TicTacToe on Arduino

Lukas Luschin
Hochschule Konstanz
Technik, Wirtschaft und Gestaltung
Germany, Konstanz
lukas.luschin@htwg-konstanz.de

Julian Altmeyer
Hochschule Konstanz
Technik, Wirtschaft und Gestaltung
Germany, Konstanz
julian.altmeyer@htwg-konstanz.de

TicTacToe is one of the most well-known computer games in the world. Nowadays it is also used as a basic programming exercise for learning how such a game is made up. It was implemented on different devices many times. This article is about the implementation of that game on an Arduino Uno Microcontroller board in combination with a 2.8" TFT touch display. The main aim is to develop a consistent architecture, where the game logic is clearly differentiated from the system. A scalable graphic library, to easily add and create new content. And finally, a user-friendly graphical interface.

I. INTRODUCTION (*HEADING 1*)

This project is about of recreating the game called "TicTacToe". The game is a paper and pencil game, where two competing persons set an assigned sign alternating in a three by three grid. The target is to set this sign in 3 adjacent fields. "TicTacToe" is referable to a game called "noughts and crosses" appeared in 1864 in Britain and it was also one of the first games for the computer. Within the scope of this project, "TicTacToe" was implemented on an Arduino Uno Microcontroller board and is controlled with a mounted touch display. Therefore, a specific software library was developed, to easily create and administrate the graphical user interface and to manage all user interactions. A major priority, in addition to the library and the controller, are the usability and the extensibility of the system.

The idea of the project was given by our course as an interesting and challenging possibility to deepen the knowledge in a ubiquitous system. We also chose that topic because of its potential to work with a microcontroller, learn the functionality of a capacitive touch display, establish a communication between these elements and to develop a fully operational system in accordance with our priorities.

II. STATE OF THE ART

First of all, it is required to look for other solutions out there. By doing the research it was noticed that there is a big amount of projects which solved our problem already. Most of the projects used a touch display or blinking LEDs to represent the game. But some of them implemented the game in a very creative way, e.g. a robotic arm playing TicTacToe. The following section will describe three of these existing projects in more detail.

One TicTacToe project decided to use a 3x3 LED grid, for example [1]. Each glowing LED represents the special TicTacToe mark players can set. To distinguish between the two players, each LED can adopt a different color. There are hardware buttons placed next to the LEDs to turn the LED on and set the mark. In this version, it is not possible to play the game against a human player, because the creator implemented an Arduino controlled artificial intelligence.

The next project which shall be shown used a robotic arm playing the game [2]. The TicTacToe grid is drawn on a paper. There are several pieces which can be placed on the paper/board. The pieces represent the TicTacToe mark of the players. A web cam captures the board and sends the image to a computer. The image is analyzed by a visual basic program. A special algorithm decides where the piece should be placed next. After that is done, the Arduino will move the robotic arm to the selected position by the program.

The third project which fits the most to this research is called "Tic Tac Touch" [3]. The creator used only an Arduino with a TFT touch screen provided by Adafruit. The 2.8" touch shield is directly put on top of the Arduino. The game has score tracking and a basic game logic. But there is no menu or artificial intelligence.

In summary, you can say that there are many working TicTacToe games out there, which used an Arduino for implementation. Some of them programmed an artificial intelligence or created a robot which can play the game. But most of the projects out there have one thing in common: They don't have a proper software architecture. Especially the projects using a touch display. A well-organized structure of the code is missing. This project is focused on creating a good software architecture. The next chapter shows possible solutions for this problem.

III. METHODOLOGY

To get started with the project there is the need for an Arduino Uno board and a touch screen. It should be possible to control the screen with the library provided by the touch

display manufacturer. Functions for drawing on the screen and receiving touch events from it are necessary.

As mentioned earlier one of the priorities of this research is to create a well-structured software architecture for the game TicTacToe on the Arduino board. One possible solution to get this is for example to create a model-view-controller (MVC) architecture. In this case the data (model), the game logic (controller) and the graphical user interface (view) are completely separated. With this architecture, you can easily change, adapt and maintain the software. This makes production in the present and in the future much more efficient and productive.

Furthermore, it is necessary to have an observer-like pattern. Every time the controller state changes, the view will be notified and redrawn automatically. With this pattern, you can decouple the view and the controller.

These architectures and patterns can be found in modern programs nowadays. To implement such an architecture, it is beneficial to have a programming language which supports classes, like C++ or Java. Of course, in this case the programming language C++ should be used since we are developing on a microcontroller.

IV. System overview

The developed system can be separated logically into several parts.
Following literature was used in the project:

"Building Arduino Projects for the Internet of Things Experiments with Real-World Applications" by Adeel Javed

"Embedded-Systeme mit der Arduino-Plattform" by Klaus Dembowski

A. Systemoverview

The platform which is used in the project is an Arduino Uno microcontroller board. It is based on an Atmega328P with 14 digital input / output pins and 6 analog inputs. On the board is all needed hardware to support the microcontroller and to accomplish a quick start. It also has an USB connection for a simple communication with a computer and a separate power connection to run the board on its own [4].

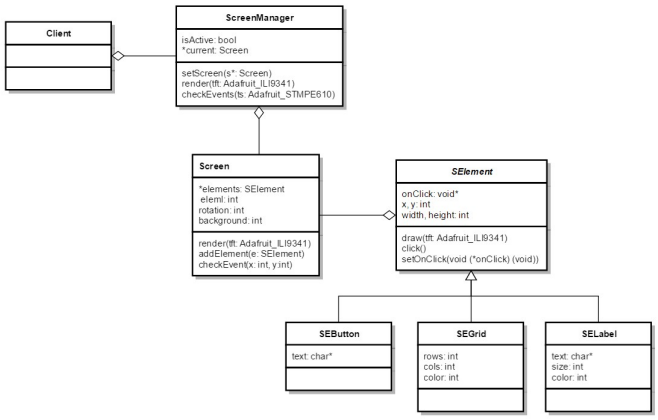
The system will be developed in the in-house SDK from Arduino, called "Arduino Studio". "The Arduino Studio is a new open source development environment for the Arduino Programming language" [5]. It provides a basic source code editor, a compiler for the created projects and a possibility to directly program the connected microcontroller board. It already contains some basic functions of C and C++ and it is highly extensible for custom software libraries. The main programming language is C++.

The other component which is used is a 2.8" TFT LCD touch display with a resolution of 240x320 pixels. It has a build in controller with RAM buffering, so that almost no computing must be done by the Arduino below. The display supports two modes: 8-bit and SPI. It can be powered with 3.3V or 5V such as the Arduino board [6].

B. Library

The specific software library which was developed is based on the Adafruit_ILI9341 library. This is a library for the Adafruit ILI9341 display products [3]. It's not very optimized, but for the project it's irrelevant, because there are no high frequently changes in the elements which are used. The specific library of the project is more an extension of the Adafruit library.

The idea here was to abstract the paradigm of a "screen"-based architecture. The elementary fractions where separated into C++ classes. The main functionality here is that the rendering on the display will be done completely by the library. The programmer just defines the graphical screens with concrete values in the main system, store these in an instance of a defined structure and hand it over to a by the library offered screen manager.



All this functionality will be used in by the programmer in the main system. For a basic use, an instance of the screen manager is required. The main routine has to call in each sequence the "render" and the "checkEvents" methods to establish the functionality to the user.

The screen-class is meant to be the main container. Screen elements will be added here. This element is transmitted to the screen manager where the member function of screen is called to render the defined elements. At this juncture, the screen manager is responsible to decide if there is a need to draw the current screen, so that just in case of a change a new screen will be rendered. This procedure avoids flickering.
The other main functionality of the screen manager is to detect tabs on the display. If so, it will transfer the exact coordinates of the tab to the currently loaded screen. It will decide if there is an element at this position and as the case may be call the defined function of the element. In case there is more than one element on top of each other at the tabbed position, all concerning elements will be executed in the same order as they were added to the screen.

The screen elements which are defined and added to a screen object are concrete implementations of the abstract class “screen element”. The main purpose of this technique is that any concrete implementation which inheritates of the superclass can be added to a screen object. The “screen element” structure also contains a function pointer, which needs to be defined per each instance by the programmer. This function will be called in case of a tab on this element.

For the project, we implemented three concrete elements. First there are two basic elements, which can be found in every graphical application. These are a button and a label. The button contains a framed rectangle with a lettering and the label is just a writing, that can be defined in color and size.
The last implementation is a grid element, which is needed for the playing field. It can be defined in size and color as well and in addition to that also in the number of cells in vertical and horizontal direction.

C. Graphical user interface

Another great aspect was the graphical user interface. The main requirements were, that it is easy to use, self-explanatory and internally consistent. Therefore, it is necessary to understand the options the GUI has to offer and to use significant key words which the user might know from his previous gaming experience. Those keywords are for example:

- Singleplayer
- Multiplayer
- Options
- Credits
- Exit

It is very important that the user don’t has to search for an option so that all attention can be spend to the actual game.
Another issue to attract a user is the design of the several elements. Important is, that it is upright and consistent. That means that all elements follow the same design like color, shape or font-style. It should look like something the user already knows. For example, he should realize that it is a button and he can click on it. It should be a mix-up of all possibilities the programmer can imagine.
This is achievable by the named issues, using key words and a straight design.

The third aspect regarding the internal consistence, is guaranteed by the specific software library. Only one interaction per element is allowed. It ensures, that the behavior of an element stays the same for the whole time it is active.

The last visualization element which should be mentioned is the intro. It consists of the Title of the game (“TicTacToe”) and some appealing animations. Because of the badly optimized Adafruit library and the fact that the specific library has not been designed to create animations. A completely separated routine was implemented in the startup sequence of the system to make this possible.

D. Gamelogic

The third part of the project was to reengineer and integrate the game and its logic. Related to the previous architecture there are some requirements to this controller.
It has to be a class on its own to be created in the main routine.
It has to support an “observable”-like architecture, so that a method can be defined and committed as parameter to wait for user or computer interaction and update the screen.
It has to have a clear order of actions the system could call to allow a smooth and accurate procedure.

The concrete actions in the game are pretty simple. In one turn the player must set the mark, the system has to verify the validity of the choice and gives feedback to the screen manager. Then the next player starts with his turn.

Most of the routine the controller itself is able to manage. Only on “start”, “player set mark” and “GUI callback” an interaction with the system or the user is necessary.

E. Computer driven player

The idea was to implements a single player mode, where you try to beat a computer driven player. The implementation would contain several TicTacToe-strategies which can be found on the world-wide-web.

Unfortunately, we were unable to develop such an entity because of a lack of time.

F. Extensibility

As mentioned, the extensibility was the last aspect of the project. Every part of the system is scalable, so that it is simple to create and to implement new content.

An example here might be a screen for the settings. It could be realized by adding a new button in the existing main menu screen.

New graphical elements can be added by inherit from the screen elements class.

V. Conclusion

From the requirements point of view, the project is succeeded. The Result is a working TicTacToe game running on an Arduino Uno board. The game can be played with a touch screen on top of the Arduino and is suitable for two human players. The system is user-friendly, scalable and consistent. To achieve this, modern programming patterns were used.
The project gave an understanding of the operating principle of a microcontroller in combination with a touch display.

VI. References

[1] "Arduino Games, Tic Tac Toe and Light Out"; Author: zefram; https://www.thingiverse.com/thing:212826

[2] "Robotic arm playing tic tac toe controller by Arduino"; Author: Jose L. B. Vasquez; https://www.hackster.io/sistemasymicros/robotic-arm-playing-tic-tac-toe-controlled-by-arduino-8e3244

[3] "Tic Tac Touch", Author: Jeremy; http://thecustomgeek.com/2011/07/18/tic-tac-touch/

[4] “Arduino Uno & Genuino Uno” Author: Arduino.org; “https://www.arduino.cc/en/Main/ArduinoBoardUno”

[5] “Arduino Studio” Author: Arduino Labs; “http://labs.arduino.org/tiki-index.php?page=Arduino+Studio”

[6] ”Adafruit 2.82 TFT Touch Shield” Author: Adafruit.com “https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2/overview”

[7] “Adafruit_ILI9341 graphic tibrary” Author: ladyada; “https://github.com/adafruit/Adafruit_ILI9341”