UbiCom TicTacToe: Blog

**Blog 1 Tic Tac Toe for Arduino:**

**Hi and welcome to our blog!**

In this blog we want to show our progress in our mini project as a part of ubiquitous computing.

The team consists of Julian Altmeyer and Lukas Luschin. We are two students of Applied Computer Science at HTWG in Konstanz Germany in the 5th semester.

**What ist the project about?**

As the title mostly reveals, our project is about implementing the well known game TicTacToe on an arduiono uno microcontroller board. The game as itself will be controlled with a small display on top, which allows users input over touch. Furthermore there will be the option to select between playing round based versus another human (or other creatures which are able to tab on a screen) or a single player mode, where your opponend will be emulated by the arduino.

**So how to start?**

We decided to devide our project into several parts which we will blog here separatly. First of all we have to create a reliable library for the display which offers us all the needed functionality to create a userfriendly and good-looking graphical user interface. Next Step is to create exactly this UI. What we want to define is which possibilities the user has and how to establish that. If this is done we can start with creating an architecture for our game and code it. At last we implement the single players ‘AI‘, because this will be a less important feature.

**Blog 2 Connecting to the Display:**

greetings!

After a lot of formality we finaly want to start with our project.

As meantioned we start with the connection to the display. We use a 2.8“ TFT Display with touchscreen. Here we found a great tutorial and also a software library with lot of functionality and demonstration projects from adafruit.

Tutorial:

<https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2/overview>

Library:

https://github.com/adafruit/Adafruit\_ILI9341

With this knowledge and features we created a for the project more specific software library with functionality like UI-elements for the menu and the game.

Therefore we try to consider every need of our game.

**Creating a screen**

So we abstract the idea of an screen based architecture, where you can add several elements to. These elements are defined by several classes with different content. All those element classes have the same base-class which includes an abstract „drawing“ and an „onClick“ function.

Furthermore we need an instance which manages this Screens to avoid unneccessary redrawing of a screen.

The final order of functions calling is that our main instance calls a rendering function in the screen manager, which decides if the current screen needs to be rendered. If yes, the render function of the actual screen instance is called. It includes the creation of the screen and the filling with the defined elements.

**Touch events**

The Screen Manager also takes responsibility for touch events. It gets the last touched position on the screen and hands it over to the actual screen instance. It checks if there is an element at this position and calls the corresponding „onClick“ function where appropriate.

So the final architecture looks like this:

* UML hier einfügen!

**Blog 3 GUI:**

Welcome back

After creating a framework for our graphical user interface it’s time for another design phase. The objective is to create a GUI which is understandable, easy in use and optically attractive as well.

Therefore it is necessary to understand the options the GUI has to offer and to use significant key words.

In our case, we decide to create four main menu items:

* Singleplayer / one player
* Multiplayer / two player
* Options
* Credits

Because that this key words are wisely chosen, there is no need to explain what they’re doing.

Okay, so the main menu is defined. Now let’s have a closer look on the program flow.

**Startup**

Once the system started, a short “intro” will appear. It contains the title of the game and some fancy animation to attract the user. After a few seconds the main menu will be shown, where the user enter his choice.

**Singleplayer / Multiplayer**

This option will initialize the game controller and a new game will be started (either with two players or one player and a computer. We’ll see later). The main part of the game-screen will be the three by three grid for the game itself. On input the user specific sign will be drawn. Traditionally it will be an ‘X’ and a ‘O’. Furthermore there is a button for exiting the game and on top an info panel to give minor information about the current game.

**Options**

The “option” item could contain several settings or attributes which the system offers the user to specify.

**Credits**

The last menu item will open another screen with general information about the system, the game itself and about the guys who created that stuff.

The main menu is created. In the next steps we’ll create the game-controller and integrate it in the system. It might be that we won’t use all of the menu items, but we’ll keep them for presentation.

**Blog 4 Gamelogic:**

Hello again

Now the system is prepared and it’s time to integrate the game-controller. For 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 should call to allow a smooth and accurate procedure.

Here we defined a sequence for the TicTacToe game:

* Diagramm

All “player interactions” is functionality defined in the system. Either it’s waiting for a touch on the display or computing the next step.

**Blog 5 Résumé:**

welcome to our last blog,

this time we just want to review the project and talk a little bit about the extensibility of our system.

**First things first**

What we've done is, crested a specific software library for the game tic tac toe, developed a user-friendly GUI, implemented all needed mechanisms to detect tabs on the display and render screens. Last but not least we developed a game-controller and integrated it in our system for a two player tictactoe game. All based on an Arduino Uno Microcontrollerboard.

**Extensibility**

The last aspect we spent attention to is the extensibility. Every part of the system is scalable, so that everyone who want's to can extend the software with own content. An example is the library we developed. If there is the need to create further GUI-items, you can do that just by simple inheritance.

Another planed, but not implemented idea is, to create a computer controlled player to play against. This wasn't done because for a lack of time, but wouldn't take much effort to implement.

To conclute, the project was successful. All expected objectives are achieved and we have a mostly new game to play with.