Helping subsistence farmers in West Africa

Tom Jansen
Vrije Universiteit Amsterdam, 2015

Abstract. Subsistence farming or agriculture is a form of farming where farmers mainly focus on growing enough food to be self-sufficient. Especially in African countries, where people are very dependent of own-grown food, this type of farming is very common. Subsistence farming, however, in these countries has so much to gain and has so much potential. Improving the farming skills of the farmers could make significant contributions to the reduction of hunger. Unfortunately, farmers often haven't had enough agricultural education to optimally grow their own food. In this paper, an application is presented that will help subsistence farmers to improve their farming and thus their quality and quantity of food. The application helps the farmers to identify diseases of their crops and animals and will present them ways to manage the diseases and prevent them in the future. Furthermore, an attempt is made to provide some ways on how to continue from this point on.

Introduction

According to the Africa Human Development Report 2012 more than 75% of cereals and almost all crops in Africa come from domestic agriculture. In rural areas, 70% of the extremely poor population rely on agriculture as their crucial source of income [1]. These are only a couple of numbers that represent the importance of agriculture in Africa, let alone the importance of agriculture in the rural areas in Africa. The numbers above particularly focus on agriculture as a source of income. In this paper, however, we focus on agriculture as a means to grow food for own consumption: subsistence farming. Subsistence farming is a form of farming where farmers grow their crops and livestock to be self-sufficient. In this paper, emphasis will be placed upon the farmers in rural areas in West Africa. Subsistence farming, for most farmers, is their way to stay alive. They grow their own food because they are too poor to purchase it. In many rural areas in Africa, furthermore, there is a lack of good education. This means that lots of farmers never actually learned certain farming skills. Most skills are taught by parents, relatives or people from the same village or area. Many subsistence farmers, therefore, have a lot to gain when they are enabled to improve their farming skills.

In this paper, an application is presented that will help to solve the problem presented above. The application will help the farmers to identify, manage and prevent problems and diseases that commonly occur. The application revolves around a database that contains helpful information about several diseases. Identification possibilities are shown to the farmers, helping them to define the disease their crops or animal have, and when the farmers have clicked on the identification a new window pops up providing them with some information on how to manage and prevent the disease. The idea alone sounds very simple, but that leaves us with another issue: how to make this accessible for farmers in rural areas. The application runs on Sugar, a framework that runs on the so called XO Laptop. This laptop is commonly used in Africa, but mostly for educational purposes. Providing not merely children, however, with this laptop but for example also farmers in rural areas could make a big difference for them. The application (or activity as called for Sugar) works very simple and gives concrete information about the diseases. To properly present the activity, first we will talk about the concept. After the concept is clear, the process of development and the evaluation of the activity are handed out. Finally, there is some room for a discussion and the conclusion of the paper.

Concept

Before really going into detail about the development process, first we elaborate on the goal and the idea behind that goal. The desirable goal is to deliver subsistence farmers a way to improve their farming skills and to increase their harvest. To reach this goal, an activity is developed that will help the farmers to recognize, manage and prevent diseases of their livestock and crops. The activity to be developed had to be simple but most of all adequate and unambiguous. The simplicity has to do with how easy it is to use the activity. Since the activity focusses on subsistence farmers in West-Africa, where not everybody has had proper education and where illiteracy is a big problem [2], everything has to be simple and clear. In order to make it accessible for as much people as possible, the activity is most of all icon and pictures based. Because pictures sometimes can be ambiguous, every picture does have a supporting text. Finding a disease or pest should be possible within only a couple of clicks. In order to get a clear idea of how everything had to look like, an activity description is added:

Activity Description:

Goal of the app:
To help farmers recognize/prevent/manage diseases and pests their crops and livestock are struggling with.

Functional requirements:
- Help farmers recognize, manage and prevent diseases and pests

Nonfunctional requirements:
- Easy to use
- Simple and clear interface
- As little text as possible (due to illiteracy)

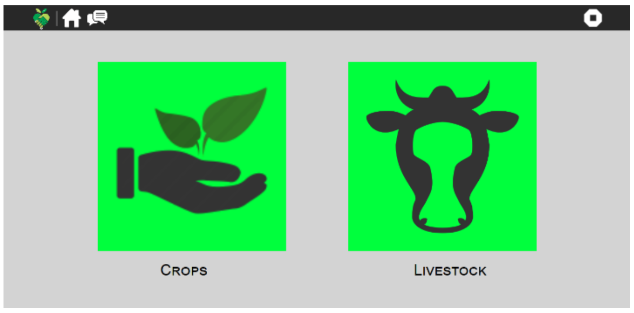
Use case:
1) Activity starts, the start menu is shown
2) Farmer clicks at the tile that represents crops
3) Farmer picks the kind of crop he has troubles with
4) A list with all kinds of diseases pops up, from which he can pick his
5) Show the solution and give the farmer some help to prevent them in the future

The activity description presents a brief overview of what the activity is supposed to do. The whole activity had to work on the XO Laptop, a laptop that is commonly used in those parts of Africa. The XO Laptop runs on the Sugar environment, so the activity obviously had to be Sugar based. As shown in the activity use case in the activity description, the idea was to create three interfaces or menus. First a menu with two tiles: crops and livestock. Then two menus that contain all the crops and livestock in the database. Clicking on a particular crop or animal then presents a list of all the known diseases of that particular crop or animal.

Development

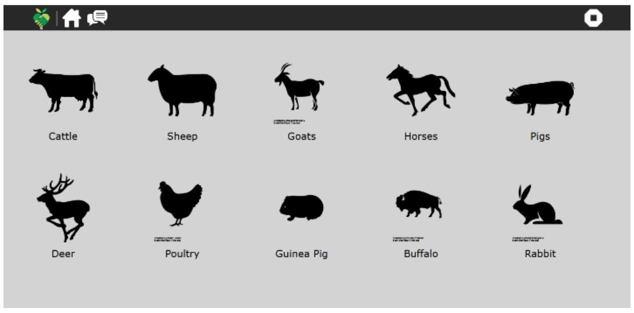
The Sugar environment is mainly used for XO Laptops and is therefore not a programming environment with a lot of information and tips on the internet. The Sugar activities that run on the environment can be coded both in Python and JavaScript. Where Python is the more recommended way to do it, JavaScript is the easier one because programming and testing can be done on any ordinary computer. If the activity is coded in Python, one would need a Sugar environment to actually test the environment (which is possible on Linux based computers). For coding in JavaScript there is a nice environment that made Sugar more accessible: Sugarizer [3]. This online Sugar platform can be downloaded on many devices and coded activities can then be tested. The Subsistence Farming activity this paper is about, is also created using Sugarizer and JavaScript.

Now that the concept and environment are clear, it is time to get into detail about the actual development. As stated in the activity description, there were three interfaces or menus to create: the main menu, a crop and livestock menu and an interface representing all the diseases per crop or animal. The whole Sugar interface actually is a canvas where you render all the pages into. The toolbar of the interface, however, is a part of the page that always stays the same. That is also the reason the rest of the page is a canvas where you render content into. Our toolbar only consists of four buttons. The two outermost buttons, the activity and stop button, are standard in every Sugar activity. The left most button is the activity button, where one can see the activity icon with a description when clicked. The right most button is the stop button and exits the activity. The other two buttons right from the activity button are a home and forum button. The home button leads one to the main menu and the forum button is not active yet, more on this in the discussion section. The main menu only consists of two buttons, a crop and livestock button, as shown on the right. Many Sugar activities that are programmed with JavaScript use the Enyo framework [4]. Enyo is a JavaScript framework for building (mostly HTML5) apps and is often used for Sugar activities. In Enyo, every page or part of a page is defined as a 'kind'. Every kind contains a couple of properties that define the page, most important are the components of the kind. In the components you specify what you want to render into the page. For example the components that represent the main menu are as follows:



```
{components: [
    {name: "crops", kind: "Image", src: "icons/crops.svg", classes: "cropsbutton", ontap: "showCrops"},
    {name: "livestock", kind: "Image", src: "icons/livestock.svg", classes: "livestockbutton", ontap: "showLivestock"},
    {content: "Crops", classes: "crops_text"},
    {content: "Livestock", classes: "livestock_text"},
```

Here, the two images are given a source, class (where you define the style) and an “ontap” function that is defined later and where one can define what happens if the button is clicked. There is one more thing to take into account here that has not been mentioned yet. Since the activity is meant for subsistence farmers in West-Africa, multi-linguicity has to be supported. In this paper, all the code and images are in English but it cannot be assumed that all the farmers understand English. Fortunately, Sugar is able to translate activities which makes it possible to support multi-linguicity.



The second menu or interface is ‘coded-wise’ the same as the main menu. All the buttons are defined as a component and have a supporting text, that is a component on its own as well. Clicking on the icon of an animal will then lead to the page of the particular animal. The icons that are used and can be seen in the screenshot come from The Noun Project [5] and are licensed under the Creative Commons or Open Domain license.

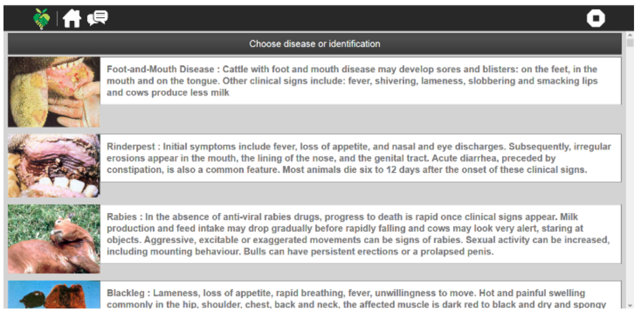
In the concept, it is already mentioned that the crucial part of the activity revolves around a database containing all the information about the diseases. This comes forward in the third and final interface where all the diseases per crop or animal are presented. The database we are talking about is a large JSON file where every variable looks like the following line of code:

```
var cattle = [
    { livestock:"Cattle", disease:"Foot-and-Mouth Disease", identification:"Cattle with foot and mouth disease may develop sores and blisters: on the feet, in the mouth and on the tongue. Other clinical signs include: fever, shivering, lameness, slobbering and smacking lips and cows produce less milk", management:"You can help prevent the disease by being familiar with the clinical signs of foot and mouth disease so you can notify APHA immediately if you suspect it and practising strict biosecurity on your premises.", other:"", src: "icons/livestockdiseases/cattle_fmd.svg"
},
```

Every crop and animal have a variable in the database file, and every variable consists of a list of diseases. Every disease is specified in a number of name/value pairs:

- Livestock (or crop): where the name of the animal or crop is given
- Disease: the name of the disease
- Identification: features of the diseases which make it recognizable
- Management: how to manage and possibly prevent the disease
- Other: possible extra information
- Src: the source of the supporting picture of the disease

The third interface renders the suitable information from the database (disease, identification and src) into a list of text and pictures that is shown below.



Foot-and-Mouth Disease : Cattle with foot and mouth disease may develop sores and blisters: on the feet, in the mouth and on the tongue. Other clinical signs include: fever, shivering, lameness, slobbering and smacking lips and cows produce less milk

Rinderpest : Initial symptoms include fever, loss of appetite, and nasal and eye discharges. Subsequently, irregular erosions appear in the mouth, the lining of the nose, and the genital tract. Acute diarrhea, preceded by constipation, is also a common feature. Most animals die six to 12 days after the onset of these clinical signs.

Rabies : In the absence of anti-viral rabies drugs, progress to death is rapid once clinical signs appear. Milk production and feed intake may drop gradually before rapidly falling and cows may look very alert, staring at objects. Aggressive, excitable or exaggerated movements can be signs of rabies. Sexual activity can be increased, including mounting behaviour. Bulls can have persistent erections or a prolapsed penis.

Blackleg : Lameness, loss of appetite, rapid breathing, fever, unwillingness to move. Hot and painful swelling commonly in the hip, shoulder, chest, back and neck, the affected muscle is dark red to black and dry and spongy

To make this possible, first an empty list is created with an undefined number of list items. The Enyo framework has some standard kinds and classes that represent lists as well, these are easy to use and save a lot of work. The content of the list items and the source images are defined when an icon of a crop or animal is clicked. When this page is being loaded, the database is being searched for diseases of the selected crop or animal. The number of list items is defined by the number of diseases it finds. After the number of list items is defined, the content and source of the images are set. When a list item is clicked, a pop-up screen is shown with the management (and prevention if available) of the chosen disease. The farmer now knows what do and hopefully his crops or animals can overcome the disease.

Evaluation

Looking back at the requirements of the activity that were set in the activity description, it can be said that both the functional and nonfunctional are fulfilled. The activity helps the farmers to recognize, prevent and manage diseases by accessing a database with the right information for the farmer. This information can be accessed within only a couple of clicks. When the disease is found within the least amount of possible clicks, the right information can be accessed by the farmer within only three clicks. The activity relies on images as much as possible, and has a supporting text with every image to make the purpose of the image unambiguous. All the interfaces or menus consist of clear tiles, that make them as convenient as possible.

Discussion

Although the activity is finished and meets the criteria that were set in the activity description, there are some points for discussion. The activity can be improved to make it even more useful for the farmers. A functional requirement that is not in the activity description, but a requirement that has always been kept in mind is the fact that farmers can help each other. There is only one way to let them help each other; by making it possible for the farmers to communicate. If the activity would contain a forum (the already existing button in the toolbar) where farmers can reach out to each other, the activity would be even more functional then it already is. This way, farmers can help each other with diseases or problems that are not yet in the database. Next step would be to make it possible for the farmers to add information to the database. This way the activity can expand without any help from the outside. The information would become more and more adequate and the database would expand as well. To make the database accessible for multiple users, it has be stored and shared somehow. Internet would be the simplest solution, but internet is exactly one of those things most subsistence farmers do not have access to. There is, however, a solution: ERS. ERS works as shown in Figure 1 and has already been implemented in several cases with the XO-Laptop in West Africa [6]. It makes it possible to share data even within small local networks. ERS works with three types of components: the contributors, bridges and a global server. Looking at this with respect to the activity described in this paper, all of the contributors (the farmers) would have a database which can be sent to and received from the bridges. These bridges make it possible to connect several isolated networks to each other and the global server. This would make it possible for farmers to access, mutate and expand the database

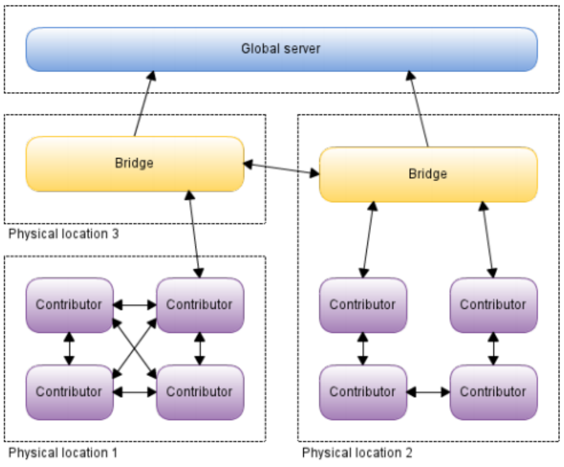


Figure 1: The functioning of ERS

Conclusion

In this paper, an activity is presented that can contribute to subsistence farming in (mostly) rural areas in Africa. Subsistence farming is especially in rural areas in Africa very common. Despite the fact that it is very commonly carried out, the way it is carried out can be improved a lot. Improving subsistence farming can contribute to the living standard of the farmers; their food stocks can grow and they might even be able to sell some of their stocks. The subsistence farming activity runs on the Sugar environment for XO-Laptops and will help the farmers to recognize, manage and prevent diseases. In addition to the presented activity, some recommendations are put forward. Implementing a forum to make communication possible and using ERS to store and share the database would improve the activity as it is.

References

[1] United Nations Development Programme, *Africa Human Development Report*, United Nations Publications, 2012, 65-66
[2] UNESCO, *Regional fact sheet – Sub-Saharan Africa*, 2010, 1-2
[3] http://sugarizer.org/
[4] http://enyojs.com/
[5] http://thenounproject.com/
[6] M. Charlaganov, P. Cudré-mauroux, C. Guéret, M. Grund, & T. Macicas, *The Entity Registry System: Implementing 5-Star Linked Data Without the Web*, 2013, 4-6