fix this issue, the Bitmap encoding of the pictogram was parsed to the save dialogue instead, which made the preview unaffected by sudden cache clearings.

4.3 Sprint Closure

The issues for the sprint have been resolved. In addition, contact was established with the customers, where they expressed an opinion on the current UI, which they wanted to have changed. This opens for a new issue for following sprints to change the UI of the application. Additionally, the customers expressed the requirement of loading old pictograms for editing, where the pictograms should not be overwritten, but save an extra modified version instead. Another application, *Pictosearch*, worked on by another group, is needed to search for pictograms. However, *Pictosearch* at its current state crashes at startup, and needs to be resolved before further work regarding loading of pictograms can be pursued. The application still crashes, but seems to have been reduced to memory issues where the heap runs out of space. It is believed that this can be minimise by reducing the amount of camera images stored in the cache to the ones taken for a specific pictogram. Furthermore, tests are believed to be crucial in future sprints to minimise bugs before a final release is published to the customers.

FiXme Note: samme spørgsmål som i sprint 1, skal current product backlog listes

5. Sprint 3

It was decided that the third sprint should run from the 15th of April to the 5th of May. The primary focus of this sprint was a major GUI overhaul. The secondary focus was adding a few missing functionalities. In addition to this, it was also decided that the application needed to be renamed once again, from *Pictocreator* to *Piktotegner*, since the customers are Danish, the name should also be in Danish.

5.1 Sprint Backlog

Consider new name for title and name in save dialogue
The name of the title and name in the save dialogue have to be changed so it is more understandable for the customer.

Resize rotated obejcts
Be able to rotate an object and resize it afterwards.

Resize
Resize on rotated objects is handled poorly and needs a fix.

Hitbox on rotated objects during resize
During resize of a rotated objects, the hitbox moves too much and needs to be stabilized.

Camera Fragment
The way import pictures worked through the camera was deemed unlogical by the customers, they desired something more akin to Instagram that did not require them to go into another fragment, but rather open sort of a dialogue box instead.

GUI - rearrange canvas
Rearrange all buttons in accordance with customer expectation.

Load drawstack
Change load functionality to load drawstack if the pictogram is saved by *Piktotegner*. This is only possible after the database has been updated to accommodate this change.

Text on buttons
There have to be text below the icon of every button.

Radio button for "public" save dialogue
Make two radio buttons for the save dialogue. The first should be for the private setting and the other for the public setting.

Custom Colour button

Need a button to pick a custom colour.

5.2 User Interface

A meeting was established with two of the customers, Drazenko and Pernille [2, 3]. From the meeting it was found that the general drawing interface seemed suitable, however, after they tried to use the application, some flaws were found. The flaws were mainly the way to take photos, which was deemed unintuitive, as you had to take multiple pictures and then later choose which one to load into the pictogram. Furthermore, a new position ordering of the drawing tools was requested. This was mainly a request from Pernille, as she found that options for the drawing tools should be located near each other. Additionally, the colour theme of the application was changed such that the graphical components developed by the *GIRAF Components* group was used.

The implemented version can be seen in Appendix B

Prototypes

Before the user interface (UI) changes are performed in the application, paper prototypes are created to get a first intuition of how the UI should look. These prototypes are based upon the talk with the customers [2, 3].

Rearrange UI Components

Based on the request from the customers, the UI components of the drawing surface should be rearranged. Two prototypes were drawn for this, which can be seen in Figure 5.1 and Figure 5.2. The idea of Figure 5.1 was to move the components that had to do with the creation of a pictogram down in the *drawFragment*. Furthermore, to arrange the options for the drawing tools near the tools themselves. However, it was deemed that the application became asymmetrical due to this change.

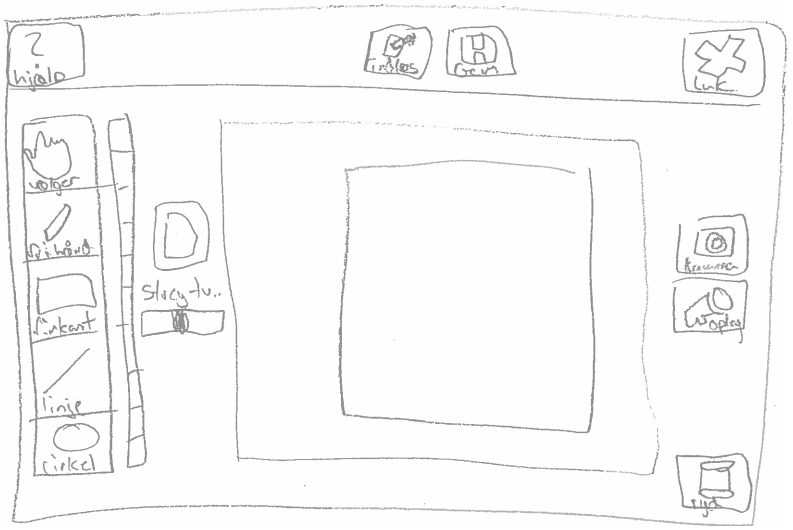


Figure 5.1: Version 1 drawfragment.

The idea of Figure 5.2 was to keep everything drawing related in the *drawFragment* and additional features in the top bar. In addition, the options of the drawing tools were separated from the tools themselves, to regain symmetry.

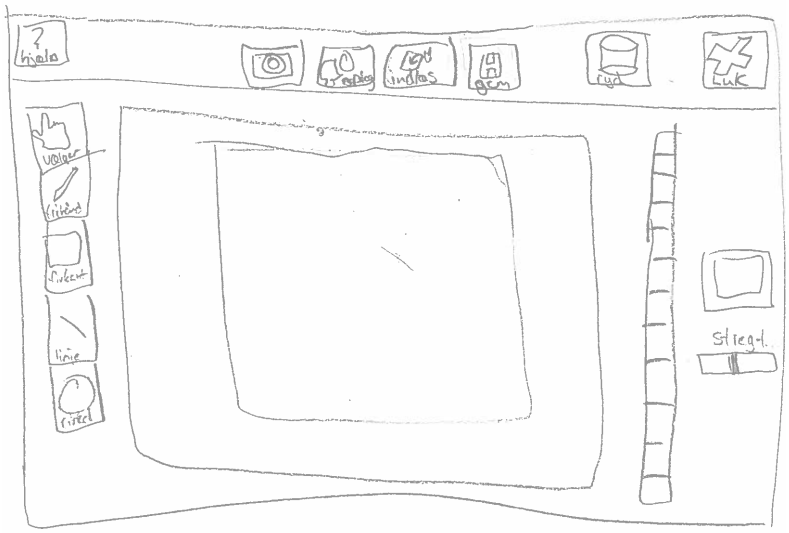


Figure 5.2: Version 2 drawfragment.

Parts of the two prototypes were found satisfactory, and as of such a combination of the UI was performed. It resulted in still having the separation of tools and their options, but making the *drawFragment* being a more general layout, featuring the creation of pictograms instead of the drawing part solely.

Camera Dialogue

The customers found it unintuitive to be able to take multiple pictures at a time and then later load some of the pictures for a pictogram. For that reason, the camera dialogue should be changed to only be able to take one picture at a time. When the picture is taken it can be verified, and then added to the pictogram, or discarded.

Figure 5.3 shows a prototype of the camera dialogue, where inspiration has come from popular image applications such as the inbuilt iOS camera and Instagram. Seen in Figure 5.3a is the dialogue as it should look like when preparing to take a picture of a tree. The button to the lower left is to swap between colour and black-white mode, which is a request of the customers. In the lower right corner, a button to swap between front and back camera can be seen. Finally, in the lower center, a button to take the picture can be seen.

FiXme Warning: reference her.

When the picture is taken, you are transferred to the UI seen in Figure 5.3b, where you have the option to accept the taken picture or to decline it, and return to the previous UI to take a new picture.

In the whole camera dialogue you always have the option to quit the dialogue by pressing the button in the upper right corner.

(a) Take Picture (b) Verify Picture

Figure 5.3: Camera Dialogue

Additional Work

The implemented version of the UI is not tested yet, and as of such can not be determined as final yet. Future sprints would have to focus on presenting the UI to the customers and perform usability tests to ensure that a good quality of the application is achieved.

FiXme Warning: discuss t title, seems weird

5.3 Implementation

In this section, the important issues from the sprint backlog of sprint 3 are detailed and explained.

Consider New Name for Title and Name in Save Dialogue

These two fields needed to have the text changed, as both the customers as well as other people had trouble understanding the difference between information contained in the two fields. The previous text in Danish, "titel på pictogram" and "pictogram navn", was changed to "tekst på piktogram" and "piktogram navn". The reason it is such a slight change is that the main confusion seemed to be the result of the use of the word "titel", Danish for title, and we think that by replacing it with the word "tekst", Danish for text, the meaning becomes more clear. It is not yet known if this is the case, as this change has not yet been checked with the customers, so there is a risk it changes nothing, but is checked in the next meeting with the customers.

Resize

Resizing an already rotated object would result in an odd and unintuitive behaviour. The problem was that the dragged vector just added its dimensions to the rectangles width and height. However, if the rectangle is rotated, the dragged vector would still add its dimensions to the rectangle as if the rectangle was not rotated. This issue would for example have a opposite resizing if the rectangle was rotated 90deg, as can be seen in Figure 5.4. To change this behaviour we tried some different ideas, which resulted in failure but will still be documented to show our effort towards this issue.

FiXme Warning: look at the figures again, make sure they are detailed enough

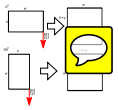


Figure 5.4: Example showing previous resize rotate combination

The initial approach was to rotate the vector with the same angle as the rectangle is rotated and then apply the dimensions as previously. However, this approach was mathematically flawed as the vector could end up with negative dimensions after rotation and decrease the size of the rectangle where the motion of the user intuitively should increase the size of the rectangle.

In an attempt to reuse the idea in the previous approach, we realised that the rotation of the vector should be around a specific line in 3D. An example to show this idea could be shown using $\vec{a} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, and let us say the rectangle is rotated by 90 deg. This means the vector has to be rotated 180° around a line that is tilted 45°, as illustrated in Figure 5.5, resulting in a new vector $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$. The problem with this approach was that we could not mathematically express where the line should be for the general case. We believe that if this line could be determined, this approach would be sufficient to solve this issue. However, we would also have to examine how much we should rotate around the to be determined line.

Since we could not find that line, we decided to take another approach and instead of looking at the drag vector, we look at how the hitbox changes. The drag vector was now applied to the hitbox and we looked at how much the hitbox would change in size. This size change would then be applied to the rectangle inside the hitbox, but had the same problems as in the original problem as when the rectangle is rotated above 90° its heigth should be changed when the hitbox width was changed.

Next approach expanded upon the idea of using the hitbox. We looked at where the edges of the rectangle connected with the hitbox and when the dimensions of the hitbox changed, we place these edges at the same ratios. For example, assume that the an edge of the rectangle connected with the hitbox with a ratio of $20\% - 80\%$, it should keep that ratio after the hitbox was resized. However, this approach would conceptually stretch the rectangle instead of resizing it, resulting in it no longer being a rectangle. This result could not be represented on the android device as the object was considered a rectangle at all times, due to this, it was never implemented. Yet again we took the idea in another direction, and looked at the width and height of the rectangle as vectors. The angle of these vectors would be divided by the angle of the drag vector, to find a factor of how much each side had to be affected. For example, let us assume that the width vector has an angle of 60°, the height vector an angle of 150°, and the drag vector 30°. The factor would then be found as follows:

$$\begin{aligned} w' &= v/w \\ h' &= 1 - w' \end{aligned} \tag{5.1}$$

where,

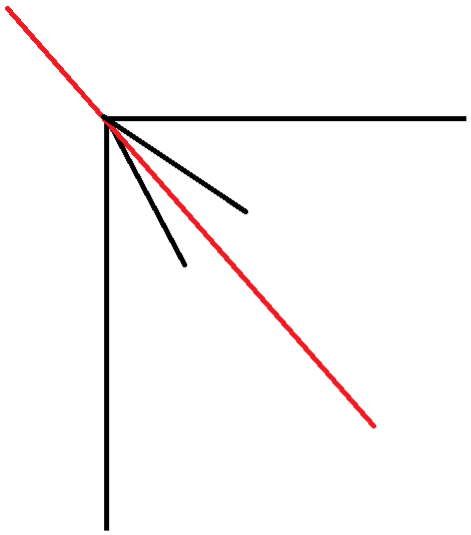
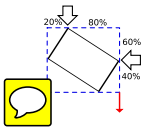
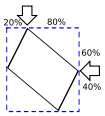


Figure 5.5: Example showing 2nd resize approach



(a) The hitbox is not resized yet.



(b) The rectangle stretches after resizing the hitbox.

Figure 5.6: Example showing 4th resize approach.

w is the angle of the width vector.

v is the angle of the drag vector.

w' is the factor of much the width should be affected.

h' is the factor of how much the height should be affected.

The equation would be opposite when the angle of the width vector was above 90°. With the above example the factors would be $w' = 0.5$ and $h' = 0.5$, which

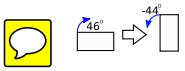


Figure 5.7: Example showing the current and possibly final resize approach

means half the length of the drag vector is added to the width and height. A problem occurs when the angle of the width vector would be a lot smaller or larger than the drag vector angle and for example add twice the length of the drag vector.

The current and possibly final approach is to reset the starting point of the rectangle and swap the width and height. This happens when the rotation angle of the rectangle is between 45° and 315°, which is corresponds to the possible angles to be between 45° and −45°. We implemented this approach since we observed that the resizing was reasonably accurate as long as the rotation angle was below 45°. As an example, let the rotation of the rectangle be 46°, this means the dimensions of the rectangles are swapped and the rotation angle is sat to −44°, as can be seen in Figure 5.7.

Custom Colour Button

The Colour choices available in the application were unchangeable as they had hard coded values. Due to this, a need for additional colour choices would arise, if a customer needed a specific colour that was not available. This functionality was added by using an already created implementation of a colour picker created by the group responsible for graphical components in the *GIRAF* project. The colour picker has a full spectrum of colours, once a colour is chosen it can be shaded. In addition to the colour picker, we created a button to allow for quick reselection of the last colour chosen in the colour picker. This button is added because we believe that having to go into the colour picker to reselect would be a nuisance.

Drawstack

After a pictogram is created and saved with *Piktotegner*, the customers want to be able to load it again, without loss of pictogram information. This information consists of the drawstack of a pictogram, such that different painting entities can be edited individually and not a singular bitmap. If the pictogram is saved in the database from some other source that is not *Piktotegner*, it is the singular bitmap you load.

Save Drawstack

The idea to save the drawstack on the database was to convert the drawstack into a byte array. Java can provide this service if you implement the *Serializable* interface, which is a keyword where you do not need any implementation if the data types of the fields are all *Serializable*. However, *RectF* and *Paint* from the Android Graphics Library did not implement the *Serializable* interface.

At first this was tried to be solved by making *Serializable* versions of those classes, which then extended *RectF* and *Paint* respectively but implement the *Serializable* interface. This method resulted in the drawstack being able to be converted to a byte array, however, the information of the stroke width and color of entities were lost. This was due to the *Paint* and *RectF* classes again including fields that were not *Serializable*. As of such, integers were introduced

to contain the information of those objects, in order to bypass the usage of those objects as fields and thereby be able to convert the drawstack to a byte array. *Paint* and *RectF* could be solved in this way, but when containing a camera picture represented as a bitmap in the drawstack, difficulty arose when converting to a byte array.

If the drawstack contains a bitmap, a solution is not as easy as introducing integers to hold information. A library is instead tried to manually convert the bitmap to a byte array and back again. However, for the current sprint a satisfying solution is not found, because if the bitmap had to retain a certain quality, the conversion was too resource demanding. As of such, for future work, a way to save the drawstack when a bitmap is an element thereof is needed. At present, the drawstack only gets saved if there is no bitmap for the pictogram, else the singular bitmap saving and loading is performed.

Load Drawstack

For loading a drawstack, it is not a problem, as the drawstack has already been saved as a byte array and the classes have been made *Serializable*. Then it is just a matter of using the developed method to convert the byte array to a drawstack, see List 5.1.

```
1 public static Object deserialize(byte[] data) throws
      IOException, ClassNotFoundException {
2         ByteArrayInputStream in = new
              ByteArrayInputStream(data);
3         ObjectInputStream is = new ObjectInputStream(in);
4         return is.readObject();
5 }
```

Listing 5.1: Method for converting byte array to object

5.4 Sprint Closure

The issues of sprint 3 have been resolved to some degree, due to usability tests not being performed in sprint 3. Additionally, the camera dialogue still needs some development, as a bug occur that results in stretching of the dialogue unintentionally.

The application was presented to the customers at the sprint end meeting, where general positive feedback was received. A few requests were proposed, such as the possibility of assigning pictograms to specific citizens instead of the whole institution. In relation to this, it was found that there is only a need of one guardian profile per institution, which is why the distinction between guardians is not necessary, but is kept as it is resolved in the guardians using a shared profile. This sharing of a profile is, however, a temporary solution, as the guardians should have their own profile with their private settings.

Furthermore, it was found that resizing of camera pictures should be changed to scaling, such that the aspect ratio is kept.

For the next sprint, these issues are focused on, in addition to usability and regular tests, to ensure a good application quality.

6. Sprint 4

It was decided that the fourth sprint should run from the 6th of May to the 20th of May. The main focus for this sprint was implement the final important problems that have not been solved in the previous sprints. Another focus for this sprint was to make the application ready to be launched for the customers. To make sure that the customers can use the application a usability test is performed.

6.1 Sprint Backlog

For sprint 4, the sprint backlog consists of the following.

Negative width and height during resize
Allowing negative width and heigh during resize gives chaotic behaviour and should thus be constrained.

FiXme Warning: Remove this one, it is already mentioned in sprint 2

Freehand entity rotation
Freehand entities have an odd rotational behaviour e.g. when drawn from bottom-left to top-right.

Bug in the camera fragment
A bug occurs when a picture is taken after the camera has been swapped between front and back camera, and when the camera fragment is accessed again there is no camera feed.

Change help pictures
Make new help pictures illustrating tutorials with the new user interface.

Changes to accessibility settings
Change the settings from private and public to be for citizen specific or available to whole institution.

Pictures should fill whole canvas
Pictures taken with the camera should fill the whole canvas.

Resizing camera pictures
Resizing camera pictures should be limited to scaling, such that the width and height ratio is kept.

Preview button with suitable display
Make the preview button show the selected tool instead of always a rectangle.

6.2 Implementation

In this section the main problems from the sprint backlog are detailed and explained.

Negative width and height during resize

Entities could be resized to negative sizes resulting in odd behaviours. To fix this issue, we constrained the width and height of the entities to not go below zero. If that happens they are sat to zero.

Bug in the Camera Fragment

When using the camera to take pictures after the camera has been swapped between back and front camera, a bug was found. The bug was that the next time the camera fragment was accessed, the camera was not available and had to be solved. The bug was determined to be that the camera was not released when the camera fragment was closed. Two surfaces were used in the camera fragment, so the releasing of camera could not only be performed in the *onPause* method, but had to be added to the *surfaceDestroyed* method.

Change Help Pictures

Piktotegner has grown to be an application with numerous features for creating pictograms. As of such, due to the scale of the application, people may be confused when using the application the first time. The help pictures come as a possible solution to this, where the user can open the help menu and look at help pictures to get a better understanding of how the application works. Help pictures already exists from last semester, but have become outdated due to UI and functionality changes, which also gives a reason for this update.

The help pictures are structured as a mini sequence, such that each help picture contains two images, aligned top and bottom. The top image shows where to press to perform some action and the bottom image shows the result of performing that action. In total, 24 new help pictures have been made and to view them you browse through them with a left and right arrow button. An example of such a help picture can be seen in Figure 6.1.

Changes to accessibility settings

A decision was made to create a feature which could add a pictogram to specific users in the system. Previously we have the option of assigning the pictogram to the user, as private, or available to everyone, as public.

We changed the public to be institution wide, meaning the pictogram was available to the members of the institution. The private option was changed to be available for specific citizens, chosen from a list which contains the citizens assigned to the currently logged in guardian. To avoid confusion, it was decided that the button to add citizens and the list with the citizens would be hidden until the user checks the citizen button.

This change required a significant overhaul in the GUI of the save dialogue, the resulting GUI can be seen in Figure 6.2, where the citizen button is checked and two citizens are chosen.

When the button *Tilføj Borger* is clicked a new dialogue will show up where the citizens can be clicked, as seen in Figure 6.3, where the darker fields are chosen whereas the light is not. The dialogue is a component from the GUI

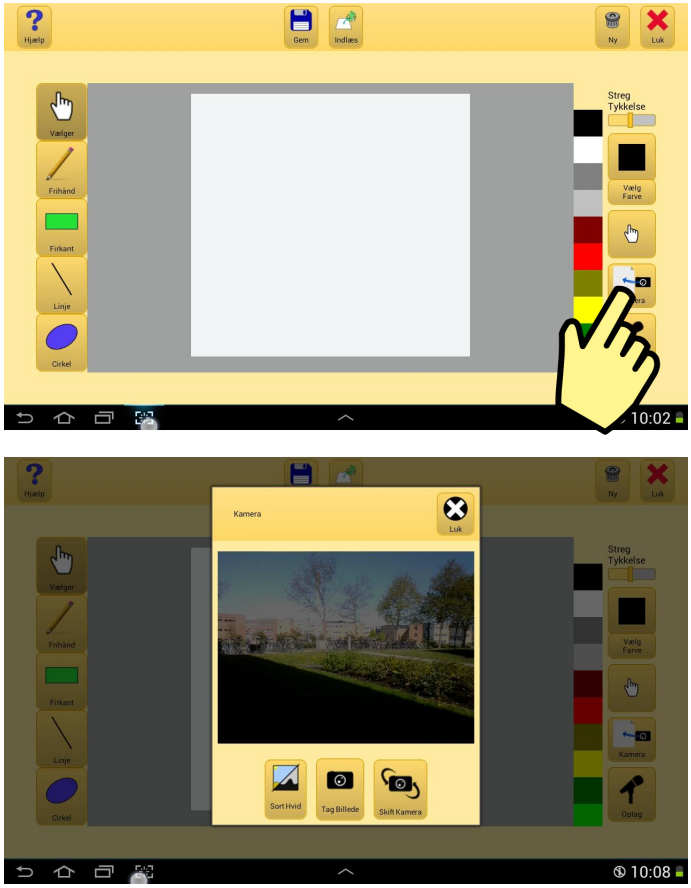


Figure 6.1: Help picture for opening camera dialogue.

library created by the GUI group in the multi-project. The text colour on the radio buttons are white due to the library, but should have been black as they are not clearly visible.

Pictures should fill whole canvas

A request was made by the customers that once a picture was taken with the camera it should fill the drawing field. To do this we looked at the height and width of the drawing field and found the relation between the picture and the drawing field, and created a new bitmap where the bitmap was scaled with this relation in a proper width and height ratio. This scaling made the picture fit the drawing field in size. The placement of the picture was decided from the picture's centre, which was assigned to the drawing fields centre.

Resizing camera pictures

A request from the customer was that the pictures from the camera should scale instead of resizing its dimensions independently such that the picture would retain its quality. To constrain the resizing of pictures we found the ratio between its width and height and multiplied the drag vector with this

Figure 6.2: Example showing the button where citizen can be given access to a pictogram.

ratio.

The ratio was found by dividing the width with height, as the height of the pictures from the camera is always smaller than the width. The new dimension is then found by finding the length from the top-left corner of the picture to where the user clicks. The size of the projection on the x-axis of this vector is assigned to the width of the picture, and the height is assigned to the projection on the y-axis of the vector multiplied with the ratio.

Changing Preview Button

The current preview button always displayed a square with the fill and stroke colour matching the currently selected colours. This preview of a square could cause confusion regarding what type of drawing tool the user had selected.

To avoid this confusion, we changed the button's display to match the user's selected drawing tool. When the user selected the freehand or line tool, the button's display would change to a line with the fill colour as its colour. The rectangle tool is displayed as it did previously, and the oval tool displayed an oval figure with the same colour structure as in the rectangle. Selection tool is also a tool available for the user, and when chosen, the *previewButton* would display the selection tool icon.

The button also had the feature of swapping colours as mentioned in Section 3.3, but this feature is unavailable when the freehand, line, and selection tool are chosen, as it is not intuitive to swap to a colour that is not visible.

6.3 Usability Test

To ensure that the customers are able to use the application and that it is easy to understand, a usability test is performed. The reason why the application

Figure 6.3: The dialogue where citizen are chosen for access.

has to be easy to understand is that the customers are not used to android tablets. Moreover, the customers are receiving a tablet with the application installed and are therefore not told how to use the application beforehand.

To make a usability test, several tasks are constructed which makes the testees try all parts of the application. The testees then have to try to complete these tasks without help. If they are unable to complete the tasks without help, they can ask the tester for help. Problems are then described and put into categories depending on how the testees complete the tasks. The problems are categorised into three categories which are cosmetic, serious, and critical. Problems that are categorised as cosmetic are defined as tasks that are slowing the testees down by a few seconds. When a problem is categorised as serious, the testees are unable to solve the problem without any help or are stuck for several seconds. The worst type of problem is critical, this problem occurs when the testees are unable to solve the problem and the person sitting next to the testees has to show them how to perform the task.

The usability test is performed in the end of this sprint and the issues found are therefore not resolved in this sprint. However, suggested solutions for the main problems are described in Appendix B.

P1 - Change Camera Picture to Black and White

A problem occurred when the testee had to take a picture of their chair in black and white. The problem was that one of the teestees took the picture and thought that they could change the colour of the picture afterwards. Another testee did not read the tasks well enough and therefore forgot to pick black and white.

The reason why this is a cosmetic problem is that the first testee had some problems finding out how to change the colour of the picture. However, the

Problems	Description	Category
P1	Change camera picture to black and white	Cosmetic
P2	Clear canvas	Cosmetic
P3	Drawing entities	Serious
P4	Move entities behind another entity	Serious
P5	Swap colours	Critical
P6	Add tag	Cosmetic
P7	Delete entity	Serious
P8	Colour picker	Cosmetic
P9	Play sound	Cosmetic

Table 6.1: Discovered usability problems.

second testee found the colour changer right when the testee was told that it should be in black and white.

P2 - Clear Canvas

When the testees had to clear the canvas, after drawing, they had problems finding the button to do so. However, after looking for a few seconds the testees were able to locate the button, which is why the problem is cosmetic.

P3 - Drawing Entities

Drawing entities is done by swiping the screen. The start position of the swipe is then the first corner and the end of the swipe is the opposite corner, the entity is then drawn using these corners. What the testees did when trying to draw an entity was that they pressed the screen and believed that a circle would be created.

This problem is a serious problem since the testees used several seconds figuring how to draw an entity. However, none of the testees needed help to be able to draw an entity, and for that reason it is not deemed a critical problem.

P4 - Move Entities Behind Another Entity

Moving an entity behind another entity was a problem for one of the testees, since the testee was unable to understand where the button to do this was. When told which button pushed an entity back, the testee felt that the icon was not understandable. Furthermore, the testee thought that the button would push an entity in front of another instead of behind an entity. Therefore, the testee needed help figuring out how to use this functionality. Since the testee had so many problems with this task it is deemed a serious problem.

P5 - Swap Colours

The button to swap colours was not understandable by the testees. Furthermore, none of the testees figured out how the button worked even after getting told how it works. For that reason this problem is a critical problem and should be fixed in the next sprint of *Piktotegner*.

P6 - Add Tag

The problem occurred when one of the testees should add a tag to the tag list. The testee did, however, only need a little help to figure out how to add the

tag to the list. As just one of the testees were not able to add it to the list and had few problems doing this, the problem is deemed to be a cosmetic problem.

P7 - Delete Entity

When deleting an entity, the user have to select the selection tool and then select the entity that has to be removed. However, none of the testees understood that they had to select the selection tool to do this. Both of the testees thought that there would be an eraser to remove entities. Furthermore, both of the testees tried to double press the entity because they thought that would select the entity. However, after few seconds and a bit of help they were both able to see that selecting an entity would allow them to delete it. Therefore the problem is deemed to be a serious problem.

P8 - Colour Picker

One of the testees had a problem with the colour picker. The problem was that the testee thought that pressing the button that picks the last picked colour would open the colour picker. However, after several seconds the testee figured out that pressing the colour picker would lead them to picking the colour. Since only one of the testees had problems, it is deemed to be a cosmetic problem.

P9 - Play Sound

To play the sound of a pictogram, the user has to go into the recording dialogue, which happens by pressing the record button. The testees did not find this understandable as the record button should be for recording sound only. However, the testees, after a few seconds, found that the record dialogue would be able to play the sound of a pictogram, which is why the problem is a cosmetic problem.

6.4 Sprint Closure

The issues mentioned in the sprint backlog have been solved but new issues have also been discovered through the usability test as mentioned in Section 6.3. Replacement of pictograms in the database was not implemented in the sprint, but is recommended to consider for future groups further developing the application.

We also refactored the code such that it is easier to understand for future developers. The refactored code was also shown to another development group as part of a code review and gave us feedback. The feedback was taken into consideration and changes were performed when they were deemed reasonable. The resulting code should now be easier for the next development group to expand upon if further development of this application is performed.

After the presentation of the application during the sprint end meeting, new issues were mentioned which are added to the product backlog. An example of an issue was that the audience were slightly confused of what the preview button was actually doing, so a rethinking of this button should be done. The issues found are further explained Appendix B.

This concludes the development phase of the project, all remaining issues will be written in Appendix B and added to the product backlog for easy reference for future developers.

7. Collaboration - Audio

Sound for pictograms was deemed important by the customers, and as of such, *Piktotegner* and *Piktooplæser* chose to cooperate in developing libraries for this. The libraries is mainly split into two parts, a media player and a text to speech library. The cooperation for developing the libraries was done by way of pair programming, with one person from each group per library. As of such, a mutual agreement was found for how the libraries should be developed. A description of these as well as motivation and a description of their implementation will be given hereafter.

7.1 PictoMediaPlayer

Associating sound to pictograms was stated as of great importance from the customers. For that reason a library is developed that provides an easy to use sound playing service.

Both *Piktotegner* and *Piktooplæser* need the sound playing feature. *Piktotegner* needs it to get a preview of the sound recorded, to be saved when creating a pictogram. *Piktooplæser* needs the feature as the key feature of *Piktooplæser* is to get the sound of the pictograms played sequentially.

When initialising the library it needs a context and can additionally take a path for the sound file, a pictogram, or a bytearray. The context is needed for the PictoMediaPlayer to know where to play the sound. In List 7.1 we see a constructor for the PictoMediaPlayer.

```
1 public PictoMediaPlayer (Context activity, String path)
2 {
3     this.activity = activity;
4     assignMediaPlayer();
5     setDataSource(path);
6 }
```

Listing 7.1: Constructor for PictoMediaPlayer.

When assigning a datasource to the PictoMediaPlayer the *setDataSource* method, seen in List 7.2, is called. The method releases an old MediaPlayer and assigns a new one if sound has already been attached to the MediaPlayer. Then the MediaPlayer gets a file descriptor assigned as source.

```
1 public void setDataSource(String path)
2 {
3     if(hasSound)
4     {
5         mediaPlayer.release();
6         assignMediaPlayer();
```

```
7       }
8
9       try
10      {
11          FileInputStream fileInputStream = new
                FileInputStream(path);
12
13          mediaPlayer.setDataSource(fileInputStream.getFD());
14          hasSound = true;
15      }
16      catch (IOException e)
17      {
18          e.getStackTrace();
19      }
20  }
```

Listing 7.2: SetDataSource method of PictoMediaPlayer.

The *OnCompletionListener*, seen in List 7.3, is necessary for external sources to detect when a sound is finished playing. The listener allows external sources to respond to a sound finished playing, e.g. a button icon change.

```
1   private final MediaPlayer.OnCompletionListener
        onCompletionListener = new
        MediaPlayer.OnCompletionListener()
2   {
3       @Override
4       public void onCompletion(MediaPlayer mp)
5       {
6           isPlaying = false;
7           if(customListener != null)
8           {
9               customListener.soundDonePlaying();
10          }
11      }
12  };
13
14  public interface CompleteListener
15  {
16      public void soundDonePlaying();
17  }
```

Listing 7.3: onCompletionListener method of PictoMediaPlayer.

For *Piktooplæser* to play the sound of multiple pictograms in succession we made the *playListOfPictograms* method as seen in List 7.4. We first store the old listener in a temporary variable and stop playing sounds. Then we overwrite the method which is run after a sound is complete, so that it plays the sound of the next pictogram in the list. When all pictograms in the list have been read the old listener is restored.

```
1   public void playListOfPictograms(ArrayList<Pictogram>
        pictogramList)
2   {
3       pictogramListIndex = 0;
```

```
4       this.pictogramList = pictogramList;
5       TempCompleteListener = customListener;
6
7       if(isPlaying)
8       {
9           stopSound();
10      }
11      this.setCustomListener(this);
12      this.setDataSource(pictogramList.get(pictogramListIndex));
13      this.playSound();
14  }
15
16  @Override
17  public void soundDonePlaying()
18  {
19      pictogramListIndex ++;
20      if (pictogramListIndex < pictogramList.size() &&
            pictogramList.get(pictogramListIndex) != null)
21      {
22          setDataSource(pictogramList.get(pictogramListIndex));
23          playSound();
24      }
25      else
26      {
27          this.setCustomListener(TempCompleteListener);
28          pictogramListIndex = 0;
29      }
30  }
```

Listing 7.4: playListOfPictograms method of PictoMediaPlayer.

To illustrate the ease of use of the PictoMediaPlayer we created the example seen in List 7.5. We here see how a PictoMediaPlayer is used to play a sound and writing "Sound is finished playing" when the sound is finished playing.

```
1   public class example implements CompleteListener
2   {
3       private void examplemethod(Context context, String path)
4       {
5           PictoMediaPlayer pictoMediaPlayer = new
                PictoMediaPlayer(context, path);
6
7           pictoMediaPlayer.setCustomListener(this);
8           pictoMediaPlayer.playSound();
9       }
10
11      @Override
12      public void soundDonePlaying()
13      {
14          System.out.print("Sound is finished playing");
15      }
16  }
```

Listing 7.5: Example of PictoMediaPlayer.

7.2 Implementing Text-To-Speech

The motivation for creating a Text-To-Speech (TTS) functionality is that the customers should be able to play the sound of all pictograms, even if a pictogram does not have a sound associated with it. The reason why the customers need to play a sound of a pictogram is to teach the autistic people to associate a pictogram with a sound. The *Piktooplæser* application should use the TTS functionality whenever a pictogram without a sound is added to the sequence to be read. Furthermore, *Piktooplæser* should add the newly created TTS sound, so the pictogram have the specific sound for later usage. The newly generated TTS sound should therefore be stored in the local database, which will synchronize the updated pictograms with the other users, such that the sound is only generated once. For *Piktotegner*, with pictograms created by the customer where no sound is recorded, a sound should be created by reading the inline text of the pictogram. This should of course also be stored to the database for later usage.

Available Text-To-Speech Tools

To enable the TTS function, the different available TTS tools should be compared, so the best suitable tool for this project can be chosen. There are a lot of different TTS tools, but there are some requirements:

- The tool must be free to use
- The tool must be able to speak the given text in danish
- The tool must be usable with an android application

From these requirements, the build-in Android TTS can be excluded, because it does not support the danish language. Tools which fulfil these requirements have been found and the ones we chose to compare are listed below:

- Voice RSS
- eSpeak
- JeSpeak
- Google Translate TTS

Voice RSS

Voice RSS [4] is an online TTS tool with an API. Voice RSS have a Danish voice for their TTS, but have a limitation of usage. It is only the first 350 TTS requests per day which are free, thereafter there must be paid a fee for more requests. The sound of the TTS is relatively good, and the Danish voice is understandable. To make a request to Voice RSS, the device must be connected to the internet.

eSpeak

eSpeak is a free open source TTS build for Linux and Windows with many supported languages, including Danish [5]. The advantage of eSpeak is that it is free to use, and do not need internet access to work. The disadvantage is that it is not made directly for use with Android.

JeSpeak

JeSpeak is a Java Library built from eSpeak [6]. The advantage for this TTS tool is that it is a Java Library which makes it compatible with the Java written *GIRAF* project. The downside is that it is not very well documented and requires not only Android SDK, but also Android NDK.

Google Translate TTS

Google Translate TTS is a possibility to use their API to make web based request and receive a sound file. It is very similar to Voice RSS, but has higher limit on the number of requests. It has a wide range of languages, including a quite good Danish TTS.

Comparison and Selection of TTS Tool

As seen, there are four examined solutions for enabling TTS in the *GIRAF* project. A comparison of the different TTS tools can be seen in Table 7.1.

TTS Tool	Advantage	Disadvantag	Requires Internet Access	Price
Voice RSS	Easy well documented API, understandable voice	Requires internet access, limit in requests	Yes	Free up to 350 requests per day, hereafter a fee starting from 5$
eSpeak	Free, does not require internet access	Not built for use with Android	No	Free, Open Source
JeSpeak	Free, made for Java	Not made directly for use with Android.	No	Free
Google Translate TTS	Free, supports many languages, easy to implement	Requires internet access	Yes	Free

Table 7.1: Comparison table for different TTS tools.

Due to this comparison, Google Translate TTS has been chosen. It is very easy to implement, compared to e.g. JeSpeak, free to use, and it has a good Danish TTS.

Handling TTS

As seen in Section 7.2, Google Translate TTS is chosen as the TTS tool for this project. As it is a web-based tool, it must be considered how it should be used, and how to handle the situation where there is no internet connection.

There is also the possibility that the creator of the pictogram has recorded his own reading of the pictogram. In this case the TTS should not be used. From these statements, a flowchart of the behavior of playing a pictograms sound can be seen on Figure 7.1.

With the behavior in place, the TTS and soundplaying method can now be implemented. Note that once the sound has been generated, it is stored to the pictogram in the database, so it should only be generated once. The updated pictogram is synchronized with the remote database, so every other tablet also gets the sound file.

Implementation

The implementation of the TTS functionality is created by opening an url connection to *Google Translate*'s TTS method. When the url connection is established, a stream starts to return bytes. These bytes are then saved in an array so they can be translated into sound afterwards. How to get the bytes and save them can be seen in List 7.6.

```
1  private void DownloadFile() {
2      try{
3          URL url = new URL(soundURL);
4
5          URLConnection urlConnection = url.openConnection();
6          InputStream inputStream =
               urlConnection.getInputStream();
7          BufferedInputStream bis = new
               BufferedInputStream(inputStream);
8          ByteArrayBuffer byteArrayBuffer = new
               ByteArrayBuffer(50);
9          int current = 0;
10         while ((current = bis.read()) != -1)
11             byteArrayBuffer.append((byte) current);
12
13         SoundData = byteArrayBuffer.toByteArray();
14     }[...]
15 }
```

Listing 7.6: Download Sound event

Before calling this method, it must be ensured that the device has a valid internet connection, because if there is no internet connection you are not able to connect to *Google Translate*. To check if there is an internet connection, a connection manager checks if it can get network activity. If the connection manager does not find an internet connection the *DownloadFile* method cannot be called.

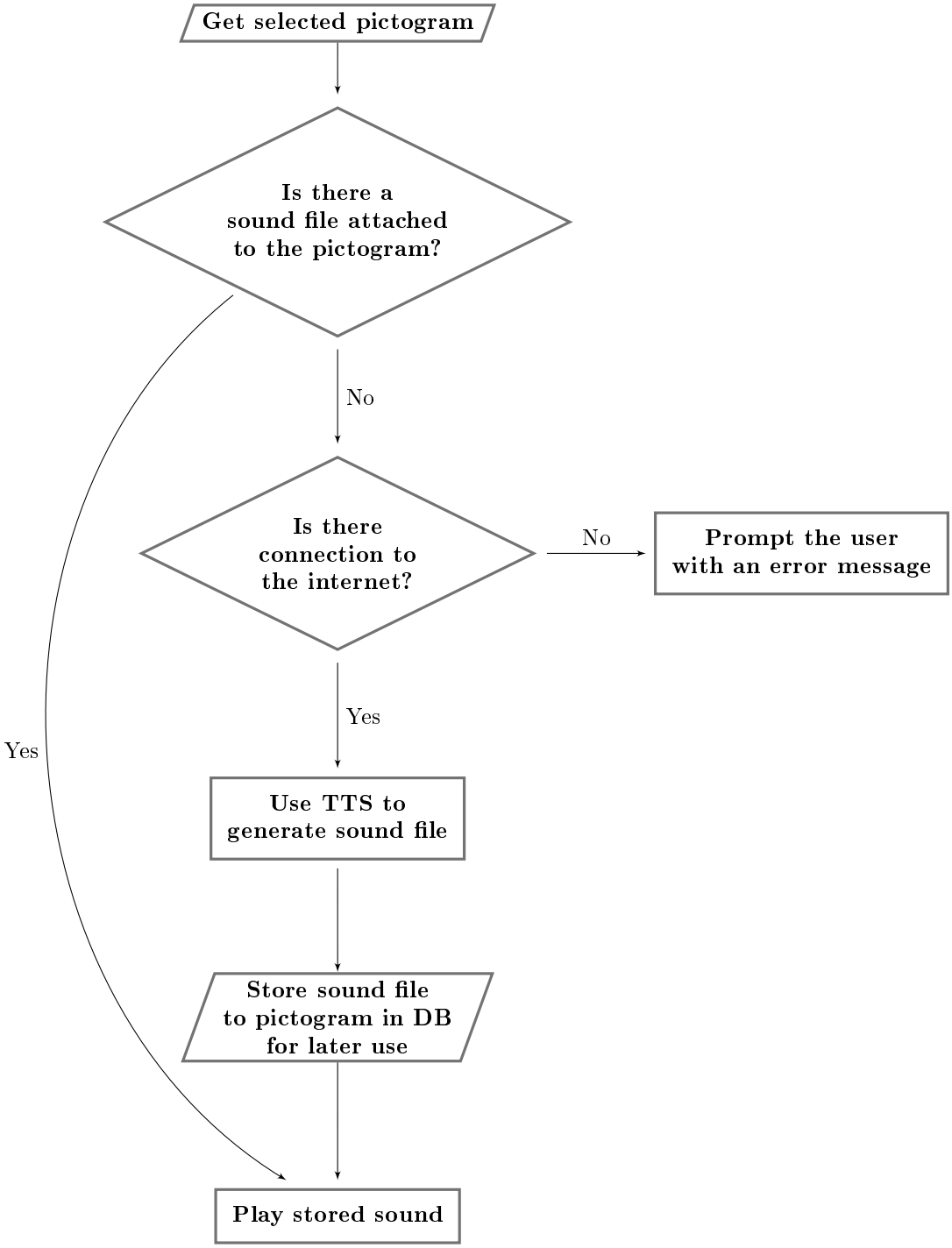


Figure 7.1: Flowchart for the use of sound on pictograms

8. Common Activity - Code Review Between sw606f14 & sw608f14

This chapter was written in collaboration between the groups sw606f14 and sw608f14 and is in both reports.

A code review cooperation was performed between the *Webadmin* (sw606f14) and *Piktotegner* (sw608f14) groups. It was done in order to secure a better code quality, to find bugs, and as a learning experience. By code quality meaning e.g. suitable comments, reducing redundancy, and following the code standards, camel casing for example.

The code review was constructed in an informal way, as it made us able to cover more code in the same time span as a formal code review. The formal code review may find more serious bugs, however, it would only be able to cover a limited amount of code with the time at hand, thus the prioritisation of an informal code review over a formal version.

Approach

In order to secure that the two groups understood each others applications' architecture, a presentation was performed before the code review. For *Webadmin*, it was told how it is based on the model-view-controller pattern. For *Piktotegner*, a class diagram showing how entities are structured as well as their respective handlers were presented. This gave insight to how the code, to be reviewed, was structured, in order to avoid confusion that could help us get to review more code and to give a better review of this code.

The review was done in a pair of two, such that two groups from *Webadmin* reviewed *Piktotegner* and vice versa. The presentation was the first time the groups got presented for each others code, and as of such we sat in the same room when reviewing. This ensured that if any questions arose, regarding the architecture and language specific operations, a group from the other application could then fast be consulted.

The corrections and suggestions found during code reviewing was documented with a quick title of the comment, a description of it, as well as the file to look at and the corresponding line number(s).

In hindsight, it would be good if the two groups had more experience in the languages of the code they reviewed, as *Webadmin* is primarily programmed in *PHP* and *Piktotegner* is programmed in *Java*. If the groups had more experience with these different languages, more bugs would likely be discovered.

However, despite this, it was found that the code reviewing was helpful and is a practice that should be continued in the future.

We find it a good practice to review code, however, if the comments given on ones code is not addressed, the code review does not give much value. Therefore both groups addressed the problems found during the code review session as the first activity the day after the code review.

Experience

The overall impression of the code review activity was positive, and is agreed upon by both groups. About 2500 lines per project group, including comments, was covered by the code review, which may have been too much for a four hours code review. It is believed that instead of having a singular lengthy code review activity, it would be better to perform several shorter ones, in order to not get tired and do inefficient code reviewing.

Webadmin was lacking comments to document their code, it was recommended that all functions was given a comment to concisely describe them. *Webadmin* had inconsistency in their naming conventions. *camelCase* and underscores for variable names, *PascalCase* and underscores for functions, and *camelCase*, *PascalCase*, underscores, and dashes for *css*, were used. Furthermore, redundant code was found in the *PHP* classes and it was suggested that the similar code could be moved to a new function or separate library. Lastly, suggestions for optimization of *SQL* queries was given.

Piktotegner had a lot of comments explaining both complex functions as well as the simple ones, some comments were deemed unnecessary or being superfluous. The variable names followed a coding convention but a few were found to not follow this convention or were found to have non describing names. Most properties had no constraints which might cause an error in the application if the properties were to be used improperly.

In addition to this, an opinion of both groups is that having people not familiar with the code try and read the code gives the advantage that it checks which parts of the code is hard to read and what code may need more documentation or refactoring.

9. Reflections

Working in a multi project setting has been a great learning experience. Starting the multi project, a deal of confusion existed as to how the project should be structured, as the project guidelines have been very different from previous semesters.

That being said, after the first couple of weeks getting used to the setting, the development was fun, however, it requires that you are willing to cooperate with other groups, and is necessary to get the whole software package to be consistent. Furthermore, it helped loosen up the project groups and provided a more social environment than previous semesters.

When questions arose, it was a matter of knocking on the door of the other group's group room, and the issue would be quickly resolved. In addition to this, SCRUM was utilised to manage the software development. It was done in a SCRUM of SCRUMs where a status meeting was held each week. We found this as a good technique, as when groups had problems, they could be found before too many resources were spent.

The product and sprint backlogs of most of the applications were organised in the tool called Redmine, and we found this a useful tool. This was useful because it made you able to monitor your own project as well as the whole multi project, to see if you are on track. In addition to this, it was a way to give a concise description of each issue and ensure that no issues were forgotten.

In order to estimate the issues, to ensure that the sprint backlogs were fitting, we used planning poker. We found planning poker a useful tool, and is highly recommended for future groups working with an agile methodology as it is a low resource estimation technique, which at the same time, for us, was fairly accurate.

Git was also a useful tool, as it helped ensure version control, and is recommended for future students. In addition to this, because we work in a big multi project, it is mandatory to have a tool such as GIT. The reason for this is that we depend on libraries developed by other applications, such as the database library and pictogram library. You do not have to adapt to changes in their developed libraries immediately, where you could risk getting interrupted all the time, to adapt to the changes made in those libraries. Instead you can wait to update your submodules for a fitting time in your schedule. However, it takes some time getting used to Git, in addition, submodules were good but some git structure was problematic. This occurred if we were dependent on a submodule which is then dependent on another submodule that we are also dependent on. The submodule would then use our submodule which may have another revision and would have to be adjusted thereto. For future students, it is recommended to use a development branch if you are making some unstable

changes to your code. Furthermore, before pushing you should ensure that your code is compilable.

In relation to this comes the automatic build service Jenkins. It was set to automatically build the different changes when new revisions were added to Git. People who may have pushed non-compilable code would then be notified, such that they could correct their mistakes. As good tool as Jenkins was, it got in use late in the development cycle, and if we were to do the multi project development again, we would recommend getting Jenkins in use as early as possible.

In order to also ensure that we had a role in the multi project, we had a person responsible for making icons for all the applications in the multi setting. This ensured that we also provided a role other than developing the *Piktotegner* application, in order to feel that we provided more to the whole project. Future students are also recommended to engage in such activities, being it icon creation, Git, Jenkins, or Redmine responsibility.

Customer communication was a positive experience, although it was difficult to establish contact. This is found to primarily be due to not contacting the customers in timely manner. As the other responsibility roles, we also had a person responsible for establishing contact with the customers. We found that usability tests was performed too late, as it was done in the two final days of sprint 4, thus the experiences from the usability tests could not be implemented in the application. In the future, it would be wise to perform the usability tests earlier, in order to correct on the issues found during the tests.

During development of the application, the features developed were based on customer communication, apart from bugs, as the customers wanted a stable application, so those improvements were made autonomously. There was, however, one feature requested by some of the customers that did not get implemented, the eraser tool. The eraser tool was attempted to be developed, but as the drawing is based on a vector graphics feel, deleting parts of an entity did not make much sense. When introducing a customer to the ability to removal of one entity at a time as an alternative, it was agreed upon to be a suitable replacement.

To summarize, the multi project was a great learning experience and is recommended to try for any student, as you get to work in a very different setting than what you are used to in previous semesters. Furthermore, we found that the development process went well, and the contact with the customers was appreciated as it is those the application is developed for.

Bibliography

[1] Johannes Lindhart Borresen Mark Faldborg Birgir Mar Eliasson, René Kjær Hornbjerg Andersen. Croc - en android app til pictogram konstruktion, may 2013. http://projekter.aau.dk/projekter/da/studentthesis/croc--en-android-app-til-pictogram-konstruktion(98b40067-3bd5-486a-b31c-554812464e83).html.

[2] Drazenko Banjak. Egebakken, 2014. drazenko.banjak1@skolekom.dk.

[3] Pernille Hansen Krogh. Mymo, 2014. p@mymo.dk.

[4] Voice RSS. Voice rss. http://voicerss.org/[Last seen: 2014/05/06].

[5] Jonathan Duddington. espeak text to speech. http://espeak.sourceforge.net/[Last seen: 2014/05/06].

[6] Pierre-David Belanger. Jespeak. http://jespeak.sourceforge.net/[Last seen: 2014/05/06].

A. Croc UI Screenshots

Figure A.1: UI screenshot of camera dialogue.

Figure A.2: UI screenshot of audio dialogue.

B. Product Backlog

Editing pictograms
It should be possible to edit pictograms, when it is the same author editing it. By editing we mean a change to either its name, sound, picture, access settings, or tags.

Crop camera pictures
A request from a customer was to crop pictures taken with the camera.

Undo button
Some colleagues suggested to add an undo button, as it was tedious to select the object first and then delete it every time they made a mistake.

Categorising pictograms
Newly created pictograms could have the possibility of being categorised when saved.

Change colour of drawn entities
It should be possible to change the colour of already drawn entities. This could be by either drag-and-dropping a colour onto an entity or selecting the entity and then choosing a colour.

Camera fragment buttons
During the usability test, the testees had trouble reaching the *capture* button when they tried to take a picture.

Template pictures
A suggestion was to add a setup assistant for template pictures, such that users could add some customisable figures into their pictogram. An example of this could be a stick figure, but with a customisable posture.

Memory issues
The application used a lot of high quality pictures which caused memory issues on certain tablets. These were pictures seen in the help dialogue and camera pictures.

Send the guardian id to *Pictosearch*
A guardian ID is received from *Launcher* when *Piktotegner* is launched. The guardian id is used to determine whether people should be able to save a pictogram. Other *GIRAF* applications may also need the guardian ID, such as *Pictosearch* which is launchable from *Piktotegner*, and should therefore be passed to them.

New icon for clearing the canvas
The current icon is a trash can with the text *Ny* which means new. The text does not hint at the functionality usually associated with such an icon, and should thus be changed to something more appropriate. A suggestion to such an icon would be the usual blank paper with a folded edge and a plus sign, as seen in most text editors and graphics painting application.

Close and exit icons
During the usability test, the testees stated that the citizens associate the current close and exit icons with cancelled activities. Therefore, these icons should be changed or removed.

Flatten button icon
The flatten button, which puts entities to the back of the canvas, was not clearly understood by the testees during the usability test and should therefore be redesigned.

Pinching
During the usability test, testees tried to resize an entity by pinching it on the touchscreen. This functionality is not currently supported, but since people are used to this gesture when using touch pad applications, this functionality should be considered.

Names in save dialogue
The text fields in the save dialogue were causing confusion both to the customers and our colleagues. The idea of these fields was that the inline text was the text written on the pictogram whereas the name was the actual name of the pictogram which the guardian used as reference. An example of this is two pictures, one with a red car and the other with a blue car, having the same inline text "Car", but for reference the guardian would like to know the colour of the car through the name, thus "Red car" and "Blue car".

Rethink preview button
Both in the usability tests and the sprint end meeting, the preview button was found unintuitive. The doubt was first the double functionality of the button, as it is a preview button and a swap colour button. A solution could be to scrap the preview button and instead have a clearer illustration of what tool is used, and at the same time have a new button with its functionality being the swapping of colours.

Alternative help dialogue
During the sprint 4 sprint end meeting it was proposed that the help pictures of the help dialogue should be made larger. As an alternative to this, it was proposed to provide short video clips to demonstrate how to use the application, and is the suggestion we recommend for future groups. If the help pictures solution is kept, clarifying text for these pictures would be beneficial.

Placement of timer
The application *Timer*, developed by another *GIRAF* group, provides a

timer to keep track on how long a citizen is provided for a given activity. The timer is placed in the top right corner of the tablet and should be taken into account, as at this point in time it overlays the exit button, or the *Timer* application should add a functionality to move this timer.

Reconsider selection of entities

During the usability tests, it was evident that the testees found the selection tool unintuitive. With training they may get used to the tool, however, they tried some other gestures to select an entity and is worth considering. Gestures used was double click on a given entity to select it or long clicking. Maybe these forms of gestures should replace the selection tool or be an alternative way to select entities.

C. Turnkey – Piktotegner

If you would like to work with an application that is essential to the creation of new pictograms, you should choose *Piktotegner*. *Piktotegner* is an android application which provides a pictogram creation environment for tablets. It is an application where the essential functionality to create pictograms is already established, but where usability improvement is needed.

When working on this application, we recommend performing several usability tests as well as stability testing, as there is some unclosed memory leaking issues. Unit testing is also recommended, as none of this has been performed.

As of such, a GUI redesign is recommended, based on your performed usability tests and knowledge from the DEB course.

If you find that the application becomes well polished for the users' needs you can embark in a whole new part of the pictogram creation environment. What is thought of here is a template environment, where you do not have to draw your own images, but can instead construct images from some predefined image structures, such as balls, stickmen, animals, machines and so forth, but where the user gets to choose the posture, colour theme, and accessories.

If you want to delve into how the elements of the image is drawn, rotated, and resized, be prepared for linear algebra, with rotation matrices, and euclidian geometry (taught in Advanced Algorithms).

C.1 Application structure

When launching the application, you start with the MainActivity.java. This is where all fragments and UI is launched from.

The drawing entities and their corresponding handlers are basically structured as seen in Figure C.1, and may need some restructuring.

Apart from this, the application is structured into the mainActivity, which contatins a drawFragment, record dialogue, camera dialogue, save dialogue and help dialogue. We recommend when you start to examine each class, as the code is well documented and in itself should serve as sufficient documentation.

Furthermore, we recommend Vogella's android tutorials, as they have been of great help throughout development.

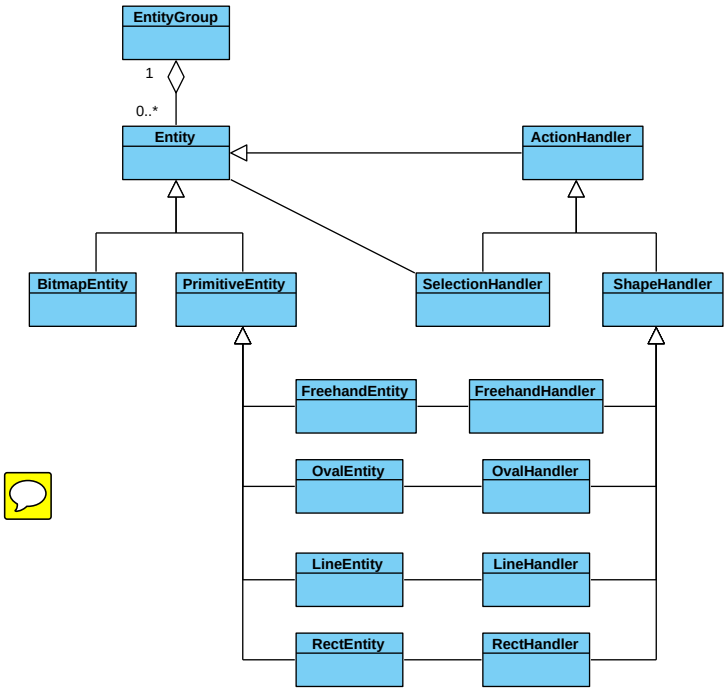


Figure C.1: Class Diagram with entities and handlers