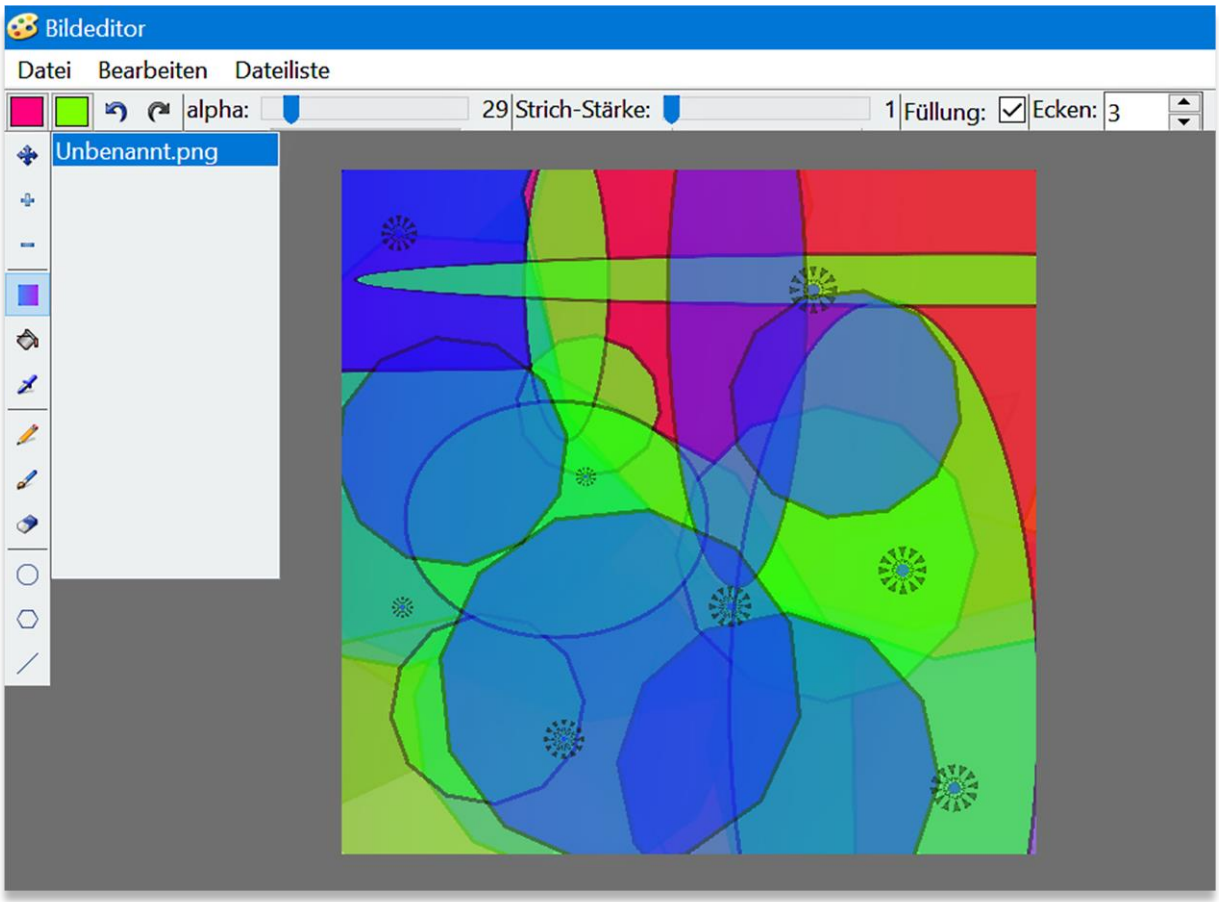
Projekt „Bildeditor“

Komplexe Leistung im

Fach Informatiksysteme



Henning Beyer

Klasse A20/2

Schuljahr 2020/21

Inhaltsverzeichnis

1 Einleitung

Eine Bildbearbeitungssoftware mit allen Tools, die man gerade braucht: Das wünscht sich fast jeder, der schon öfter mit Bilddateien arbeiten musste. Ob für ein Kunstwerk, für ein hübsches Profilbild oder eine Skizze: überall hilft ein Bildbearbeitungsprogramm, wie Photoshop oder Gimp. Jedoch sind die meisten der Programme nicht immer kostenlos, einfach zu bedienen oder umfangreich genug, um reibungslos mit ihnen arbeiten zu können. Aber ein Programm, nämlich Paint.NET, ist nahezu in allen Kriterien perfekt. Es bietet auf kompakter Oberfläche alle nötigen Tools zur Bildbearbeitung und ist äußerst einfach zu bedienen. Deshalb ist es auch zu meinem primären Bildbearbeitungsprogramm geworden. Paint.NET entstand damals um 2004 als Studentenprojekt und wurde in nur vier Monaten fertiggestellt.[1] Aufgrund der Einfachheit und gleichzeitigen Qualität von Paint.NET, hat mich dieses Programm dazu inspiriert ein ähnliches Bildbearbeitungsprogramm „Bildeditor“ für diese komplexe Leistung zu programmieren.

2 Projektbeschreibung

Die Aufgabe für diese komplexe Leistung ist es, eine kleine Anwendung in Richtung Betriebswirtschaft, Naturwissenschaften oder Computerspiele in Pascal zu schreiben und diese wie in einer Facharbeit zu dokumentieren. Für diese komplexe Leistung habe ich das Programm „Bildeditor“ mithilfe der Lazarus-IDE geschrieben.

Mein Programm „Bildeditor“ ist ein kleines Bildbearbeitungsprogramm, welches darauf abzielt mit einer übersichtlichen Benutzeroberfläche, relativ viele Möglichkeiten zur Bearbeitung eines Bildes zu bieten. Hierbei sind neben den klassischen Zeichentools, wie Bleistift, Pinsel oder Farbeimer, auch wie in Paint.NET, einige Bildeffekte wie z.B. der radiale Weichzeichner oder das Verschärfen implementiert. Mit diesem Programm können auch Bilder vom Dateityp PNG, JPG, TIFF und BMP geöffnet und gespeichert werden. Gleichzeitig bietet eine Dateiliste die Möglichkeit, gleich mehrere Bilddateien auf einmal zu bearbeiten.

[1] Vgl. Wikipedia, 2021.

2.1 Dokumentation der Programmentwicklung

Am 08.05.2021 haben wir die Aufgabenstellung für die komplexe Leistung erhalten. Von da an habe ich mir in der ersten Woche Gedanken über meine Projektidee gemacht. Zuerst hatte ich die Idee eine Anwendung zu entwickeln, welche Bilddateien für die Datenanalyse vorbearbeitet. Dass heißt, es sollten Bilddateien je nach Wunsch z.B. verkleinert werden oder auch ein ganzer Datensatz an Bildern so aufbereitet werden, dass z.B. ein maschineller Lernalgorithmus daraus zusätzliche Strukturen erkennen könnte. Diese Idee war von der Oberfläche her gut umzusetzen, aber der Aufwand wäre viel zu groß gewesen, die einzelnen Algorithmen für die Verarbeitung selbst zu programmieren. Leider gab es auch keine Lazarus-Packages, welche solche Bildverarbeitungs-Algorithmen implementierten. Ich habe zu dieser Idee am Anfang der zweiten Woche einen Programmentwurf für die Benutzeroberfläche angefertigt, welchen ich auch für die Idee meines Programms „Bildeditor“ übernommen habe. Im Entwurf hatte ich bereits die Listbox-Komponente mit Mehrfachauswahl, ebenso wie das Öffnen mehrerer Bilddateien implementiert.

Am Ende der zweiten Woche suchte ich weiter nach einem passenden Lazarus Package für meine erste Idee, konnte aber keine passenden Packages finden. Stattdessen bin ich dabei auf das Package BGRABitmap gestoßen, welches mich zu meiner Idee des „Bildeditors“ gebracht hat. Nämlich beinhaltet dieses Package alles, was für ein Bildbearbeitungsprogramm gebraucht wird. Und tatsächlich wurde schon mit diesem Package ein Bildbearbeitungsprogramm erstellt: LazPaint. Dieses ist ebenfalls ähnlich wie Paint.NET aufgebaut, dient jedoch nur zur Vorstellung der Package-Inhalte.[2]

Seit der Nutzung von BGRABitmap konnte ich ab der dritten Woche am 19.05.2021 die Codierung des Projektes vornehmen. Dabei implementierte ich zuerst alle Zeichenwerkzeuge und eine Zoomfunktion, um in das Bild hinein- und herauszoomen zu können. In der vierten Woche implementierte ich das Speichern von Bildern, alle Bildeffekte, sowie eine Rückgängig und Wiederholen Funktion. Die restliche Zeit verbrachte ich mit der Testphase und dem Schreiben der Dokumentation des Programmes.

[2] Vgl. Circular, 2021.

3 Benutzeroberfläche

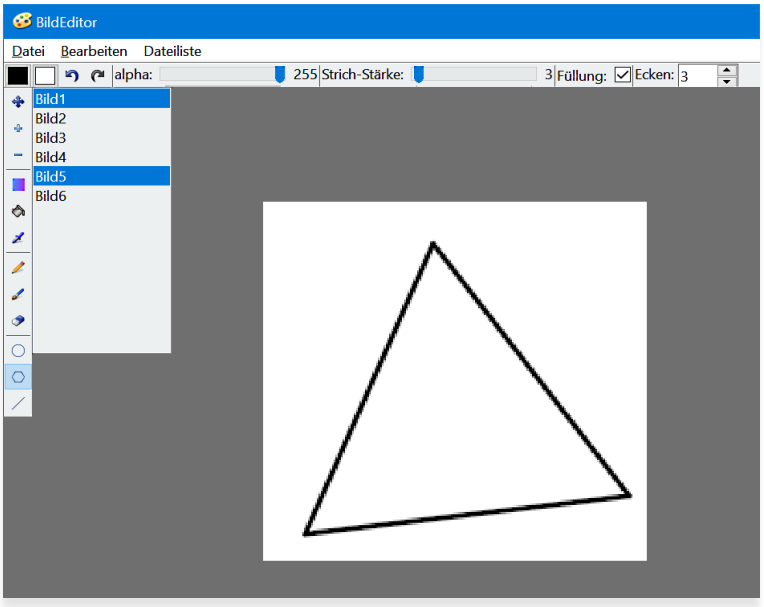


Abb. 1: Die Benutzeroberfläche des Programms „Bildeditor“ mit gezeichnetem Dreieck und sechs geladenen Bildern in der Dateiliste, von denen zwei Bilder ausgewählt sind.

Die Benutzeroberfläche besteht, wie in der Abbildung, ausfolgenden Komponenten:

- einem Hauptmenü mit drei Schaltflächen
- einer Toolbar mit den Zeichentools
- einer Toolbar für andere Tools und Einstellungen für das Malen
- eine Dateiliste, mit der auf alle geladenen Bildbitmaps zugegriffen werden kann
- und einer Zeichenfläche, auf die gemalt werden kann

3.1 Toolbars

Im Programm sind zwei Toolbars implementiert. Die linke Toolbar enthält die Schaltflächen für die Zeichentools. Die obere Toolbar enthält einige Komponenten für die Festlegung von Zeichentoolparametern dieser Zeichentools, sowie die Schaltflächen für Rückgängig und für das Aufheben des Rückgängig-Schrittes.

3.1.1 Zeichentools

Die ersten oberen Tools sind Tools für Bewegen des Bildes und das Heran- und Herauszoomen. Diese Tools dienen dazu das Zeichnen auf zu großen und zu kleinen Bildern angenehmer zu machen. Beim Bewege-Tool kann die Bildfläche mit

Linksklick und Bewegung der Maus bewegt werden; aber es ist auch möglich das Bild mit der rechten Maustaste zu bewegen ohne dieses Tool zu aktivieren. Das Zoomen erfolgt mit jedem Klick auf die Schaltfläche. Es kann ebenfalls mit dem Mausrad gezoomt werden. Noch beim Zoomen anzusprechen ist, dass das Bild bei jedem Zoom in der Mitte des Fensters platziert wird, also nicht an bzw. von der Stelle des Mauszeigers zoomt.

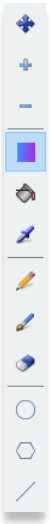


Abb. 2: Die linke Toolbar mit ihren Zeichentools.

Die nächsten drei Tools sind der lineare Farbverlauf, der Farbeimer und das Pipetten-Tool. Das Gradienten-Tool malt einen linearen Farbverlauf über die ganze Bildfläche mit der ausgewählten Transparenz. Das Eimer-Tool füllt, wie bei anderen Programmen auch, eine Fläche mit der ausgewählten Farbe aus. Und die Pipette wählt die Farbe des angeklickten Bildpixels aus.

Danach folgen das Bleistift-Tool, das Pinsel-Tool und der Radiergummi: Der Bleistift malt immer mit einer Pixelgröße von einem Pixel; der Pinsel malt mit einer Antialiasing-Linie und der Radiergummi radiert mit Transparenz. Bei allen drei Tools ist die Transparenz einstellbar und - außer beim Bleistift - kann auch die Linienstärke eingestellt werden.

Die letzten drei Tools in der linken Toolbar sind dann noch das Ellipsen-Tool, das Vieleck-Tool und das Linien-Tool. Alle Tools malen natürlich die Form ihres Schaltflächensymbols. Linienstärke, Transparenz, Füllung und Anzahl der Ecken können mit der oberen Toolbar festgelegt werden. Beim Halten der Shifttaste malen alle Tools gleichmäßig: Das heißt, dass die Ellipse als Kreis, das Vieleck mit Grundseite nach unten und die Linie im rechten Winkel gezeichnet wird.

Interessant beim Vieleck-Tool sind neben der Lösung des Algorithmus mit Vektoren, welche mit einer Rotationsmatrix rotiert werden, auch die speziellen Formen, welche bei großer Linienstärke und geringer Größe entstehen können:

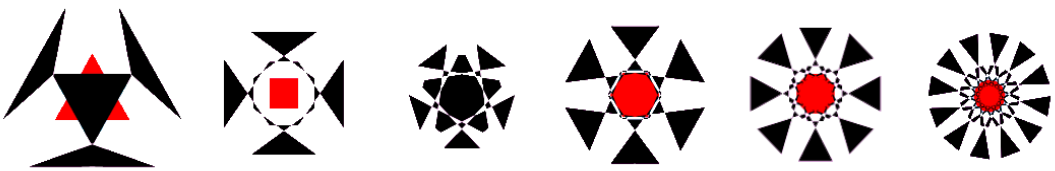


Abb. 3: Spezielle Formen des Formen-Tools bei großer Linienstärke. Die Anzahl der äußeren Dreiecke entspricht der Anzahl an Ecken.

3.2 Hauptmenü

Das Hauptmenü besitzt die drei Schaltflächen Datei, Bearbeiten und Dateiliste und schafft eine übersichtlichere Benutzeroberfläche.

3.2.1 Datei

Unter dem Reiter Datei kann der Benutzer neue Projekte erstellen, mehrere Bilder öffnen, sowie speichern und der Benutzer kann hier ebenfalls das Programm beenden.

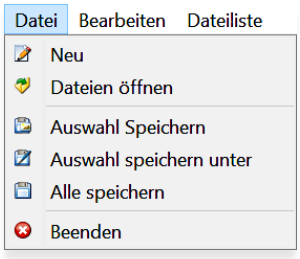


Abb. 4: Das Popupfenster für Datei

Falls auf „Neu“ geklickt wird, öffnet sich ein Fenster, wie in Abbildung 5, in welchem der Benutzer die Maße und den Namen des neuen Bildes angeben kann.

Wenn auf „Dateien öffnen“ geklickt wird, öffnet sich ein Dialog, mit welchem der Nutzer mehrere Bilddateien gleichzeitig öffnen kann. Unterstützte Dateiformate beim Öffnen aber auch beim Speichern sind PNG, JPG, TIFF und BMP.

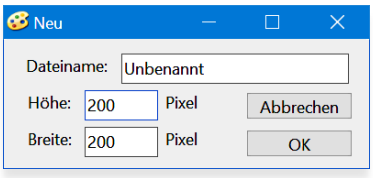


Abb. 5: Das Fenster, um ein neues Bild zu erstellen

Falls auf eine Schaltfläche für das Speichern geklickt wird, öffnet sich ebenfalls ein Fenster, mit welchem der Nutzer simpel alle zum Speichern ausgewählten Bilder speichern kann. Der Benutzer kann entscheiden, ob er alle Bilddateien in ein Verzeichnis speichert oder sich einzeln für jede Datei mit einer Vorschau des ausgewählten Bildes entscheiden. Zum Speichern kann der Benutzer auf der Oberfläche den Speicherpfad mit einem Dialog und das gewünschte Dateiformat über ein Dropdown-Menü auswählen.

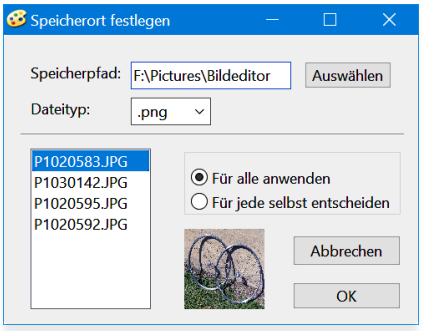


Abb. 6: Das Fenster, um Bilddateien zu speichern

3.2.2 Bearbeiten

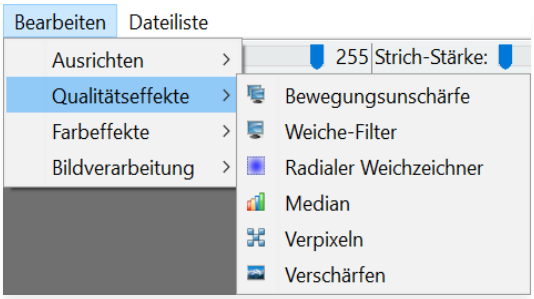


Abb. 7: Das Popupfenster für „Bearbeiten“ mit ausgeklappten Qualitätseffekten

Unter Bearbeiten sind 15 verschiedene Effekte implementiert, die auf das Bild angewendet werden können. Für manche der Bildeffekte von BGRABitmap gibt es eigene Fenster, um die Einstellung der Parameter vorzunehmen, jedoch werden die meisten Effekte nur mit Klick auf die jeweilige Schaltfläche angewandt. Einige Bildeffekte neben denen in Abbildung 7, sind unter anderem noch der Grauton, das Resampling, das Spiegeln des Bildes oder das Hervorheben der Konturen.

3.2.3 Dateiliste

Dieser Reiter ist dazu da, praktische Prozeduren für die Dateiliste mit zusammenzufassen. Mit „Anzeigen“ kann der Nutzer die Dateiliste ein- und ausblenden und mit „Mehrfachauswahl“ kann man die Mehrfachauswahl an- und abschalten. Die Mehrfachauswahl wird momentan nur für das auswählen bestimmter Bilddateien für das Speichern oder Löschen verwendet.

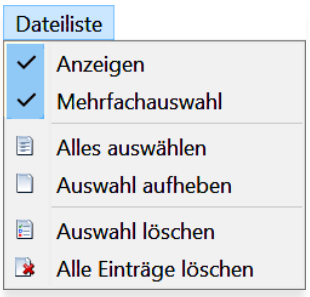


Abb. 8: Das Popupfenster für Dateiliste

3.3 Shortcuts

Für die schnelle Benutzung der Oberfläche sind folgende Shortcuts implementiert:

- Strg + N: Neues Bild erstellen
- Strg + O: Bilder im Verzeichnis Öffnen
- Strg + S: Alle veränderten Bilder speichern
- Strg + Z: Zeichenschritt rückgängig machen
- Strg + R: Rückgängig-Schritte rückgängig machen
- RMB auf Dateiliste: Angeklicktes Element abwählen
- RMB auf Bild: Das Bild kann bewegt werden
- LMB + Shift: Mit Kreis-, Form- und Linientool gleichmäßig malen

4 Programmablaufplan

Für diese Komplexe Leistung war es explizit die Aufgabe, ein Struktogramm, Syntax-Diagramm oder einen Programmablaufplan für das ganze Programm anzufertigen. Aufgrund der umfangreichen Programmstruktur mit vielen einzelnen Prozeduren und Schaltflächen, stelle ich den PAP möglichst übersichtlich und deshalb zusammengefasst dar in Abbildung 9 dar:

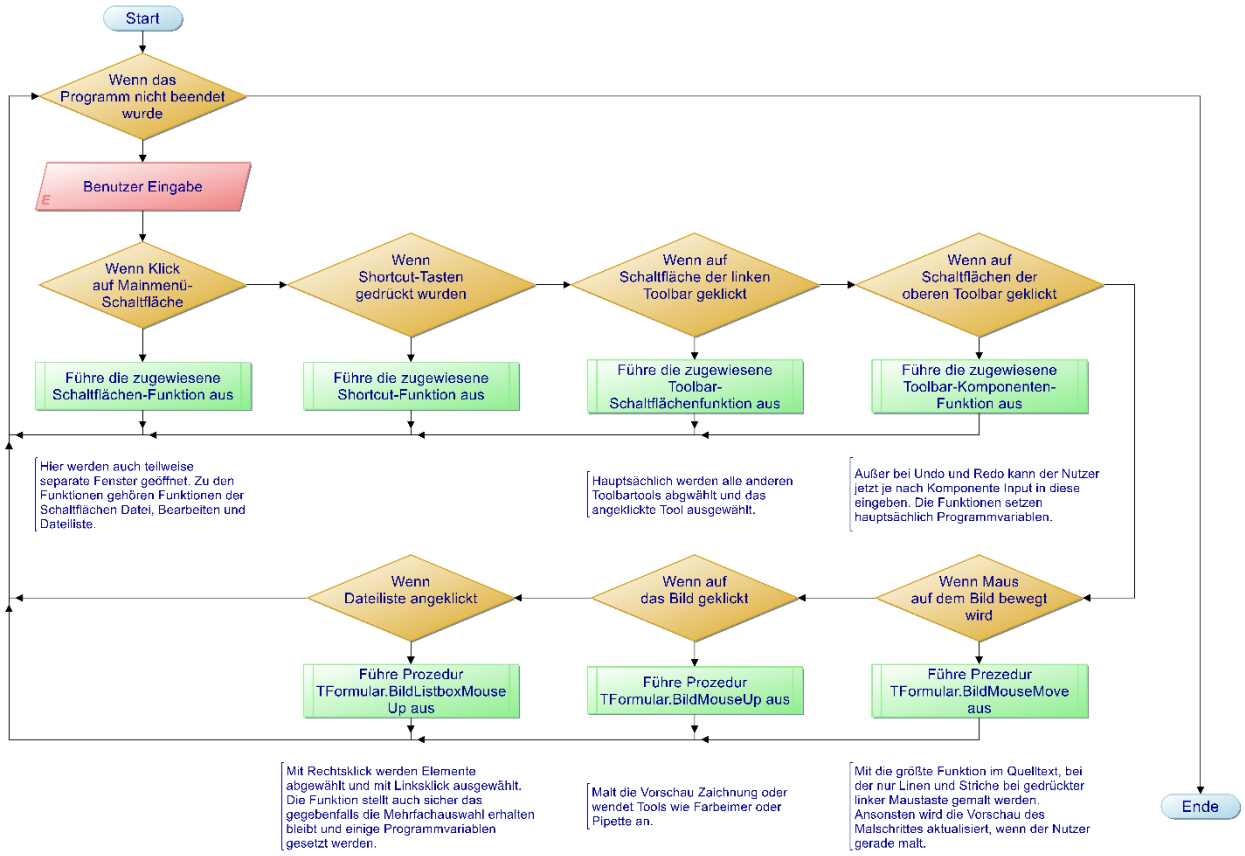


Abb. 9: Der Programmablaufplan für das gesamte Programm Bildeditor. Die Kommentare beschreiben die Funktionen genauer.

5 Testen des Programmes

5.1 Allgemeiner Programmtest

Ziel dieses Tests, ist es das gesamte Programm auf Fehler und Mängel zu prüfen. Dazu teste ich alle Programmteile möglichst in Kombination miteinander. Ich teste neben der Funktionalität auch die Benutzerfreundlichkeit.

Insgesamt sind in diesem Test nur folgende zwei Fehler aufgetreten:

- Wenn nach dem Klick auf „Alle Einträge löschen“ eine neue Bitmap mit „Neu“ erstellt wird, stürzt das Programm ab. Die Ursache dafür war, dass die Variable „CurrSelLBIndex“ nach Löschen der Elemente nicht auf -1 gesetzt wurde. Es wurde also am Ende ein Index außerhalb eines Arrays angewendet.
- Nach dem Öffnen einer Datei, wurde im Array „BmpisSavedArray“ der letzte Boolean auf True gesetzt, wodurch das unterste Element in der Dateiliste als schon gespeichert galt und beim Speichern nie gespeichert wurde.

Beide Fehler wurden behoben.

Neben Fehlern habe ich auch noch an diesen Stellen kleine Qualitätsmängel bemerkt:

- Der Nutzer kann das Programm einfach schließen ohne zu wichtige Dateien zu Speichern. Deshalb könnte noch eine Warnung bei nichtgespeicherten Dateien implementiert werden.
- Bei doppelten Dateinamen werden alle Dateien mit Doppelnamen nacheinander in eine Datei gespeichert, wobei die Datei, welche gerade gespeichert wird, die letzte überschreibt. Als Lösung im Falle eines doppelten Dateinamens könnte in der Dateiliste eine Zahl in Klammern angefügt werden.
- Dateien, welche geöffnet werden behalten den die Dateiendung im Namen und beim Speichern wird erneut eine Dateiendung an den Namen mit der alten Dateiendung angefügt. Hier könnte man im Falle einer Dateiendung die alte Endung einfach beim Speichern entfernen.

5.2 Skalierungstests

In diesem Test untersuche ich die Skalierung der Komponenten bei verschiedenen Auflösungen. Ich führe diesen Test durch, da ich häufig Probleme in der Skalierung bei niedriger Auflösung bemerkt habe. Ich vermute, dass bei diesem Test keine falschen Skalierungen mehr auftreten werden, da ich die Fehler schon während der Codierung behoben haben müsste.

Beim Testen bin ich von meiner ursprünglichen Auflösung von 3840 × 2100 Pixel auf 1920 × 1080 Pixel und auch einmal auf 1600 × 1200 Pixel gewechselt. Es skalierten alle Popupfenster, wie sie sollten aber nur die linke Toolbar war in der falschen Höhe platziert.

5.3 Leistungstests

Bei diesen Tests teste ich die Leistungsfähigkeit des Programms in Schnelligkeit und Arbeitsspeicherverbrauch. Der Grund dafür ist, dass man theoretisch tausende Bilder öffnen könnte und ein Absturz des Programms durch vollbelegten Arbeitsspeicher auftreten könnte. Dasselbe Problem würde bei der Programmvariable „Undoarray“ auftreten, die für jeden Malschritt eine neue Bitmap speichert. Ebenfalls überprüfe ich die Leistung beim Malen auf sehr großen Bitmaps.

Als erstes malte ich mehrmals auf einer 5000 × 5000 Pixel großen Bitmap und bemerkte, dass das Malen mit der Zeit länger dauerte. Der Grund dafür war das ständige Hinzufügen von Bitmaps in „Undoarray“: Das Hinzufügen von Bitmaps zu einem Array wird mit zunehmender Anzahl an Elementen sehr langsam. Ein Stack wäre hierfür der geeignete Datentyp. Jedoch beschränke ich aus Zeitgründen einfach die Länge des Arrays auf 15 Elemente, um eine größere Bildrate beizubehalten. Das allgemeine Malen auf der sehr großen Bitmap verlief schnell genug, war aber deutlich langsamer als auf normalgroßen Bitmaps.

Als zweites testete ich die Leistung des Programms bei dem Öffnen von sehr vielen Bilddateien. Dazu öffnete ich aus einem Testordner 3554 Bilder mit einer Größe von insgesamt 874 MB. Erstaunlicherweise öffneten alle der JPG-Dateien schon in etwa einer Minute und man konnte mit normaler Leistung weitermalen. Wie in Abbildung 11 zu sehen, wurde zuerst die CPU beansprucht, während die Bilder geladen haben, wobei der Arbeitsspeicher zunehmend belegt wurde. Ich öffnete dieselben Bilddateien noch zweimal und dann stürzte das Programm ab, da kein Arbeitsspeicher mehr zur Verfügung stand. Momentan ist hierfür noch keine Lösung implementiert, da die Lösung zu aufwendig und nicht sehr entscheidend für die Qualität des Programmes ist.

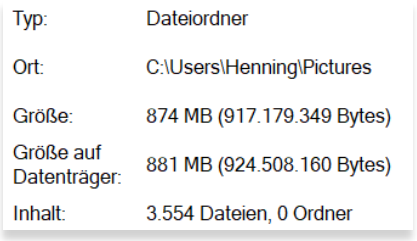


Abb. 10: Die Eigenschaften des Testordners mit den Testbildern

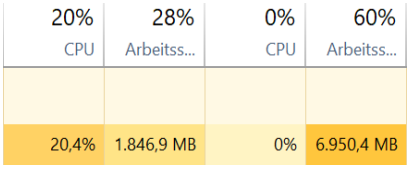


Abb. 11: Der Ressourcenverbrauch während des Ladens der Bilddateien (links) und nach Laden der Dateien (rechts)

6 Fazit

Insgesamt habe ich für diese komplexe Leistung ein sehr umfangreiches Programm geschrieben. Auch wenn an vielen Stellen am Programm noch weitergearbeitet werden könnte, denke ich, dass dieses Programm fertiggestellt ist: Bis auf die kleinen Qualitätsmängel im allgemeinen Nutzungstest, ist das Programm vollständig programmiert und es sind alle von mir geplanten Features umgesetzt. In das Programm könnte man natürlich noch wichtige Features, wie das Auswählen eines einiger Pixel oder das Arbeiten mit Ebenen implementieren. Aber der Umfang für dieses Programm würde dann einfach zu groß werden. Dennoch bin ich am Ende mit meinem Programm „Bildeditor“ sehr zufrieden und finde, dass ich mit der Erstellung des gesamten Projektes wirklich viel an wichtiger Erfahrung gewonnen habe. Das Dazulernen bei der Erstellung dieses Programmes war auch das, was dieses Projekt wirklich für mich ausmachte und warum ich mir so viel Mühe dafür gemacht habe.

7 Quellenverzeichnis

Vgl. Circular: LazPaint: in: Free Pascal, 11.01.2021 https://wiki.freepascal.org/LazPaint [12.06.2021]

Vgl. RGB -> TColor -> RGB: in: Delphi-Praxis, 13. Apr 2003, https://www.delphipraxis.net/4067-rgb-tcolor-rgb.html [12.06.2021]

Paint.NET: Wikipedia, 20.05.2021 https://de.wikipedia.org/wiki/Paint.NET [12.06.2021]

Downloads

Vgl. Aha-Soft: 32x32-free-design-icons: in: Small-Icons, http://www.small-icons.com/packs/32x32-free-design-icons.htm [12.06.2021]

Vgl. Circular: Release BGRABitmap v11.2.5, in: Github, https://github.com/bgrabitmap/bgrabitmap/releases/tag/v11.2.5 [12.06.2021]

Vgl. Lazarus: Lazarus Homepage, in: Lazarus https://www.lazarus-ide.org/ [12.06.2021]

Vgl. Paint.NET: Paint.NET – Download, in: Paint.NET, https://www.getpaint.net/download.html [12.06.2021]

8 Danksagung

Ich danke hiermit dem Autor der kostenlosen Icons, welche ich in meinem Programm für die meisten meiner Schaltflächen verwendet habe und hinterlege hierfür den Link zu aha-soft.com: http://www.aha-soft.com.

9 Anhang

Als Anhang sendete ich den Pascal-Sourcecode aller Units von meinem Projekt, sowie die Zip-Datei mit allen Projekt-Dateien.