Střední odborná škola a střední odborné učiliště Hustopeče, p. o.
Masarykovo nám. 136/1, 693 01 Hustopeče

Vývoj grafického softwaru

Závěrečná maturitní práce oboru
Informační technologie

Patrik Pacák
Autor práce

Mgr. Zdeněk Brokeš
Vedoucí práce

Střední odborná škola a střední odborné učiliště Hustopeče, p. o.
Masarykovo nám. 136/1, 693 01 Hustopeče

ŠSOŠ A SOU HUSTOPEČE, PŘÍSPĚVKOVÁ ORGANIZACE

Střední odborná škola a střední odborné učiliště Hustopeče, příspěvková organizace
Masarykovo nám. 136/1, 693 01 Hustopeče
www.sou-hustopece.cz

PŘIHLÁŠKA K MATURITNÍ PRÁCI

Školní rok: **2020/2021**

Obor: **Informační technologie**

Jméno a příjmení žáka:	Patrik Pacák
Oblast maturitní práce:	Hardware, operační systémy, sítě
Téma maturitní práce:	Vývoj grafického softwaru
Vedoucí maturitní práce:	Mgr. Zdeněk Brokeš
Náhradní vedoucí práce:	Ing. Jiří Tinka
Oponent maturitní práce:	Ing. Oldřiška Nováková
Náhradní oponent MP	Ing. et Bc. Štěpán Pavelka

Rozsah maturitní práce: min. 34 200 znaků + přílohy

Software bude obsahovat:
- Volný výběr, výběr čtvercem, - Lupa/Zoom, - Vrstvy (export gif souboru), - Přidání textu,barva,font a velikost
- Ořez , - zmenšení / zvětšení - možnost roztáhnout vertikálně, horizontálně, oběma směry
- Možnost ukládání (+ uložit jako)/otevírání formátů: jpg, jpeg, bmp, gif, png
- Možnost exportu do pdf, - otočení ve vertikální i horizontální rovině
- grafický efekt-> černo-bílá, libovolná Barva - bílá, libovolná Barva - černá.-oblast muže být zvolena výběrem
- moznost změny DPI - oblast muže být zvolena výběrem, - výstup na tiskárnu
- editace: štětec - výběr barvy a tloušťky a tvaru
- možnost volby pravidelných tvarů (kruh, elipsa, čtverec, obdélník,čára)
- možnost volby průhlednosti / krytí - oblast muže být zvolena výběrem
- možnost vyplnit plochu barvou -oblast muže být zvolena výběrem

Způsob odevzdání maturitní práce:

2 x v papírové podobě v nerozebiratelné formě a 2x v elektronické podobě uloženo na CD/DVD

V Hustopečích dne: 2.10.2020

podpis žáka

podpis vedoucího maturitní práce

podpis ředitele školy

Pozn. Náhradní vedoucí práce nebo náhradní oponent - v případě vážného důvodu zastupuje ved. MP nebo oponenta MP

Střední odborná škola a střední odborné učiliště Hustopeče, p. o.
Masarykovo nám. 136/1, 693 01 Hustopeče

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu práce
Mgr. Zdeňku Brokešovi za pomoc při zpracování mé závěrečné práce.

Prohlášení

Prohlašuji, že jsem závěrečnou práci zpracoval samostatně a použil jen prameny použité v seznamu literatury.

V Hustopečích dne _____

OBSAH

1 ÚVOD

V své práci se budu zabývat designovou a programovací stránkou vytváření softwaru na úpravu obrázku a GIFů. Jako první se podíváme na cíl práce a na výchozí podmínky. Poté přejdeme k teoretickému rozboru a odtud se odrazíme k třídám a funkcím. Skončíme závěrem a tím, co jsem si z práce odnesl.

1.1 Cíl

Cílem práce bylo vytvořit software, který půjde použít k úpravě jakýchkoliv obrázků a GIFů. Zároveň bylo cílem vytvořit přívětivé a přehledné grafické rozhraní. Hlavní prioritou bylo, aby byl editor jednoduchý na použití, přehledný a rychlý. Důležitá byla absence instalačního balíčku, aby práce se softwarem byla ještě rychlejší. Celý program jsem vytvořil v anglickém jazyce, aby ho mohl používat kdokoliv.

1.2 Prvotní plán

1.2.1 Design

Už od začátku jsem měl v hlavě několik designových verzí. Jedna z prvních verzí byly klasické ovládací prvky na levé, či pravé straně hlavního okna, které používá většina grafických softwarů. Chtěl jsem, aby byl můj program něčím originální, aby to nebyla jen napodobenina jiného známějšího softwaru, a proto jsem zvolil ovládací prvky ve spodní části hlavního okna a spoustu vedlejších oken, které se aktivují po kliknutí na ovládací prvek korespondující k danému oknu. Výsledkem je to, že pracovní plocha zůstane čistá a většina informací se zobrazuje na vedlejších oknech.

1.2.2 Harmonogram

Na každou funkci jsem si stanovil datum, do kterého bych měl tuto funkci dokončit. Záleželo na složitosti problému a často se stávalo, že jsem špatně odhadl čas. Programování je totiž jeden z oborů, u kterého je těžké odhadnout časovou náročnost problému. Nakonec mi můj vedoucí práce Mgr. Zdeněk Brokeš díky svým zkušenostem pomohl vytvořit plán, ve kterém mělo vše určeno své datum dokončení. Tento plán byl velmi důležitý, protože mi ukázal přibližnou časovou náročnost a tím mě motivoval.

1.2.3 Návaznost

Měl jsem možnost navázat na svůj minulý projekt z druhého ročníku, který obsahoval spoustu prvků podobných těm v této práci. Rozhodl jsem se v tomto projektu nepokračovat, protože obsahoval některé chyby a kód by byl velmi neuspořádaný a neoptimalizovaný.

1.3 Softwarové vybavení

1.3.1 Vývojové prostředí

Hlavní byla jednoduchost programu a jeho modernost. Proto jsem se rozhodl použít Visual Studio 2019, konkrétně zdarma verzi „Comunity 2019“ s nainstalovaným balíčkem na vývoj desktopových aplikací s architektonickým uživatelským rozhraním WPF, což jsou programovací jazyk C# a značkovací jazyk XAML.

1.3.2 Visual Studio

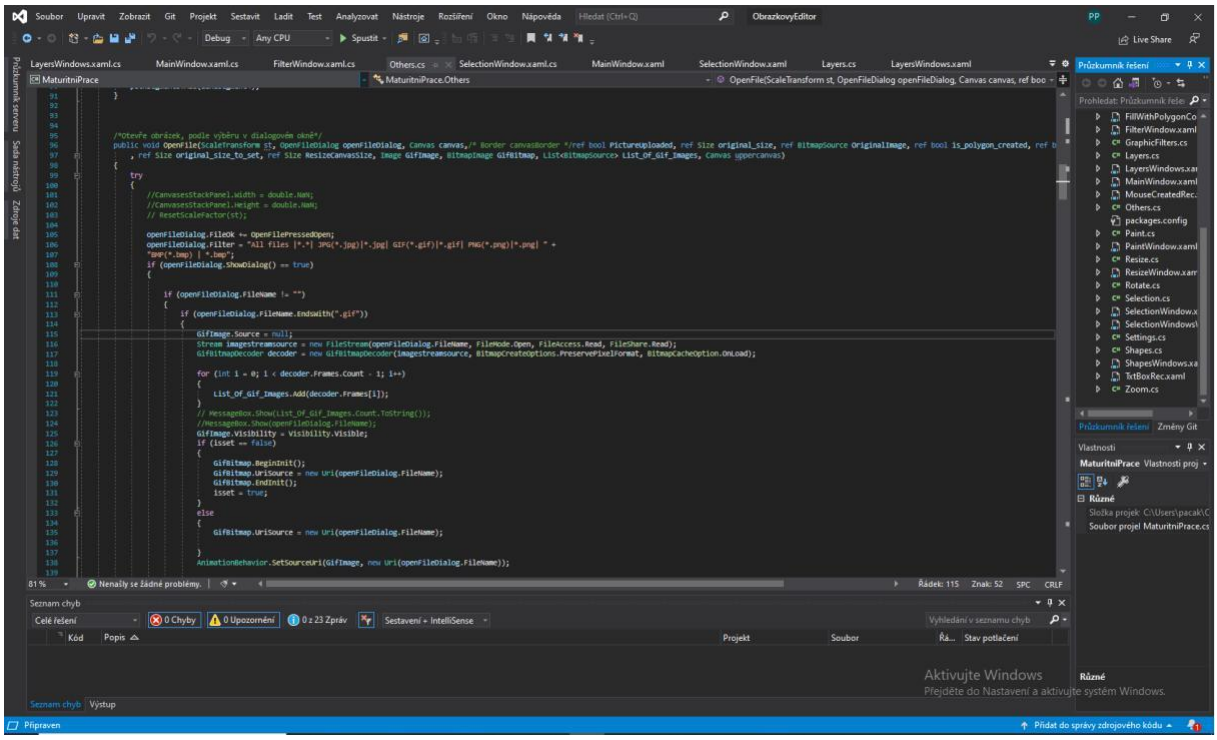
Visual Studio je sada nástrojů a služeb pro vývojáře aplikací využívající programovací jazyky jako jsou C#,C++, J Sharp a Visual Basic. Vychází každé 2 roky a vždy nabízí 3 typy licencí.

Community, je bezplatná licence a je určena pro začínající vývojáře a indie studia. Její použití je zakázáno v případě, že firma používající tento software má roční výnos větší než 1 milion USD nebo více jak 250 počítačů s tímto softwarem.

Další licencí je verze Professional, která nenabízí příliš novinek od verze Community, ale její použití je volné a není nijak omezeno.

Poslední verzí je Enterprise, která přináší spousty výhod od verze Professional a její použití také není nijak omezeno.

První verzí bylo Visual Studio 97, dnes je nejnovější Visual Studio 2019.



Obr. 1: Vývojové prostředí Visual Studio Community 2019; zdrojový kód přímo z mojí aplikace

1.3.3 .NET Framework

Jedná se o soubor technologií, který umožňuje lepší komunikaci aplikací a zároveň vytváří stabilní základnu pro jejich tvorbu.

Předchůdcem byl COM (Component Object Model), který se snažil propojit aplikace s různými rozhraními. Tento model se projevil nedostatečný, a proto ho v roce 2002 nahradil. NET Framework. Nejaktuálnější verzí je 4.7.2 a připravuje se verze 4.8. Tato verze má být také poslední a nástupcem bude. NET 5.

1.4 Architektura uživatelského rozhraní

Vybrat jsem si mohl mezi WPF a WinForms. Měl jsem už zkušenosti s WinForms, které jsem použil při mojí ročníkové práci ve druhém ročníku. Bohužel WinForms jsou zastaralé. Mají slabé vykreslovací schopnosti a téměř všechny procesy jsou pomalejší, než totožné ve WPF.

Samozřejmě nejsou WinForms tak špatné a mají i své klady. Například jsou jednodušší na pochopení a pro začátečníky jsou přívětivější. WPF jsou těžší na naučení a navíc k nim patří značkovací jazyk XAML, na rozdíl od WinForms, které fungují stylem Drag-And-Drop a používají čistě jazyk C#.

1.4.1 WPF (Windows Presentation Foundation)

WPF je nástupcem už zastaralých WinForms. Toto rozhraní je součástí. NET Framework, což je soubor technologií obsahující například právě WPF, nebo také ASP .NET a WinForms. Toto rozhraní přišlo v roce 2006 s verzí .NET Framework 3.0.

Grafickou částí tohoto rozhraní se zabývá značkovací jazyk XAML a o práci s daty se stará C# (CSharp).

1.5 Programovací jazyky

1.5.1 C#(CSharp)

Je to programovací jazyk z rodiny jazyků typu C (Dále např. C++,C, Javascript apod.). Má velmi blízko k programovacímu jazyku Java, ale půjčuje si spoustu funkcí od jazyků C a C++.

Na trh byl uveden spolu s .NET v roce 2002 a dnes se jedná o jeden z nejčastěji používaných programovacích jazyků, hlavně kvůli jeho tzv. „Garbage Collector“, který usnadňuje spoustu práce. Využívá hlavně OOP (Object Oriented Programing), který uživateli pomáhá vytvářet třídy a objekty.

```
else if (removedpoint == false && IsStopped == false)
{
    draw = true;
    listOfPoints.Add(p);
    Ellipse ellipse = new Ellipse();

    Line line = new Line();
    line.Stroke = new SolidColorBrush(Colors.Yellow);
    line.X1 = listOfPoints[listOfPoints.Count - 1].X;
    line.Y1 = listOfPoints[listOfPoints.Count - 1].Y;
    line.X2 = line.X1;
    line.Y2 = line.Y1;
    ellipse.Width = 5;
    ellipse.Height = 5;
    ellipse.Fill = new SolidColorBrush(Colors.Orange);
    listOfElipses.Add(ellipse);
    listOfLines.Add(line);
    Canvas.SetLeft(ellipse, listOfPoints[listOfPoints.Count - 1].X - 2);
    Canvas.SetTop(ellipse, listOfPoints[listOfPoints.Count - 1].Y - 2);
    canvas.Children.Add(ellipse);
    canvas.Children.Add(line);
    TextBlock textBox = new TextBlock();
    textBox.Width = 100;
    textBox.Text = Convert.ToString(String.Format("{0};{1}", Math.Round(listOfPoints[listOfPoints.Count - 1].X) * ratio, Math.Round(listOfPoints[listOfPoints.Count - 1].Y) * ratio));
    selectionWindowMenu.textboxstackpanel.Children.Add(textBox);

}
```

Obr. 2: Příklad C# kódu; část z mého zdrojového kódu

1.5.2 XAML

XAML je zkratka pro Extensible Application Markup Language. Je založený na XML a používá se hlavně na návrh uživatelského prostředí ve WPF a Silverlight. Slouží hlavně k usnadnění zápisu, protože vše, co můžeme napsat pomocí XAML, tak lze zapsat i pomocí C# nebo Visual Basic. XAML tedy vlastně dává příkazy programovacímu jazyku, jaké objekty by měl vytvořit na jaké pozici a jaké vlastnosti by měly mít. Kdybychom měli to stejné, co napíšeme v XAML napsat v programovacím jazyku, zápis by byl mnohem složitější a delší.

První verze tohoto jazyka vyšla v roce 2007 a poslední v roce 2010 s názvem v 2009.

Obr. 3: Příklad XAML kódu; část z mého zdrojového kódu.

2 ROZBOR ZÁKLADNÍCH PRINCIPŮ

2.1 Otevírání obrázků o různých velikostech a formátech

Jelikož se jedná o editor obrázků, dobře fungující import musel být samozřejmostí. Podporuje formáty typu *.jpg, *.bmp, *.gif a *.png. Při výběru je možná filtrace vybraných formátů. U formátu .png podporuje alfa kanál, a tak se pozadí obrázku zobrazuje transparentně.

Formát typu .gif se zobrazuje animovaně a po otevření je rozložen na vrstvy. K tomu, aby se mi GIFy bezproblémově otevíraly a přehrávaly, jsem použil XamlAnimatedGif (knihovna specializovaná na přehrávání GIFů). Pro export GIFů jsem použil Ngif (knihovna specializovaná na export GIFů). Obě tyto knihovny nejsou implementovány v základním balíčku pro vývoj desktopových aplikací ve WPF a musel jsem je doinstalovat jako NuGet balíčky pomocí rozhraní pro instalaci NuGet balíčků ve Visual Studiu. Balíčky jsou dostupné zde:

https://github.com/XamlAnimatedGif/XamlAnimatedGif

https://www.codeproject.com/Articles/11505/NGif-Animated-GIF-Encoder-for-NET

Pokud je otevřený obrázek pixelově větší, než pracovní plocha, jeho velikost se přizpůsobí velikosti hlavního okna a jeho originální velikost je vzata v potaz při ukládání (záleží na podobě jeho úpravy).

2.2 Ukládání a Export

Ukládání bylo velmi důležité, protože je to ve své podstatě výsledek mojí práce. Program by měl být schopný obrázek upravit, ale k čemu by byla úprava, kdyby jej nešlo uložit? Proto jsem na této funkci pracoval už od začátku a v průběhu programování jsem ji upravoval a přidával další možnosti.

Ukládat/Exportovat se dá jak obyčejné obrázky, tak obrázky ve formátech typu .png s alfa kanálem. Lze exportovat i formáty typu .gif, které se uloží jako animovaný sled obrázků.

Možnosti, jak uložit obrázek jsou dvě. Buď uživatel klikne na tlačítko Save nebo Save As. Pokud zvolí Save, obrázek se uloží jako kopie s přízviskem „-EditorCopy“ ve složce, ve které se nachází původní obrázek. Pokud dojde ke stejnému uložení znovu, přepíše se tento obrázek za nový.

Druhou možností je uložení pomocí Save As. Umožňuje vybrat cestu, kam obrázek uložit, formát a jeho název.

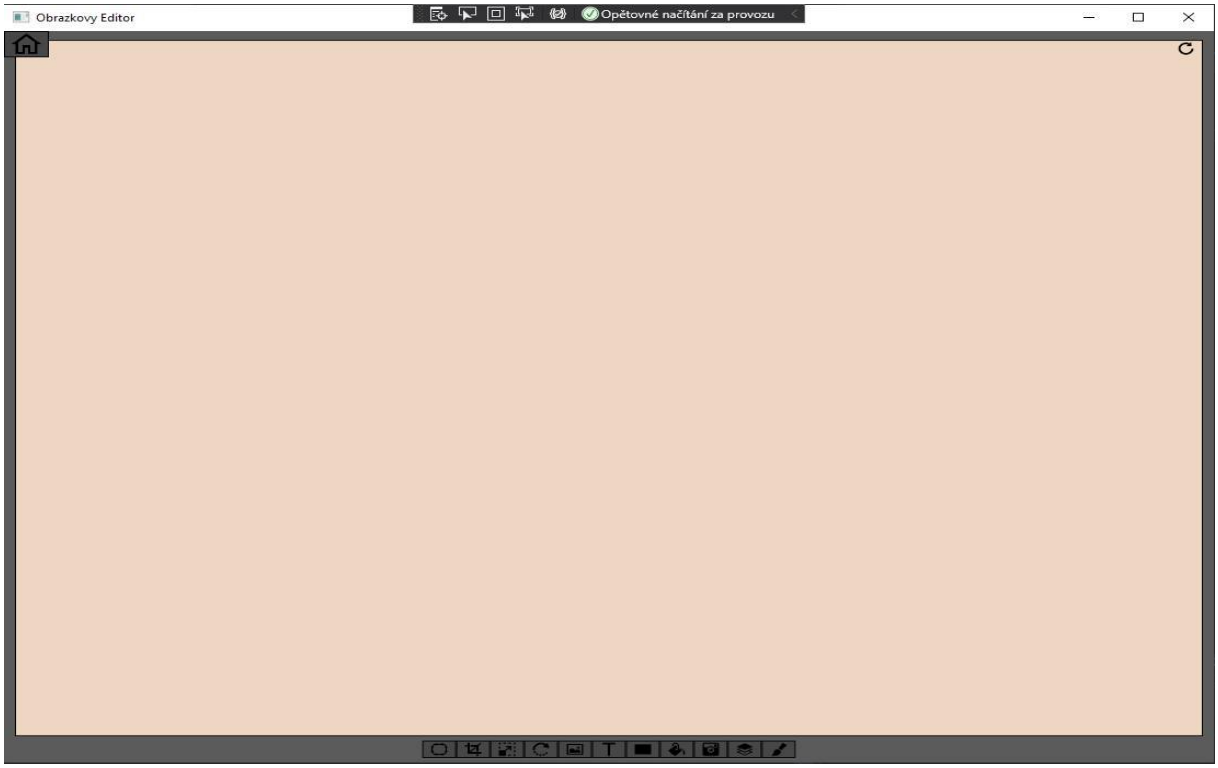
2.3 Hlavní okno

Design a rozpoložení hlavního okna byly velmi důležité aspekty, vzhledem k tomu, že je to plocha, na kterou se uživatel dívá nejvíce. Vlevo nahoře se nachází ikona domu, která po kliknutí rozbalí ze strany vysouvací menu s možnostmi Open, Save, Save As, Settings a Print. Pokud se uživatel rozhodne, že nechce být v menu, stačí kliknout na pracovní plochu a menu se opět zabalí zpátky.

V pravém horním rohu se nachází tlačítko Reset. Po jeho kliknutí se otevřený obrázek na pracovní ploše vrátí do podoby, ve které byl při otevření.

V dolní části hlavního okna se nachází, jak já tomu říkám „Tool Belt“, což je sada nástrojů, určená k úpravě obrázků. Po kliknutí na jeden z těchto nástrojů se otevře vedlejší okno korespondující k danému nástroji.

Ve středu hlavního okna se nachází pracovní plocha. Na této ploše se otevírají obrázky. Po každém otevření se obrázek přizpůsobí pracovní ploše svou velikostí.



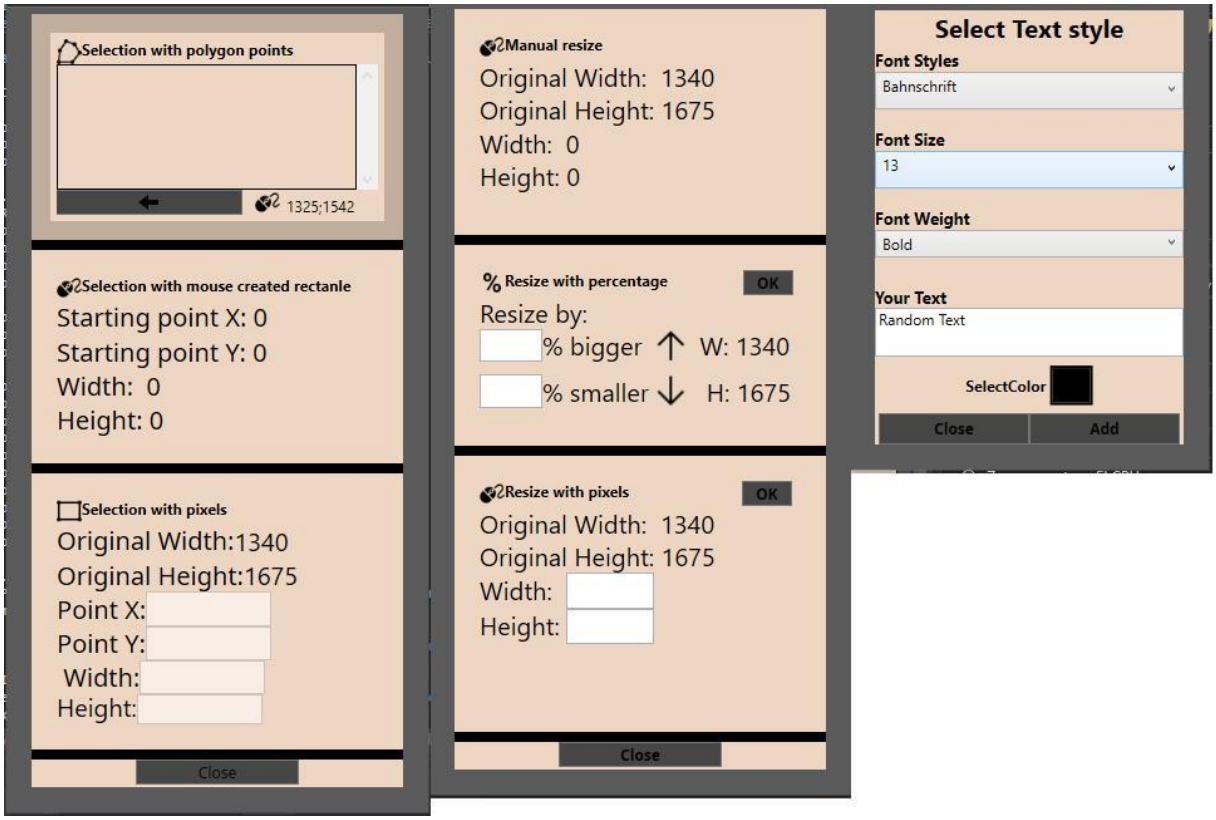
Obr. 4: Hlavní okno aplikace; screenshot z mé aplikace

2.4 Vedlejší okna

Udělat individuální okna pro každý nástroj bylo velmi velké rozhodnutí, a proto jsem s ním trochu váhal, než jsem ho opravdu implementoval. Před finální implementací přišlo mnoho změn a předělávek.

Téměř vždy je možné mít otevřeno jen jedno vedlejší okno najednou. Jen v málo případech je možné mít otevřené více než jedno okno, například u nástrojů filtr a vyplnění barvou, které pracují s nástrojem výběru.

Hlavní okno a jakékoliv vedlejší okno se spolu vždy vlezou na monitor, kvůli automatickému nastavení velikosti hlavního okna při startu.



Obr. 5: Příklady vedlejších oken; screenshot z mé aplikace

2.5 Vzájemná kompatibilita mezi funkcemi

Zajistit vzájemnou kompatibilitu všech oken bylo zásadní, protože po každé úpravě obrázku by musel uživatel buď program vypnout, nebo resetovat a to by bylo velmi nepraktické. Proto je většina oken naprogramovaných tak, aby byly schopné pracovat i s úpravou z předchozího okna.

Jedna z cest, jak vyřešit problém vzájemné kompatibility funkcí, byla možnost, že se vzájemná kompatibilita vůbec neuskuteční a uživatel bude moci pracovat jen s jednou funkcí. Toto řešení by nebylo příliš praktické a ve finále by to způsobovalo akorát nechuť s programem pracovat.

Obr. 6: Porovnání originálního obrázku proti obrázku, který byl oříznut, a následně se do něj štětcem kreslilo; screenshot z mé aplikace

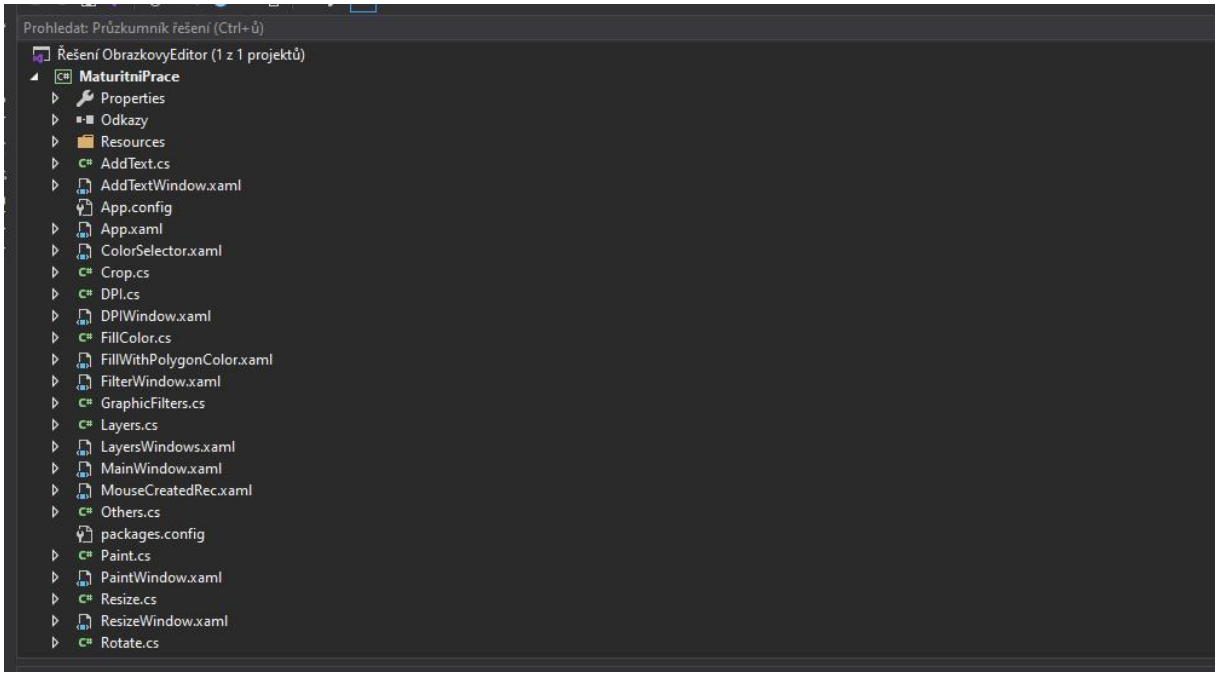
3 FUNKCE A TŘÍDY

3.1 Rozdělení do tříd

Každý implementovaný nástroj má ve finálním řešení svoji třídu, ve které se nachází korespondující metody a proměnné. Tyto třídy se zabývají pouze a jen záležitostmi daného nástroje a jejich volání probíhá v hlavním okně, v kterém se nachází i jejich objekty. Každá třída má svůj vlastní soubor s příponou .*cs.

Kdybych tento styl organizace kódu nepoužil a vyskytla by se nějaká chyba, mohl bych mít velké problémy najít její původ, vzhledem k tomu, že kód by byl jako bludiště a nedalo by se v něm vyznat a orientovat.

Občas jsem se setkal se situací, kdy jsem musel použít metody z jedné vedlejší třídy ve druhé vedlejší třídě, ale snažil jsem se tomuto zamotávání kódu co nejvíce vyhnout. Bohužel pokud jsem chtěl dosáhnout vzájemné kompatibility některých funkcí, jiná cesta nebyla k dispozici.

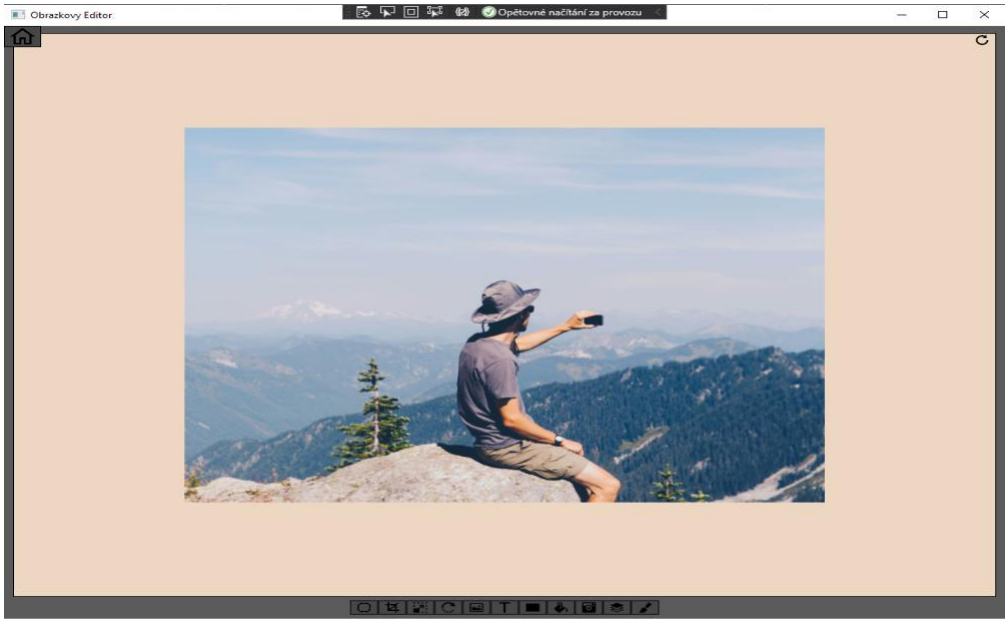


Obr. 7: Rozdělení do tříd (Nejsou zde všechny); screenshot z Visual Studia

3.2 Úprava velikosti obrázku při jeho otevření

Při otevření obrázku se spustí systém, který změní velikost obrázku tak, aby se vlezl do pracovní plochy. Toho je docíleno pomocí algoritmu, který porovnává rozměry originální pixelové velikosti obrázku a rozměry pracovní plochy. Pokud ze začátku velikost nesedí do okna, obrázek se zmenší o malou část a porovnání proběhne znovu. Takto probíhá porovnávání, dokud se obrázek nevleze. Poté se vydělí originální velikost s novou velikostí a dostaneme tak, o kolik je obrázek na pracovní ploše menší. S tímto poměrem dále pracují vedlejší okna nebo procesy ukládání.

Další možností, jak by šlo obrázek zobrazit při otevření, bylo, že by se zobrazil ve své původní velikosti a uživatel by poté lupou obrázek oddálil. Ve skutečnosti je implementované řešení mnohem praktičtější, protože u většiny otevřených obrázků, první co uživatel udělá, tak chce vidět obrázek celý.



Obr. 8: Příklad otevřeného obrázku, který by se za svých původních rozměrů nevlezl na pracovní plochu; screenshot z mé aplikace

3.3 Přizpůsobení hlavního okna na monitor

Přizpůsobení hlavního okna bylo velkým nedostatkem u práce z druhého ročníku. Aplikace byla při otevření příliš velká buď šířkově, nebo výškově. Tentokrát jsem se na tento aspekt zaměřil a udělal jej správně.

Po otevření aplikace se spustí algoritmus, který upraví velikost hlavního okna tak, aby se vlezlo na dané zobrazovací medium jak hlavní okno, tak minimálně jedno vedlejší. Neměla by tedy nastat situace, že by se okna prolínala a pokud ano, tak jen minimálně, ale s tím se v dnešní době už nepočítá, vzhledem k tomu, že dnešní monitory mají většinou rozlišení v poměru 16:9.

3.4 Ukládání a Export

Pro ukládání obrázků jsem použil knihovnu System.IO, zabývající se importem a exportem různých druhů souborů.

Systém ukládání a export obrázků je vysvětlen v rozboru, proto se podívejme na export a ukládání GIFů. Při otevření GIFu dojde k jeho rozdělení na vrstvy, které se následně nahrají do listu datového typu Bitmap. Tento list je poté upravován podle toho, jak uživatel vrstvy upravuje. Pokud dojde ke zvolení exportu.

3.5 Hlavní Okno

Hlavní okno zde slouží jako základna pro celou aplikaci a většina algoritmů se spouští právě přes kód, který stojí za tímto oknem. Téměř všechny třídy nástrojů mají své objekty zde, kde se s nimi následně pracuje, a volají se metody dané třídy. Také se zde nachází valná většina použitých proměnných.

Okno lze libovolně zmenšovat, či zvětšovat podle potřeby tak, aby byla práce pro uživatele co nejkomfortnější.

3.6 Funkce a jejich vedlejší okna

3.6.1 Výběr

- **Knihovny:**
  System.Windows System
  Windows.Controls
  System.Windows.Media,
  System.Windows.Shapes
  System
  Systm.Collections.Generic
  System.Windows.Input
- **Vlastní okno:**
  Ano
- **Popis:**

Nástroj výběru je rozdělen na tři menší nástroje a záleží na uživateli, který si vybere.

Prvním nástrojem je volný výběr pomocí bodů. Aktivovatelný je tím, že ve vedlejším okně uživatel klikne na lehce zabarvený rám, který obtéká daný výběr. Následně může vytvářet body na obrázku. Tyto vytvořené body (jejich souřadnice) se zobrazují ve vedlejším okně. Zobrazuje se zde i aktuální pozice kurzoru na obrázku. Výběr lze přerušit kliknutím na poslední vytvořený bod a následně v něm znovu pokračovat opětovným kliknutím na poslední bod. Výběr lze dokončit kliknutím na první vytvořený bod. Pokud uživatel vytvoří bod někde, kde by nechtěl, lze poslední vytvořený bod odstranit.

Po dokončení výběru se souřadnice nahrají do listu s datovým typem Point, s kterým se dále pracuje.

Druhým nástrojem je výběr pomocí myší vytvořeného čtverce. Výběr tohoto nástroje je stejný jako u minulého výběru, tentokrát ale uživatel vytváří pomocí tahu myši čtverec. Vedlejší okno zobrazuje počáteční bod, na kterém byl čtverec vytvořen a rozměr čtverce.

Třetím výběrem je výběr pomocí souřadnic. Zde se uživateli ukáží původní rozměry obrázku. Výběr začne tím, že zadá souřadnici X a Y bodu, na kterém by měl čtverec začít. Poté už stačí jen zadat šířku a výšku a tvar se dokončí.

U obou čtvercových výběrů se po dokončení nahrají rozměry a pozice čtverce do proměnné datového typu Rectangle, se kterou se dále pracuje.

Pokud si uživatel rozmyslí výběr nástroje, kliknutím na jiný typ výběru se resetuje výběr, ale je možné použít jinou metodu.

3.6.2 Ořez

- **Knihovny:**

  System.Windows

  System.Windows.Controls

  System.Windows.Media

  System.Windows.Shapes,System

  System.Collections.Generic

  System.Windows.Media.Imaging

- **Vlastní okno:**

  Ano

- **Popis:**

Ořez bylo docela jednoduché implementovat, což se projevilo i na tom, že nemá své vedlejší okno.

Lze jej použít pouze, pokud je vybrána oblast výběrem. Jeho algoritmus funguje tak, že po každém dokončení výběru, se oblast výběru přenese do třídy ořezu. U volného výběru je to datový typ list <Point> a u ostatních Rectangle.

Pokud byl proveden volný výběr pomocí bodů, převedou se body z listu na datový typ LineSegment, který určuje ohraničení obrázku. Tak se ořízne obrázek podle stanovených souřadnic.

Pokud byl proveden čtvercový výběr, provede se pomocí metody CroppedBitmap oříznutí s rozměry proměnné datového typu Rectangle.

3.6.3 Změna velikosti

- **Knihovny:**

  System.Windows

  System.Windows.Controls

  System.Windows.Media

  System.Windows

  Shapes.System.Windows.Input

  System
- **Vlastní okno:**

  Ano
- **Popis:**

Změna velikosti je stejně jako výběr, rozdělena do tří nástrojů. Zvolení nástroje je stejné jako u výběru.

Prvním nástrojem je ruční změna velikosti. Uživatel má možnost obrázek ručně roztáhnout tak, že kurzorem myši zatáhne za jakoukoliv stranu obrázku. Vytvoří se červený čtverec, který ukazuje budoucí velikosti obrázku. Ve vedlejším okně se zobrazuje originální velikost a nová velikost.

Druhým nástrojem je změna velikosti pomocí procent. Po zvolení tohoto nástroje se na pracovní ploše objeví dva obrázky. Levý obrázek je ten upravovaný a pravý ten originální. V podstatě se zde ukazuje, v jakém poměru bude obrázek zvětšen nebo zmenšen. Po kliknutí na šipky se změní velikost
o 10 %. Vedle šipek se zobrazuje aktuální výška a šířka. Aby nastala změna velikosti, je nutné, aby uživatel klikl Ok.

Třetím nástrojem je změna velikosti s užitím číselného údaje. K dispozici jsou dvě pole, do kterých lze zadat šířku a výšku. Nad nimi se

zobrazuje originální velikost. Změnu je opět nutné potvrdit tlačítkem Ok, jinak se neprojeví.

3.6.4 Rotace

- **Knihovny:**

  System.Windows

  System.Windows.Media

  System.Windows.Media.Imaging

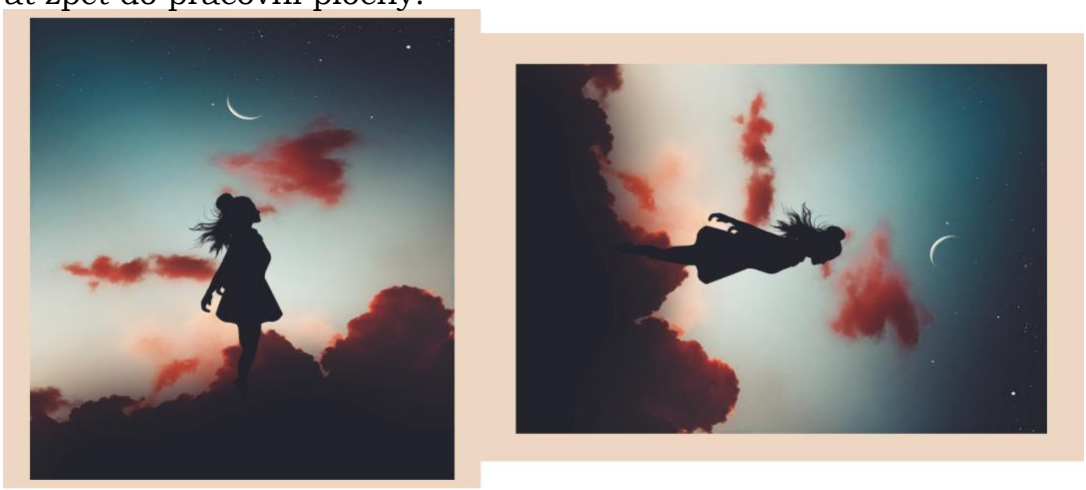
  System.Windows.Shapes

  System.Windows.Controls
- **Vlastní okno:**

  Ne
- **Popis:**

Rotace stejně jako výběr nemá své vedlejší okno, protože by zde bylo docela zbytečné. Stačí jen kliknout na nástroj v panelu a obrázek se otočí o 90 stupňů doprava.

Rotace probíhá za pomoci třídy TransformedBitmap, která mi umožňuje pomocí třídy RenderTransform otočit obrázek a následně jej nahrát zpět do pracovní plochy.



Obr. 9: Příklad otočeného obrázku pomocí nástroje Rotace; screenshot z mé aplikace

3.6.5 Změna DPI

- **Knihovny:**

  System.Windows

  System.Windows.Controls

  System.Windows.Media

  System.Windows.Media.Imaging

  System.Windows.Shapes

  System.Collections.Generic

- **Vlastní okno:**

  Ano

- **Popis:**

Tento nástroj ovládá tzv. „dots per inch“ v obrázku, neboli počet pixelů na jeden palec. Má velice jednoduché okno. Obsahuje dvě šipky, jedna je zvětšení a druhá zmenšení DPI. Funkce zvětšení DPI funguje jen potom, co uživatel DPI zmenší. Druhá šipka slouží pro zmenšení. Volba se potvrdí tlačítkem Close. Pro změnu DPI jsem použil opět třídu TransformedBitmap, tentokrát ale v kooperaci s třídou ScaleTransform.

3.6.6 Výběr barvy

- **Knihovny:**

  System

  System.Windows

  System.Windows.Input

  System.Windows.Media

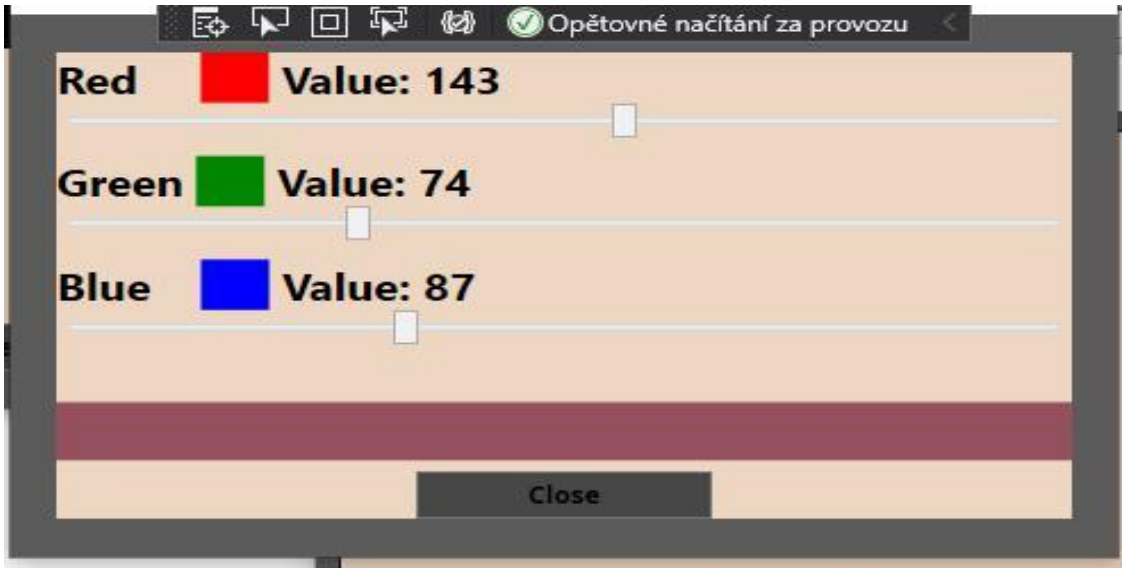
- **Vlastní okno:**

  Ano

- **Popis:**

Výběr barvy sice není jeden z hlavních nástrojů v hlavní nabídce, ale je stejně důležitý. Používá jej Vložení textu, Vložení tvarů, Vyplnění barvou a filtry. Objeví se pokaždé, když uživatel zmáčkne na vybarvený čtverec, ať už se nachází kdekoliv.

Má své vlastní vedlejší okno. Funguje na principu RGB, uživatel si posuvníky vybere hodnotu červené, zelené a modré. Finální mix barev se zobrazuje na spodní straně okna.



Obr. 10: Okno výběru barvy; screenshot z mé aplikace

3.6.7 Vložení textu

- **Knihovny:**

  System.Windows

  System.Windows.Shapes

  System.Windows.Media

  System.Windows.Controls

  System.Windows.Input

  System.IO
- **Vlastní okno:**

  Ano
- **Popis:**

Tento nástroj obsahuje spousty možností úpravy a vložení textu. Nabízí změnu fontu, velikosti textu, jeho váhu, obsah a také barvu pomocí výběru barvy. Ve vedlejším okně se nastavují všechny parametry textu. Vzorový text se vloží automaticky po otevření vedlejšího okna a budou se mu přidávat všechny změněné atributy. Text lze také přesouvat tahem myši na místo zvolené uživatelem. Pokud je uživatel spokojen s umístěním, stačí zmáčknout Add a text se přidá do obrázku. Následně už jej nelze upravovat.

U tohoto nástroje jsem pracoval hlavně s třídami RenderTargetBitmap, DrawingVisual, DrawingContext a Formatted Text. Na začátku se vytvoří prázdný objekt typu RenderTargetBitmap. Poté DrawingVisual, který je možnost vyrenderovat na objekt RendeTargetBitmap. Následně musí být vytvořen DrawingContext, který umožní samotné kreslení na DrawingVisual a pomocí právě DrawingContext se vykreslí samotný obrázek a text (Formatted Text). Následně se DrawingVisual vygeneruje na RenderTargetBitmap a tento formát už lze jednoduše konvertovat na formát, který se dá zobrazit v pracovní ploše.

3.6.8 Vložení tvarů

- **Knihovny:**

  System

  System.Windows

  System.Collections.Generic

  System.Windows.Controls,

  System.Windows.Input

  System.Windows.Media

  System.Windows.Media.Imaging

  System.Windows.Shapes

  System.IO

- **Vlastní okno:**

  Ano

- **Popis:**

Nabízeny jsou tři základní tvary. Čtverec, obdélník a kruh. Ve vedlejším okně se nachází tyto tři tvary a po kliknutí na jeden z nich se objeví na obrázku. Lze s nimi pohybovat tahem myši a jejich velikost lze měnit pomocí kolečka myši. Potvrdit přidání lze pomocí tlačítka Add. Opět lze zvolit barvu pomocí okna výběru barvy.

Prvně jsem měl v plánu udělat ve vedlejším okně samostatnou kreslící plochu, ve které by si uživatel mohl nakreslit svůj vlastní tvar. Tuto možnost jsem po zvážení zrušil, protože vyplnění oblasti barvou vykonává tutéž činnost.

Algoritmus vložení tvarů pracuje se stejnými třídami jako vložení textu, ale s jinými parametry.

3.6.9 Vyplnění oblasti barvou

- **Knihovny:**

  System

  System.Windows

  System.Windows.Controls

  System.Windows.Input

  System.Windows.Media

  System.Windows.Media.Imaging

  System.Windows.Shapes

  System.IO

- **Vlastní okno:**

  Ano

- **Popis:**

Vyplnění barvou úzce pracuje s výběrem. V korespondujícím okně nalezneme výběr barvy a průhlednosti. V případě zvolení výběru pomocí bodů se oblast vykreslí poté, co bude výběr dokončen. Pokud si uživatel zvolí výběr pomocí čtverce, oblast se vyplní po dokončení čtverce tahem myši. Pokud by chtěl uživatel nastavit průhlednost vybrané oblasti, může tak učinit posuvníkem, který se nachází po výběrem barvy. Stisknutím tlačítka Add se vyplněná oblast přidá na obrázek.

Tento nástroj funguje na podobném principu jako přidání textu a využívá téměř stejné třídy, až na přidávání oblast výběrem. Třída DrawingContext disponuje metodou DrawGeometry, ale jejím argumentem není proměnná typu polygon, ale typu StreamGeometry. Je tedy potřeba převést list všech bodů polygonu na tento datový typ. Poté jej už lze vykreslit na DrawingVisual.

3.6.10 Filtry

- **Knihovny:**

  System

  System.Windows

  System.Windows.Controls

  System.Windows.Input

  System.Windows.Media

  System.Windows.Media.Imaging

  System.Windows.Shapes

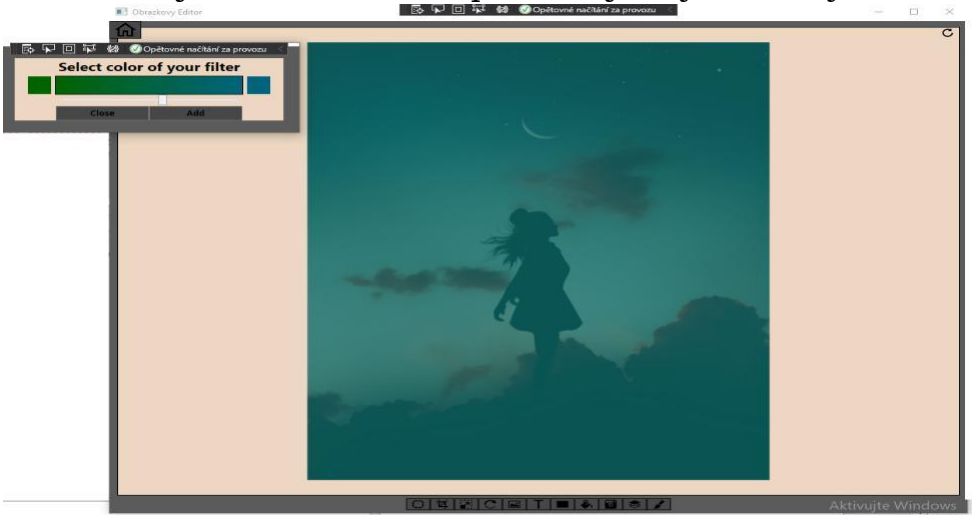
  System.IO

- **Vlastní okno:**

  Ano

- **Popis:**

Tento nástroj funguje na podobném principu jako vyplnění barvou, akorát se nevyplňuje barvou jen určitá plocha, ale celý obrázek. Obrázek se celý vyplní dvěma překrývajícími se barevnými čtverci, které dohromady dají daný filtr. Uživatel si může nastavit, jakou kombinaci barev by přesně chtěl tím, že klikne na jeden ze čtverců reprezentující výběr barvy.



Obr. 11: Příklad použití filtru; screenshot z mé aplikace

3.6.11 Tisk

- **Knihovny:**

  System.Windows.Controls

  System.Windows.Media

  System.Windows.Media.Imaging

  System.Windows.Documents

- **Vlastní okno:**

  Ano

- **Popis:**

K tisku se uživatel dostane v hlavním menu, je to úplně poslední položka. Po stisknutí se zobrazí tiskový dialog, ve kterém je možnost si vybrat tiskárnu, počet kopií atd.

Setkal jsem se zde s problémem, který mi zamezoval vytisknout obrázky s větší šířkou na papír ve stejných proporcích. Toto se poté vyřešilo použitím tříd TransformedBitmap a RotateTransform, pomocí kterých jsem obrázek mohl otočit tak, aby se vytisknul správně. Pokud je obrázek proporcemi na výšku, obrázek se neotočí a tisk proběhne rovnou.

3.6.12 Štětec

- **Knihovny:**

  System

  System.Windows

  System.Windows.Shapes

  System.Windows.Controls

  System.Windows.Media

  System.Windows.Input

- **Vlastní okno:**

  Ano

- **Popis:**

Štětec byla jedna z nejméně problémových funkcí na implementaci. Princip je jednoduchý. Při každém pohybu myší se spustí metoda, která vykreslí zvolený tvar na obrázek na pozici kurzoru.

Pokud uživatel není spokojený s výsledkem kreslení, má možnost tlačítkem Clear vše odstranit a začít od začátku. Tlačítkem Close se kreslení potvrdí a proběhne vykreslení tvarů stejně jako v nástroji Vložení tvarů, akorát ve větším měřítku, protože se musí každý tvar vykreslit samostatně.

3.6.13 Vrstvy

- **Knihovny:**

  System.Collections.Generic

  System.Windows.Shapes

  System.Windows

  System.Windows.Media

  Microsoft.Win32

  System.Windows.Media.Imaging

  System.Windows.Controls

- **Vlastní okno:**

  Ano

- **Popis:**

Při otevření se automaticky otevře vedlejší okno se snímky z daného GIFu. Jednotlivé snímky se zobrazí v okně a je s nimi možno dále pracovat.

Algoritmus otevírání spočívá v použití třídy GifBitmapDecoder, umožňující rozdělit otevřený GIF na jednotlivé vrstvy. Tyto vrstvy se poté nahrají do vedlejšího okna.

Možností úprav je hned několik. Uživatel má možnost jednotlivé snímky smazat kliknutím na červený křížek vedle dané vrstvy. Dále je možnost snímky jednotlivě upravovat jako samostatné obrázky. Daná vrstva se poté chová jako obyčejný obrázek a po dokončení její úpravy se nahraje zpátky do vrstev. Další možností je upravit počet snímků, které se zobrazí za jednu vteřinu.

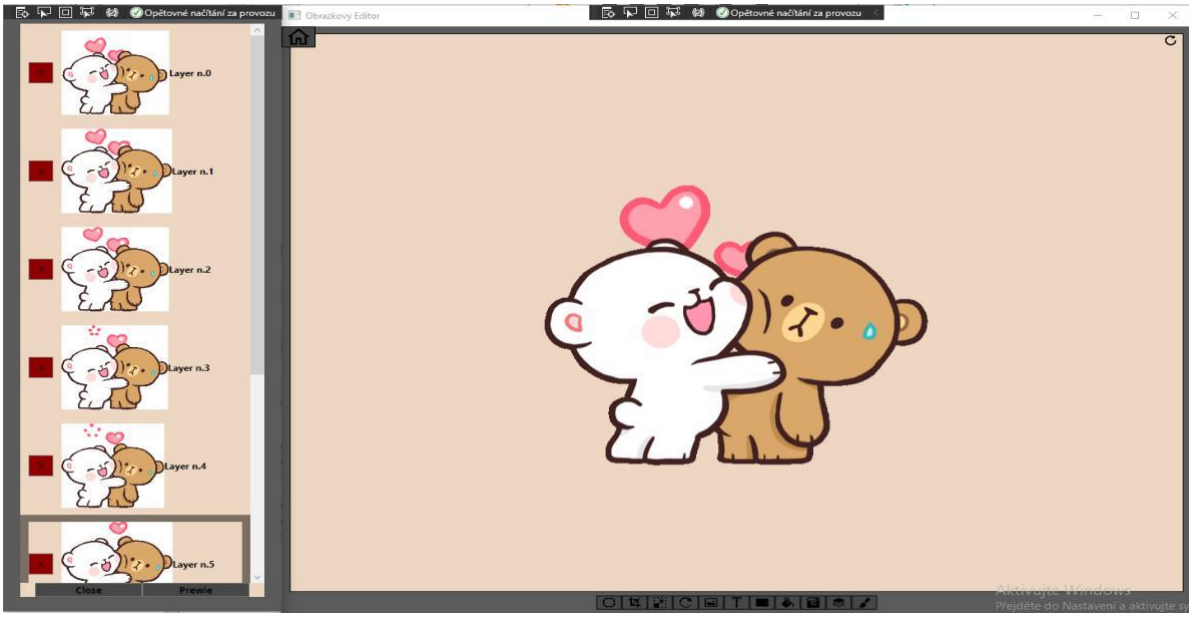
Setkal jsem se zde s problémem, který velmi prodlužoval dobu úpravy daného GIFu. Jednalo se o to, že pokud uživatel chtěl například přidat do každého obrázku text, musel udělat totéž pro každý snímek. Bylo by to velmi

zdlouhavé a nepraktické, a proto jsem to vyřešil možností ,,Apply to all frames“, která umožňuje příští úpravu přidat na všechny snímky najednou.

Pokud by uživatel chtěl vidět výsledek své úpravy, může se na ni podívat před uložením a to tlačítkem Preview. Výsledek se zobrazí na pracovní ploše.

Pokud je uživatel plně spokojen a je připravený GIF exportovat, stačí stisknout v menu Uložit jako. Po otevření se aplikace zeptá, jestli chce uživatel opravdu GIF animovat. Pokud ne, uloží se jen první snímek ve formátu png.
V opačném případě se provede export do místa zvoleného uživatelem.

Pro přehrávání jsem použil knihovnu XamlAnimatedGif a pro export Ngif. Obě mi umožnily mnohem jednodušší práci, a kdybych měl to, co dělají ony dělat sám, asi bych program včas nedokončil.



Obr. 12: Podoba okna vrstev, pár dní před dokončením; screenshot z mé aplikace

3.6.14 Zoom

- **Knihovny:**

  System.Windows

  System.Windows.Input

  Systém.Windows.Media

  System.Windows.Media.Imaging

  System.Collections.Generic

  System.Windows.Shapes

  System.Windows.Controls

- **Vlastní okno:**

  Ne

- **Popis:**

Zoom funguje pomocí kolečka na myši. Posunutím kolečka nahoru se obrázek přiblíží a naopak dolů, se obrázek oddálí. Zoom funguje na celý obrázek najednou.

Aktuální je v aplikaci druhá verze zoomu. Tato verze pracuje s celým obrázkem a nelze přibližovat na určité sektory. První verzí byl zoom, který fungoval přibližováním na určité sektory podle toho, kde se nacházel kurzor myši. Prvotní myšlenka byla dobrá, ale byla špatně implementována. Sektorové přiblížení bylo uděláno tak, že se obrázek přibližoval v prostoru daného obrázku, ale celý obrázek se nepřibližoval. Toto působilo velmi neprakticky a nefungovalo to stoprocentně. Proto byla tato verze nahrazena.

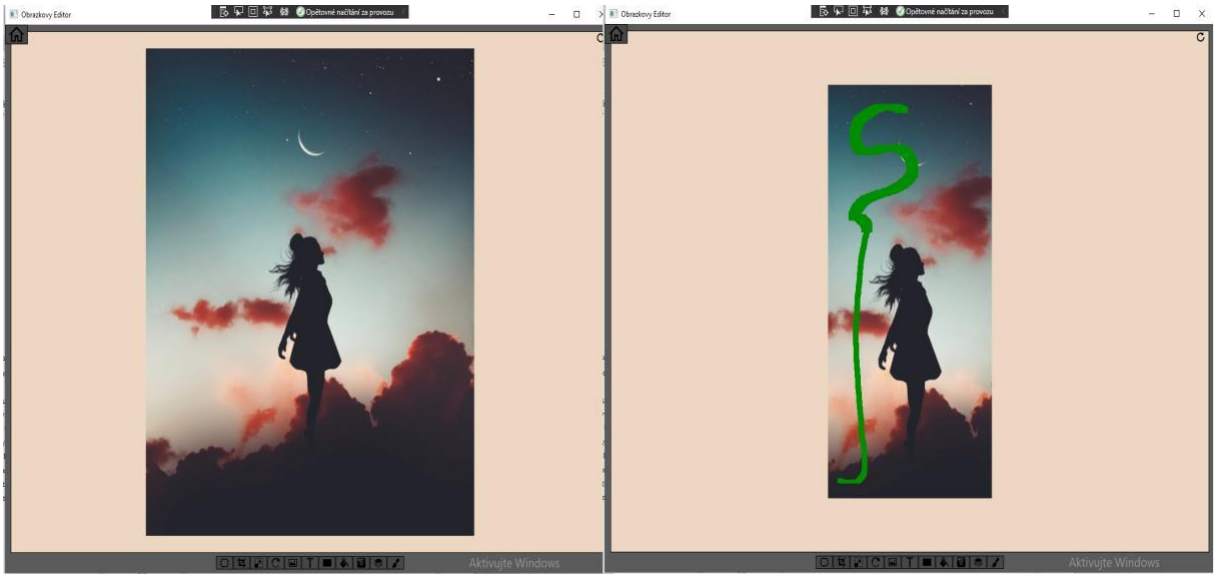
Nejnovější verze pracuje s třídami TransformedBitmap a ScaleTransform.

3.6.15 Reset

Tlačítko Reset slouží k načtení původního obrázku a odstranění všech změn.

Po stisknutí se zavřou všechna otevřená okna, vyčistí se všechny listy a proměnné se nastaví na jejich původní hodnoty. Dále se odstraní obrázek na pracovní ploše a nahradí jej ten původní.

Restart se dá provést také klávesovou zkratkou CTRL + Z.



Obr. 13: Obrázek po resetu a před ním; screenshot z mé aplikace

4 NÁPADY DO BUDOUCNA

4.1 Settings

Původně jsem měl plán vytvořit nastavení, ve kterém by si uživatel mohl přizpůsobit aplikaci podle sebe. Měly zde být možnosti zvolit si jiné barvy, jiné rozpoložení nástrojového panelu a přidávání klávesových zkratek. Bohužel na tuto funkci nevyšel čas.

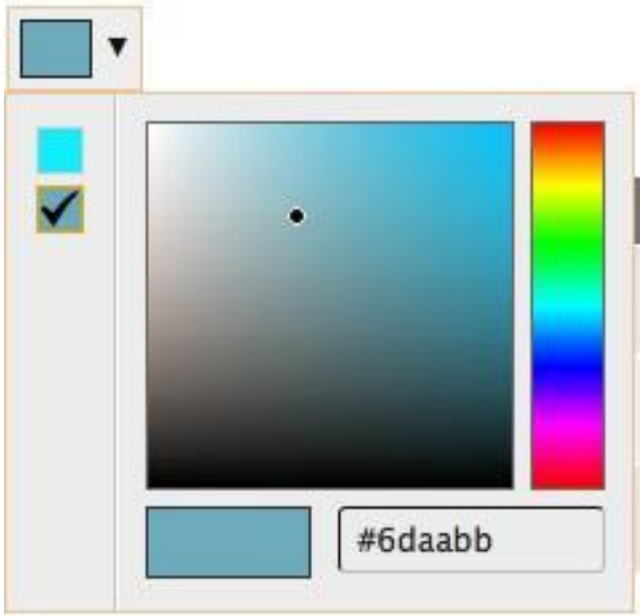
4.2 Složitější filtry

Původně měly filtry pracovat na bázi úpravy jednotlivých pixelů, aby se docílilo pravé úpravy jako u filtrů nejpoužívanějších softwarů. Bohužel úprava jednotlivých pixelů je velmi náročná a bylo mi jasné, že pokud bych se do toho pustil, zasekl bych se na mrtvém bodě, protože už jsem se snažil upravovat pixely, když jsem vytvářel první verzi okna výběru barvy. Tento pokus, jak už jsem zmiňoval, dopadl neúspěšně.

4.3 Výběr barvy barevným spektrem

Před aktuální verzí výběru barvy pomocí RGB posuvníků jsem pracoval na provedení, které by využívalo obrázek barevného spektra. Uživatel by si vybral kliknutím myší na místo v obrázku barvu.

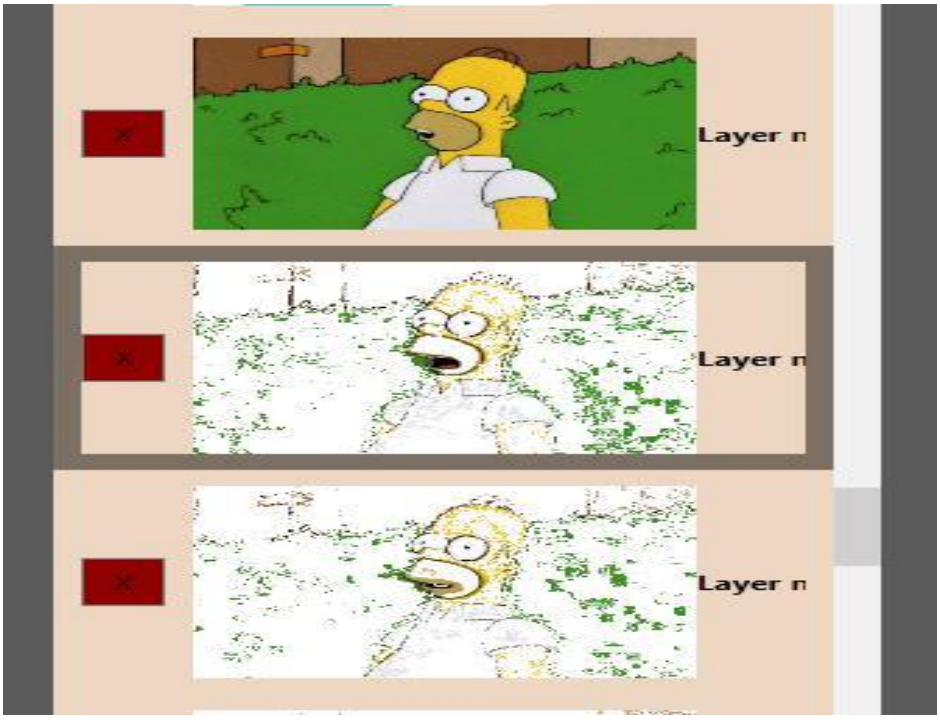
Barva by se vybrala získáním hodnot RGB daného pixelu, nad kterým by se zrovna nacházel kurzor. Po několika dnech programování jsem měl vytvořené okno, ale nefungovalo mi správně získávání RGB hodnot pixelů. Pokaždé se mi objevila barva pixelu, který se nacházel na úplně jiném místě, než na které jsem klikl. Nakonec jsem tuto myšlenku hodil do koše a pustil se do aktuální verze.



Obr. 14: Příklad toho, jak mohl vypadat výběr barvy pomocí barevného spektra; https://www.drupal.org/project/color_field

4.4 Animované GIFy

Prvně bylo v plánu udělat plně funkční editor GIFů, který otevře všechny druhy tohoto formátu. Bohužel v průběhu práce na vrstvách jsem zjistil, že dnešní GIFy už nejsou jen pole obrázků jdoucí za sebou, ale jsou animované. To znamená, že obrázky nejsou všechny plně vykreslené, ale kde je to potřeba, dokresluje se jen změna z minulého obrázku z důvodu zkrácení načítání při otevření a exportu. Pokročilejší softwary už si dokáží animované snímky dokreslit a obecně si s tímto problémem poradit. Nakonec, po vyzkoušení mnoha a mnoha knihoven/tříd jsem si řekl, že to nechám spíše do budoucna.



Obr. 15: Příklad animovaného GIFu, který částečně dokresluje jen pohyblivé části nových snímků; screenshot z mé aplikace

5 ZÁVĚR

I když zde nejsou všechny funkce, program jsem úspěšně dokončil. Odnesl jsem si spoustu nových zkušeností s jazykem C# a strukturováním jeho kódu. Naučil jsem se, že se lze zaseknout i na jednoduchých principech, pokud k nim člověk nepřistupuje správně a naopak, že i některé složité problémy můžou mít jednoduché východisko. Zopakoval jsem si spoustu učiva z hodin programování a potvrdilo se mi to, že v tomto oboru bych chtěl i nadále v životě pokračovat. Zároveň mám spousty nových poznatků k formátování dokumentů a práci se zdroji a s přílohami. Jedna z nejdůležitějších věcí, kterou si z tohoto projektu budu odnášet do života, je schopnost ubránit se prokrastinaci.

Použití programu se představuji tak, že by tento program měl uživatel v počítači a čas od času by jej zapnul, kdyby potřeboval rychle a snadno udělat nenáročnou úpravu obrázku.

V budoucnu by určitě mohl být program upraven o funkce, které se mi nepovedly a také i o nějaké pokročilejší. Dále by mohlo být upraveno grafické rozhraní, aby vypadalo profesionálněji.

6 ZDROJE

1) *Co je WPF (Windows Presentation Foundation)? - Visual Studio | Microsoft Docs. [online]. Copyright © Microsoft 2021 [cit. 22. 03. 2021]. Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/designers/getting-started-with-wpf?view=vs-2019*

2) What is .NET? An open-source developer platform.. .NET | Free. Cross-platform. Open Source. [online]. Dostupné z: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet

3) *Kapitola 1. .NET framework [online]. [cit. 2021-03-28]. Dostupné z: http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch01.html*

4) *Lekce 1 - Visual Studio - Úvod do vývojového prostředí. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2021 itnetwork.cz. Veškerý obsah webu [cit. 22. 03. 2021]. Dostupné z: https://www.itnetwork.cz/csharp/visual-studio/tutorial-visual-studio-uvod*

5) BĚHÁLEK, Marek. [online]. [cit. 2021-03-28]. Dostupné z: http://www.cs.vsb.cz/behalek/vyuka/pcsharp/index.php

6) *Jazyk XAML. Největší český web zaměřený na .NET framework [online]. Copyright © 2021 [cit. 22. 03. 2021]. Dostupné z: https://www.dotnetportal.cz/clanek/198/Jazyk-XAML*

7) *CLARK, Dan. Beginning C# Object-Oriented Programing. Springer-Verlag Berlin and Heidelberg GmbH & Co. ISBN 1430235306.*

8) VIRIUS, Miroslav. C# pro zelenáče. Praha: Neocortex, c2002. Bestseller for all. ISBN 80-863-3011-7.

7 SEZNAM OBRÁZKŮ

8 SEZNAM OBRÁZKŮ POUŽITÝCH V PŘÍKLADECH