FACHHOCHSCHULE
ERFURT UNIVERSITY
OF APPLIED SCIENCES
Angewandte
Informatik

# Icon-Editor in C#

Praxisprojekt

**Daniel Depta**

Abgabedatum: 05.04.2021

Betreuung Projektarbeit: **Prof. Dr. Uwe Altenburg**

# Inhaltsverzeichnis

# 1. Allgemeine Beschreibung

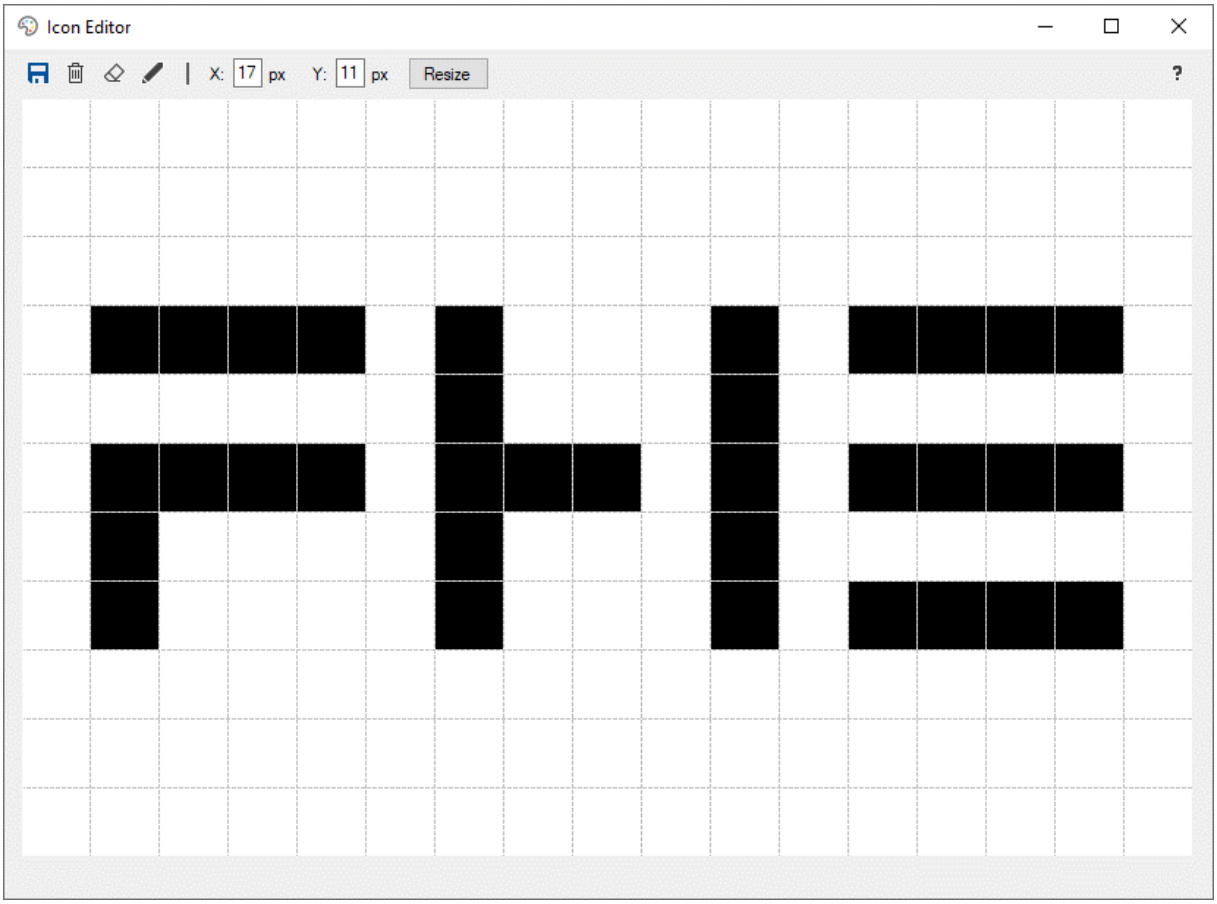
Der Icon Editor ist ein in C# entwickeltes Programm für Windows 10, um monochrome Grafiken (schwarz und weiß) zu gestalten. Die Icons können zwischen 1x1 und 64x64 Pixel groß sein. Im Programm werden die einzelnen Pixel dann entsprechend größer dargestellt, um das Zeichnen zu vereinfachen. Zwischen den Pixeln gibt es Gitternetzlinien, um die Pixel voneinander abzutrennen.

Die Zeichnung kann jederzeit als PNG-Grafik exportiert werden. Des Weiteren kann die Leinwand bei Bedarf wieder geleert werden. Standardmäßig ist der schwarze Stift ausgewählt, um die Pixel auszufüllen. Mit dem Radiergummi können einzelne Pixel entfernt werden.

Die Anzahl der Pixel kann auch während des Zeichnens angepasst werden. Das Programm kann auch vergrößert und verkleinert werden. Die Pixelgröße wird dabei immer neu berechnet, damit die Pixel so groß wie möglich und quadratisch sind.

Die Programmsprache ist in Englisch gehalten.

Das Repository ist unter [https://github.com/ddepta/iconeditor](https://github.com/ddepta/iconeditor) zu finden.



# 2. Projektmanagement

## 2.1 Aufgabenstellung

Es soll ein Icon-Editor in C# geschrieben werden, der das Icon in einem Array speichert.
Die Icons sind zunächst nur schwarz/weiß, könnten aber später auch farbig sein.

Man soll die Größe des Icons angeben, z.B. 16x16 und danach dann die Pixel zeichnen können. Die Icons sind mindestens 1x1 und maximal 64x64 Pixel groß. Sie müssen nicht quadratisch sein.

Es wäre super, wenn das eine C#-Klasse wäre, die man in ein bestehendes Projekt einbinden kann. Es muss also kein großes Programm werden, eher alles über Methoden steuerbar.

Die Programmoberfläche soll in Englisch gehalten werden. Komfortfunktionen wie Rückgängigmachen und Wiederholen wären angenehm, aber sind nicht notwendig.

## 2.2 Zielbestimmung

### 2.2.1 Musskriterien

Der fertige Icon-Editor muss folgende Kriterien erfüllen:

- Zeichenfunktion wie beispielsweise der Stift in Microsoft Paint
- Icon in einem Array speichern
- Auflösung des Icons änderbar, voreingestellt: 16x16 Pixel
- Mindestauflösung: 1x1 Pixel, Maximalauflösung: 64x64 Pixel
- Umsetzung von quadratischen und nicht-quadratischen Icons
- Hintergrund weiß, Stiftfarbe schwarz
- Umsetzung in C#
- Programmoberfläche in Englisch

### 2.2.2 Wunschkriterien

- Icons auch farbig zeichnen
- C#-Klasse, über Methoden steuerbar
- Rückgängigmachen und Wiederholen
- Export des Icons als Bild (bspw. PNG)
- Anpassbare Fenstergröße
- Export des Arrays als C-Header
- Wiedereinlesen von erstellten Icons, um sie weiter bearbeiten zu können

### 2.2.3 Abgrenzungskriterien

- Icons größer als 64x64 Pixel können nicht gezeichnet werden

## 2.3 Ablauf/Meilensteine

1. Themenfindung für das Praxisprojekt: **30.05.2020**
2. Konkretisierung Aufgabenstellung, Anforderungsanalyse: **21.10.2020**
3. Umsetzung der Leinwand (Zeichnungsfunktion): **26.10.2020**
4. Darstellung der Gitternetzlinien: **26.10.2020**
5. Leinwandgröße/Pixelgröße und Auflösung anpassbar: **16.01.2021**
6. Umsetzung der benötigten Features (Radiergummi, Leinwand leeren, speichern): **16.01.2021**
7. „Unsichtbare“ Pixel löschen, wenn Auflösung angepasst wird: **03.04.2021**
8. Validierung der Benutzereingaben: **04.04.2021**
9. Fertigstellung der Benutzeroberfläche: **04.04.2021**
10. Fertigstellung des Codes/Programms: **04.04.2021**
11. Fertigstellung Dokumentation: **05.04.2021**

## 2.4 Aufwandschätzung

| Vorgang | Dauer (h) |
|---|---|
| **Anforderungsanalyse** | 10 |
| **Entwicklung – Coding** | 45 |
| **Definition Testfälle** | 2 |
| **Testdurchführung** | 2 |
| **Dokumentation – Anforderungen** | 4 |
| **Dokumentation – Projektmanagement** | 4 |
| **Dokumentation – Implementierung** | 8 |
| **Gesamt** | 75 |

## 2.5 Projektablauf

Das Projekt wurde nach der agilen Arbeitsmethode durchgeführt. Es wurde in drei Sprints mit einer Sprintdauer von jeweils ungefähr einer Woche gearbeitet. Es wurde vorher keine feste Dauer der Sprints festgelegt, da am Projekt gearbeitet wurde, wenn neben dem Studium Zeit dafür war.

Folgende Sprints wurden durchgeführt:

1. Sprint: 18.10.2020 bis 26.10.2020
2. Sprint: 12.01.2021 bis 16.01.2021
3. Sprint: 01.04.2021 bis 05.04.2021

Während des ersten Sprints fand ein Sprint Review mit Herrn Altenburg statt. Im Vorfeld wurde bereits ein Prototyp entwickelt, um die Anforderungen zu konkretisieren. So konnten die Zielbestimmungen genauer definiert werden.
Nach dem zweiten Sprint fand ebenfalls ein Sprint Review statt, wo die Anforderungen für den dritten und letzten Sprint definiert wurden.

| **Vorgang** | **Dauer (h)** | **Fertigstellung Meilenstein** | | | | |
|---|---|---|---|---|---|---|
| **Anforderungsanalyse** | | | | | | |
| Themenfindung | 2 | 30.05.2020 | | | | |
| Aufgabenstellung, Zielbestimmung | 6,5 | | 21.10.2020 | | | |
| | | | | | | |
| **Entwicklung** | | | | | | |
| Zeichnungsfunktion | 20 | | | 26.10.2020 | | |
| Gitternetzlinien | 4 | | | 26.10.2020 | | |
| Leinwandgröße/Icon-Auflösung änderbar | 13,5 | | | | 16.01.2021 | |
| sonstige Zeichenfeatures | 11 | | | | 16.01.2021 | |
| Validierung | 3 | | | | | 04.04.2021 |
| | | | | | | |
| **Tests** | | | | | | |
| Definition Testfälle | 2,5 | | | | | 01.04.2021 |
| Testdurchführung | 1,5 | | | | | 02.04.2021 |
| | | | | | | |
| **Dokumentation** | | | | | | |
| Anforderungen | 3 | | 21.10.2020 | | | |
| Projektmanagement | 5 | | | | | 05.04.2021 |
| Implementierung | 9 | | | | | 04.04.2021 |
| | | | | | | |
| **Gesamt** | 81 | | | | | |

# 3. Produktfunktionen

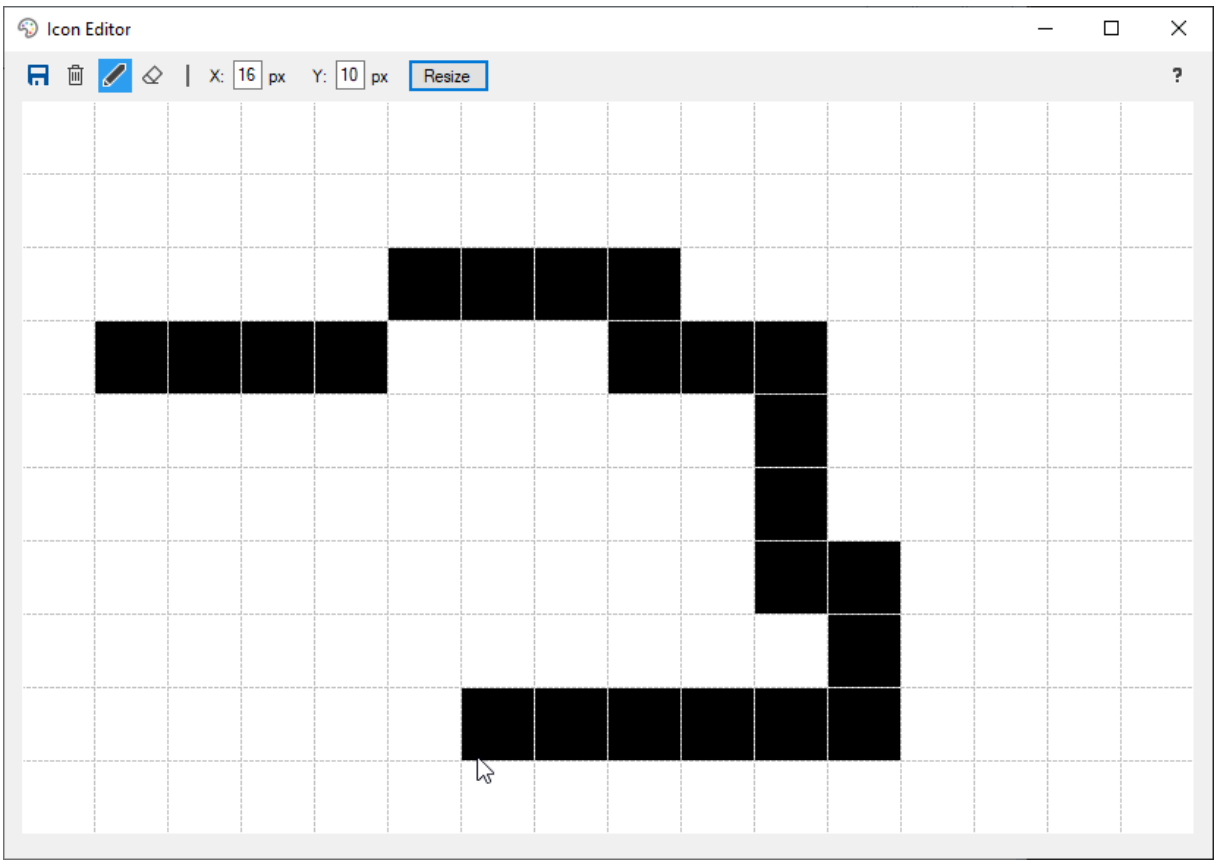
## 3.1 Zeichnen auf der Leinwand (Zeichenstift)

 Zeichenstift

Beim Start der Anwendung ist automatisch der Stift ausgewählt, mit dem auf die Leinwand gezeichnet werden kann. Wenn der Stift ausgewählt ist, ist er mit einer blauen Hintergrundfarbe hinterlegt.

Man kann per Mausklick (linke Maustaste) einzelne Pixel zeichnen, oder man hält die Maustaste gedrückt und kann somit mehrere Pixel hintereinander ausfüllen.

Beim Zeichnen wird an der Position der Maus ein schwarzes Viereck mit der berechneten Pixelgröße (siehe **3.6 Pixelgröße berechnen**) eingefügt. Zusätzlich wird die Zeichnung in einem zweidimensionalen Array gespeichert, um das Exportieren und Neuzeichnen zu realisieren.




## 3.2 Zeichnung als Grafik exportieren

 Zeichnung exportieren

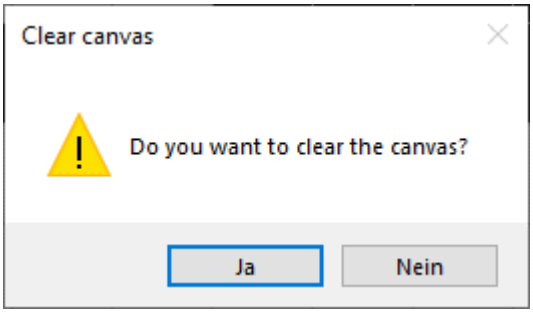
Über diesen Button kann die aktuelle Zeichnung exportiert bzw. gespeichert werden. Es öffnet sich ein „Speichern unter“-Dialog, wo man den Speicherort und den Dateinamen festlegt. Die Grafik wird im PNG-Format gespeichert, da dies eine verlustfreie Komprimierung ermöglicht.

Die Grafik wird in der Auflösung der Zeichnung gespeichert. Ist die Zeichnung beispielsweise 16x16 Pixel groß, dann ist die gespeicherte Grafik auch 16x16px groß.
Der weiße Hintergrund ist nach dem Export auch weiß und nicht transparent.

## 3.3 Leinwand leeren

 Leinwand leeren

Wenn man diesen Button betätigt, dann wird der Benutzer gefragt, ob er die Leinwand leeren möchte. Wenn er dies bestätigt, werden alle Pixel auf der Leinwand und im zweidimensionalen Icon-Array auf weiß gesetzt. Dieser Vorgang kann nicht rückgängig gemacht werden.




## 3.4 Einzelne Pixel löschen (Radiergummi)

 Radiergummi

Mit diesem Button können bereits gezeichnete Pixel wieder gelöscht (weiß gefärbt) werden. Die Funktionsweise ist dabei wie beim Zeichenstift. Man kann per Mausklick (linke Maustaste) einzelne Pixel löschen, oder man hält die Maustaste gedrückt und kann somit mehrere Pixel hintereinander löschen. Wenn der Radiergummi ausgewählt ist, ist er mit einer blauen Hintergrundfarbe hinterlegt.

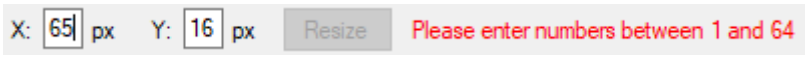
## 3.5 Auflösung (Anzahl der Pixel) verändern


Zum Start des Programms ist eine Auflösung von 16x16 voreingestellt. Als Auflösung der Zeichnung können Ganzzahlen zwischen 1x1 und 64x64 verwendet werden. Dabei müssen die beiden Seiten nicht gleichlang sein, es kann auch beispielsweise eine Auflösung von 32x16 angegeben werden. Nach der Eingabe der neuen Auflösung muss die Veränderung mit Klick auf den „Resize“-Button durchgeführt werden.

Wenn die Auflösung geändert wurde, wird die Pixelgröße neu berechnet (siehe **3.6 Pixelgröße berechnen**) und die Leinwand geleert. Danach werden die Gitternetzlinien neu eingezeichnet (siehe **3.7 Gitternetzlinien einzeichnen**). Zum Schluss wird die Zeichnung mit den neuen Dimensionen neu eingezeichnet. Dafür wird das zweidimensionale Array verwendet, welches die bisherige Zeichnung enthält.

Kommen Pixel hinzu, dann wird die bisherige Zeichnung nicht verändert, die Leinwand wird um weiße Pixel erweitert. Wenn Pixel entfernt werden, dann bleiben die noch sichtbaren Pixel der Zeichnung bestehen. Teile der Zeichnung, die jetzt nicht mehr sichtbar sind, sind unwiderruflich gelöscht.

Sollten die eingegebenen Werte zu klein, zu groß oder anderweitig ungültig sein, dann wird der „Resize“-Button deaktiviert und ein Hinweis angezeigt. Wenn die Eingabe wieder korrekt ist, wird der Button wieder aktiviert und der Hinweis ausgeblendet.



## 3.6 Pixelgröße berechnen

Die Leinwand verwendet intern die Auflösung des Programms und nicht die im Programm eingestellte Auflösung der Zeichnung. Dadurch muss die Größe eines Pixels auf die Größe der Leinwand angepasst werden.
Dazu wird die Breite und die Höhe der Pixel berechnet und abgerundet, da technisch nur ganzzahlige Pixel dargestellt werden können.

$$\lfloor Pixelbreite \rfloor = \frac{Leinwandbreite}{Anzahl\ Pixel\ (X)} - Breite\ Gitternetzlinie$$

$$\lfloor Pixelhöhe \rfloor = \frac{Leinwandhöhe}{Anzahl\ Pixel\ (Y)} - Breite\ Gitternetzlinie$$

Danach werden die Pixelbreite und die Pixelhöhe miteinander verglichen und der kleinere Wert von beiden wird sowohl für die Breite als für die Höhe genommen, um quadratische Pixel zu erzeugen. Wenn man den größeren Wert nehmen würde, würde eine Seite zu lang werden und über die Fläche der Leinwand hinausgehen.

Danach wird die neue Größe der Leinwand berechnet. Die Leinwandgröße basiert auf der Größe der Pixel. Weil die Pixelgröße abgerundet wurde, ist die Leinwand nun kleiner als eigentlich vorgegeben.

$$Leinwandbreite = (Pixelgröße * Anzahl\ Pixel\ (X) + (Anzahl\ Pixel\ (X) - Breite\ Gitternetzlinie)$$

$$Leinwandhöhe = (Pixelgröße * Anzahl\ Pixel\ (Y) + (Anzahl\ Pixel\ (Y) - Breite\ Gitternetzlinie)$$

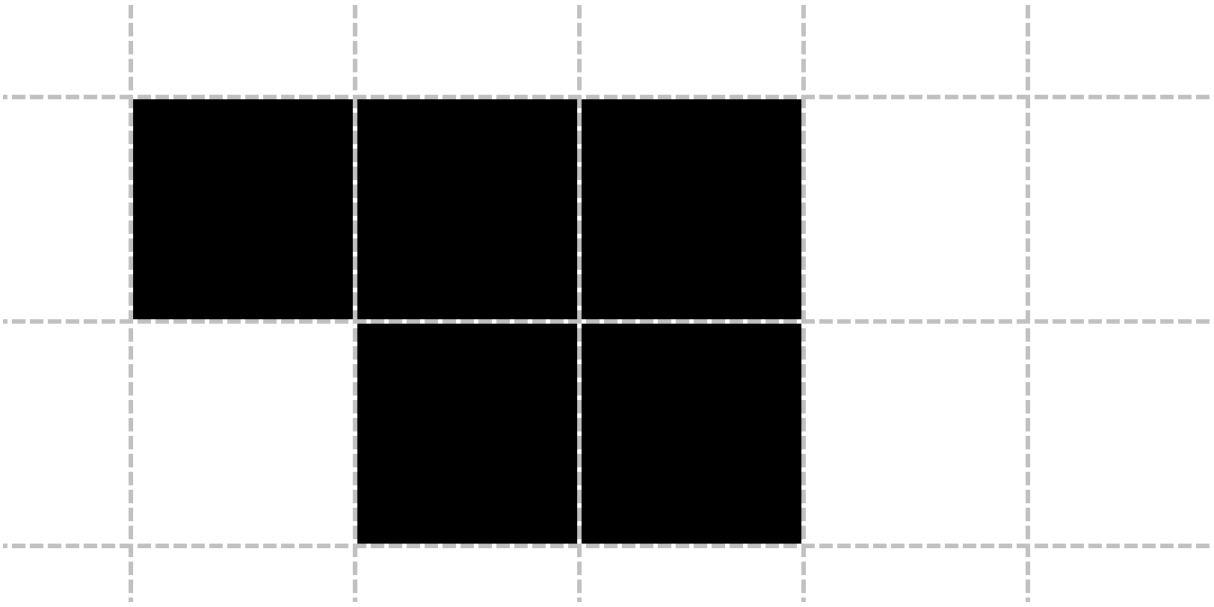
Im Programmcode ist das so umgesetzt:

```
private void CalculatePixels()
{
    canvas_width = Size.Width - 40;
    canvas_height = Size.Height - 85;
    int canvas_pixelsize_width = (canvas_width / x_size) - 1;
    int canvas_pixelsize_height = (canvas_height / y_size) - 1;

    canvas_pixelsize = canvas_pixelsize_width
        .CompareTo(canvas_pixelsize_height) == 1 ? canvas_pixelsize_height : canvas_pixelsize_width;

    canvas_width = (canvas_pixelsize * x_size) + (x_size - 1);
    canvas_height = (canvas_pixelsize * y_size) + (y_size - 1);

    Size panelSize = new Size(canvas_width, canvas_height);
    canvas.Size = panelSize;
}
```

## 3.7 Gitternetzlinien einzeichnen

Zwischen den einzelnen Pixeln befindet sich ein gestricheltes Gitternetz.
Dies wird zum Start des Programms, beim Verändern der Auflösung und beim Verändern der Fenstergröße neu gezeichnet.

Dazu werden horizontale Linien (Anzahl Y-Pixel – 1) und vertikale Linien (Anzahl X-Pixel – 1) zwischen die Pixel gezeichnet.



## 3.8 Fenstergröße ändern

Die Größe des Programms kann beliebig angepasst werden. Dazu muss am Rand des Programms (wie bei Windows-Programmen üblich) gezogen werden.

Wenn sich die Größe des Programms ändert, wird die Pixelgröße neu berechnet (siehe **3.6 Pixelgröße berechnen**) und die Leinwand geleert. Danach werden die Gitternetzlinien neu eingezeichnet (siehe **3.7 Gitternetzlinien einzeichnen**). Zum Schluss wird die Zeichnung mit den neuen Dimensionen neu eingezeichnet. Dafür wird das zweidimensionale Array verwendet, welches die bisherige Zeichnung enthält.