Střední průmyslová škola a Vyšší odborná škola, Liberec, příspěvková organizace

VANGOGH PAINTER

Maturitní práce

Autor	**Jan Holý**
Obor	**Informační technologie**
Vedoucí práce	**Ing. Tomáš Kazda DiS.**
Školní rok	**2024/2025**
Počet stran	**29**
Počet slov	**4763**

SPŠSE A VOŠ LIBEREC

Přihláška k maturitní práci

Jméno a příjmení studenta
Holý, Jan

Třída
P4A

Název práce
VanGogh Painter

Školní rok
MP2024/25

Přidělené role

Vedoucí práce

Podpis ..... Kazda, Tomáš

Oponent

Podpis ..... Smutný, Jan

Obecná ustanovení	Vypracování a odevzdání práce proběhne v souladu s platnými normami (vyhláška 177/2009 Sb.) a aktuálním dokumentem "Pokyny k vypracování prací" vydaným školou.
	Práce bude hodnocena z hlediska jejího praktického využití, zvládnutí dokumentace po věcné i formální stránce a obhajoby celé práce. Student byl seznámen s kritérii hodnocení maturitní práce.
	Práce bude odevzdána ve dvou stejnopisech vázaných pevnou nebo kroužkovou vazbou.
	Veškeré náklady na MP včetně vyhotovení obou tištěných kopií si student hradí sám.
Licenční ujednání	Ve smyslu § 60 (Školní dílo) autorského zákona č. 121/2000 Sb. poskytuji SPŠSE a VOŠ Liberec výhradní a neomezená práva k využití této mé maturitní práce.
	Bez svolení školy se zdržím jakéhokoliv komerčního využití mé práce.
	Pro výukové účely a prezentaci školy se vzdávám nároku na odměnu za užití díla.

Finanční rozvaha - odhad celkových nákladů

V Kč	Náklady celkem	Hrazené školou
Výrobní	0	0
Na služby	0	0

Jedná se o MP, jejíž vypracování si škola vyžádala? **Ano** – **Ne**

Podpis studenta (vyjadřuje souhlas s uvedenými údaji a ujednáními)

V Liberci 01.10.2024 Podpis .....

Konzultant

Práci podporuji Podpis .....

Předmětová komise

Práci doporučuji Podpis .....

Třídní učitel

Práci doporučuji Podpis .....

Garant oboru

Práci doporučuji Podpis .....

Ředitel školy

Práci doporučuji Podpis .....

Zadání maturitní práce

Název

VanGogh Painter

Předmět

WEB, MPA, MME

Téma

Cílem práce je vytvoření webové aplikace pro komunikaci s kreslícím robotem VanGogh. V prostředí budem možné vytvářet jednoduché obrazce (např. React Sketch Canvas), importovat SVG data, nastavovat měřítko kreslení a sledovat proces vykreslování.

Použité prostředky

React, TypeScript, HTML, CSS

Cíle práce

1	Knihovna a editor SVG v Canvas
2	Realizace rozhraní UI/UX
3	Komunikace s VanGogh prostřednictvím BLE API
4	Zobrazení cesty robota na webu

Osnova práce

1	Katalog funkčních a nefunkčních požadavků
2	Design UI/UX (Figma)
3	BLE API, komunikační rozhraní
4	Práce s grafikou cesty (path), změna měřítka
5	Záznam z testování, úpravy

Anotace

Práce se zabývá spojením VanGogh Painter robota s webovým rozhraním. Vychází z nekompletního a uživatelsky neideálního projektu. Přináší nové webové rozhraní s intuitivním rozhraním a vylepšení pro pohodlnější a širší možnosti použití, širší veřejnosti.

Summary

This work deals with connecting the VanGogh Painter robot with a web interface. It is based on an incomplete and user-ideal project. It brings a new web interface with an intuitive interface and enhancements for more convenient and broader possibilities of use to a wider audience.

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní práci vypracoval sám a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne 13.03.2025 .....

Jan Holý

Obsah

Úvod ... 1
1 Technologie a prostředí ... 2
1.1 WebStorm ... 2
1.2 React ... 2
1.3 Micro:bit Web Bluetooth ... 2
1.3.1 Micro:bit Web Components ... 2
1.4 SVG Editor ... 3
1.5 UART ... 3
1.6 VanGogh Extension ... 3
1.7 Figma ... 4
1.8 Převod z SVG do Turtle Graphics ... 4
2 Webová aplikace ... 6
2.1 UI/UX ... 6
2.2 Mechaniky ... 7
2.2.1 Připojení robota k webu ... 7
2.2.2 Protokol komunikace ... 8
2.2.3 Převod tvarů na cesty ... 10
2.2.4 Zobrazení cesty robota v reálném čase ... 10
2.2.5 Škálování ... 12
2.2.6 Editor SVG souborů ... 12
2.3 Popis použití ... 13
3 Problémy ... 14
3.1 Bluetooth připojení ... 14
3.2 Editor SVG ... 14
3.3 Práce s SVG ... 15
4 Testování ... 17
Závěr ... 18
Seznam zkratek a odborných výrazů ... 19
Seznam obrázků ... 21
Použité zdroje ... 22
A. Seznam přiložených souborů ... I

Úvod

Cíl této práce je vytvořit webovou aplikaci pro VanGogh Painter robota. Web je schopen přeložit SVG soubory na Turtle Graphics pokyny, kterým může robot rozumět. Také je přímo v aplikaci možnost jednoduchých úprav daného SVG souboru a škálování reálného výstupu podle potřeby pomocí pravítka, které ukazuje jeho reálnou velikost. Aplikace je uživatelsky přívětivá a jednoduchá na použití. Průběh vykreslení je graficky znázorněn zobrazením celé cesty robota a zvýrazněním již vypracované části. Celá aplikace byla nejprve navrhnuta v programu Figma a poté naprogramována převážně v TypeScript Reactu v prostředí WebStorm.

Práce byla vytvořena převážně za účelem přilákání nových zájemců o programování a zajímavé využití micro:bitu.

1 Technologie a prostředí

Práce je vypracována v TypeScript React v prostředí WebStorm od společnosti JetBrains. K spojení a komunikaci byla použita knihovna Micro:bit Web Bluetooth a její rozšíření Micro:bit Web Components. Pro úpravu SVG souborů byl použit editor vytvořen podle předlohy a upraven pro účely projektu. Design byl navrhnut v aplikaci Figma.

1.1 WebStorm

WebStorm je výkonné integrované vývojové prostředí (IDE) vyvinuté společností JetBrains, které je optimalizované pro vývoj v JavaScriptu a souvisejících technologiích, včetně Reactu. Nabízí pokročilé funkce jako automatické doplňování kódu, integrovanou podporu pro verzovací systémy a další užitečné nástroje, což vývojářům usnadňuje práci a zvyšuje jejich produktivitu. (1)

1.2 React

React je open-source JavaScriptová knihovna vyvinutá společností Meta (dříve Facebook) pro tvorbu uživatelských rozhraní. Používá komponentově orientovaný přístup, což umožňuje vývojářům vytvářet znovupoužitelné prvky aplikace. Díky využití virtuálního DOM je React velmi efektivní při aktualizaci a vykreslování uživatelského rozhraní, čímž zvyšuje výkon webových aplikací. (2)

1.3 Micro:bit Web Bluetooth

Micro:bit Web Bluetooth je technologie umožňující bezdrátovou komunikaci mezi zařízením micro:bit a webovou aplikací pomocí protokolu Bluetooth Low Energy (BLE). Díky této technologii mohou vývojáři vytvářet interaktivní webové aplikace, které komunikují s micro:bitem přímo v prohlížeči bez potřeby instalace dodatečného softwaru. To otevírá možnosti pro vzdělávací a IoT aplikace s jednoduchou konektivitou. (3)

1.3.1 Micro:bit Web Components

Micro:bit Web Components je sada webových komponent navržených pro usnadnění integrace funkcionality micro:bit do webových aplikací. Komponenty umožňují snadné ovládání micro:bitu, přístup k jeho senzorům a interakci s externím hardwarem prostřednictvím standardních webových technologií. Díky použití Web Components mohou vývojáři jednoduše přidávat micro:bit funkce do svých projektů bez nutnosti hlubšího porozumění komunikaci s hardwarem. (4)

1.4 SVG Editor

Použitý webový SVG editor je jednoduchý nástroj pro tvorbu a úpravu vektorové grafiky přímo v internetovém prohlížeči. Umožňuje uživatelům vytvářet základní tvary intuitivním způsobem bez nutnosti instalace specializovaného softwaru. Editor podporuje základní funkce jako kreslení čar a tvarů, přičemž umožňuje jejich přesné umístění. Díky tomuto editoru lze přímo na webové aplikaci tvořit a upravovat vektorovou grafiku přímo v prohlížeči s možností stažení upraveného souboru pro další využití. (5)

1.5 UART

UART je komunikační protokol, který umožňuje výměnu dat mezi počítačem a jinými zařízeními, například Micro:bitem, prostřednictvím **Bluetooth Low Energy (BLE)**. Tento způsob komunikace funguje podobně jako klasický **sériový port**, ale využívá bezdrátové spojení. (6) (7) (8)

Pomocí služby UART může webová aplikace **odesílat textové zprávy** do Micro:bitu a zároveň **přijímat odpovědi**, které informují o průběhu zpracování příkazů. Data jsou posílána v textové podobě, což umožňuje snadnou interpretaci a ladění. Tento způsob komunikace je klíčový například pro ovládání robota, kdy aplikace předává Micro:bitu instrukce k pohybu a kreslení. (6) (7) (8)

Díky UART službě je možné navázat **obousměrnou komunikaci**, což znamená, že nejen webová aplikace může posílat příkazy Micro:bitu, ale Micro:bit může zároveň posílat **stavové zprávy** zpět do aplikace. Tímto způsobem lze sledovat průběh vykonávaných operací, což je potřebné při vykreslování cesty robota a její průběh v reálném čase. (6) (7) (8)

1.6 VanGogh Extension

Micro:bit VanGogh Extension je programová část VanGogh robota, která umožňuje robotovi práci s kresbou a grafikou. Rozšíření poskytuje nástroje pro ovládání micro:bitu a umožňuje mu pohyb podle zadaných souřadnic a zvedat a pokládat pero. Je vhodné pro vzdělávací účely a podporuje kreativní programování. (9)

1.7 Figma

Figma je cloudový nástroj pro návrh uživatelského rozhraní, který umožňuje týmovou spolupráci v reálném čase. Díky svému intuitivnímu rozhraní a široké škále funkcí, jako jsou vektorové kreslení, komponenty a automatizované layouty, je ideální pro návrh moderních webových a mobilních aplikací. Figma umožňuje snadné sdílení návrhů a zpětnou vazbu, což usnadňuje iterativní vývoj a zlepšuje efektivitu designového procesu. (10)

1.8 Převod z SVG do Turtle Graphics

Převod vektorových příkazů z formátu SVG na Turtle Graphics umožňuje interpretovat složité grafické tvary jako sekvenci příkazů pro robota. Tento proces zahrnuje analýzu jednotlivých SVG příkazů a jejich konverzi na příslušné instrukce, které robot dokáže provést. (11) (12)

Prvním krokem v procesu převodu je analýza řetězce obsahujícího SVG příkazy. Každý příkaz v tomto formátu odpovídá určité operaci, například přesunu na nové souřadnice (*M*), vykreslení čáry (*L*) nebo kreslení křivky (*C, Q*). Skript nejprve rozdělí vstupní řetězec na jednotlivé segmenty, kde každému segmentu odpovídá konkrétní část cesty. (11) (12)

Po rozdělení řetězce se analyzuje každý segment a určují se příslušné parametry. Například příkaz „M“ znamená, že robot se má přesunout na určité souřadnice, zatímco příkaz „L“ značí vykreslení čáry mezi aktuální pozicí a novou souřadnicí. Při analýze se také rozlišuje mezi absolutními a relativními souřadnicemi, kde relativní hodnoty jsou vypočítávány vůči předchozí pozici robota. (11) (12)

Každý SVG příkaz je následně převeden na odpovídající instrukce Turtle Graphics. Tento převod zahrnuje výpočty úhlů a vzdáleností mezi body, což umožňuje robotovi správně vykreslit daný tvar.

- **Přímé čáry** (L, H, V) jsou převedeny na otočení robota požadovaným směrem a následný pohyb vpřed.
- **Křivky** (C, Q) jsou aproximovány pomocí menších segmentů přímých čar, což zajišťuje plynulost zakřivení.
- **Zavření cesty** (*Z*) znamená návrat robota na počáteční bod cesty. (11) (12)

Aby robot dokázal správně interpretovat SVG příkazy, je nutné provádět výpočty délek a úhlů. K tomu slouží funkce, která na základě souřadnic dvou bodů vypočítá vzdálenost mezi nimi a také úhel, pod kterým se má robot otočit. Před samotným vykreslením je vstupní cesta optimalizována. Proces zahrnuje:

- **Odstranění nadbytečných příkazů**, které neovlivňují výsledek.
- **Přeuspořádání segmentů**, aby vykreslování probíhalo efektivněji.

- **Převod absolutních souřadnic na relativní**, pokud je to vhodné. (11) (12)

Optimalizovaný SVG řetězec se poté převede na sadu příkazů Turtle Graphics, které jsou následně vykresleny. Výsledkem je sekvence příkazů, kterou lze použít k vykreslení původního SVG objektu pomocí Turtle Graphics. Každý příkaz je reprezentován číselným kódem:

- Pohyb vpřed se zapisuje jako *[1, vzdálenost]*
- Otočení vlevo či vpravo se zapisuje jako *[2, úhel]* a *[3, úhel]*
- Položení nebo zvednutí pera se zapisuje jako *[4, 0]* nebo *[5, 0]* (11) (11)

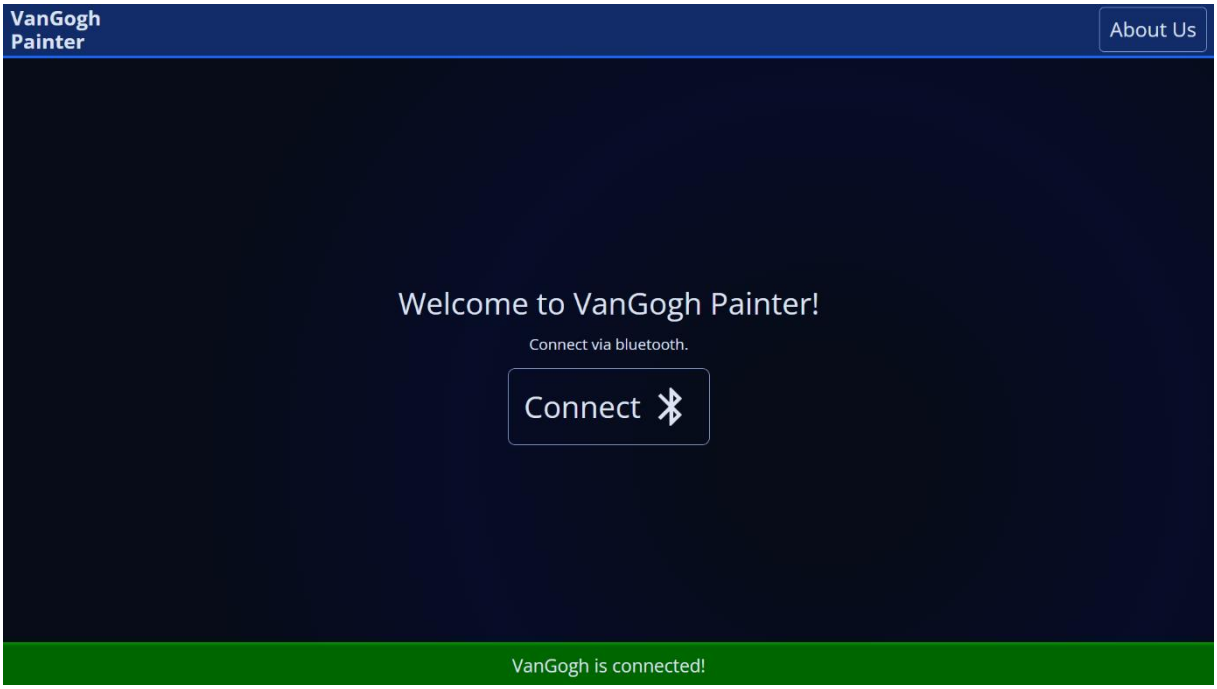
2 Webová aplikace

2.1 UI/UX

Návrh webové aplikace byl vytvořen ve Figmě a slouží jako vizuální a funkční rozvržení celého systému. Hlavním cílem designu je vytvořit moderní a přehledné uživatelské rozhraní, které bude intuitivní a snadno ovladatelné. Struktura webu je rozdělena do jasně definovaných sekcí a používá konzistentní barevné schéma, aby bylo vše vizuálně sladěné.

Důležitou součástí návrhu je využití komponentového a modulárního přístupu, což znamená, že jednotlivé prvky webu lze snadno upravovat a opakovaně používat. Tento způsob usnadňuje rozšiřování webu v budoucnu a zajišťuje, že celý design zůstane jednotný. Návrh také respektuje principy UX/UI, tedy pravidla pro správnou použitelnost a vizuální přehlednost, což zajišťuje snadnou navigaci a pohodlnou práci s webovou aplikací.

Web také podporuje světlý a tmavý režim na základě nastavení systému.



Obrázek 1 Úvodní stránka

2.2 Mechaniky

2.2.1 Připojení robota k webu

Základní prvek při práci s micro:bit robotem je jeho připojení k webové aplikaci. To je umožněno pomocí technologie Bluetooth Low Energy (BLE), která umožňuje komunikaci mezi micro:bitem a prohlížečem bez nutnosti instalace speciálních ovladačů.

Funkce *handleConnect* je hlavní částí kódu, která se stará o připojení micro:bitu k webové aplikaci. Tento proces probíhá asynchronně, protože samotné připojení přes BLE může trvat několik sekund. Na začátku kódu se kontroluje, zda je už nějaké zařízení připojeno (*if (device)*). Pokud ano, zkontroluje se stav připojení přes *device.gatt.connected*. Pokud je už nějaké zařízení připojeno, z důvodu stability a vyhnutí se nechtěným chybám a stavům, se původní zařízení odpojí a poté se pokusí o spojení s novým zařízením. Pokud zařízení připojeno není, funkce pokračuje požadavkem na výběr micro:bitu.

```
const newDevice = await
requestMicrobit(window.navigator.bluetooth);
if (newDevice) {
    setDevice(newDevice);
    microbitStore.update("device", newDevice);
    const services = await getServices(newDevice);
    microbitStore.update("services", services);
```

Tato část kódu umožňuje uživateli vybrat micro:bit pomocí *requestMicrobit()*, což je funkce, která otevře nativní dialog pro výběr dostupných BLE zařízení. Pokud je micro:bit úspěšně vybrán, uloží se do proměnné *newDevice*. Následně se provede aktualizace dat v *microbitStore*, což je objekt sloužící k ukládání informací o připojeném zařízení.

Další klíčovou částí je načtení BLE služeb. Zde se ověřuje, zda micro:bit podporuje službu *deviceInformationService*, což je jedna ze základních BLE služeb, která poskytuje informace o zařízení, jako je výrobce nebo verze firmwaru. Pokud je služba dostupná, načtou se informace o zařízení a uloží se do *microbitStore*.

Při nečekaném odpojení micro:bitu (například pokud se vybije nebo se vzdálí z dosahu) je nutné spojení znovu navázat. To je zajištěno tím, že se na web nastaví událost *gattserverdisconnected*, která při odpojení zavolá znovu funkci *handleConnect*. Na konci se aktualizuje stav aplikace (*dispatch*) a uživatel je přesměrován na stránku */upload*, kde může s micro:bitem dále pracovat.

Pro uložení informací o připojeném zařízení slouží objekt *microbitStore*, který využívá třídu *Store*. Vytvořením rozhraní *MicrobitStore* je definováno, jaká data o zařízení budou uchovávána. Vlastnost *device* obsahuje informace o BLE zařízení, *services* představuje seznam dostupných služeb a *deviceInformation* ukládá podrobnosti o micro:bitu. Třída *Store* umožňuje dynamické ukládání a správu dat souvisejících s připojeným micro:bitem.

Data jsou ukládána do objektu *data*, kde je každá hodnota přístupná podle svého klíče. Funkce *_update* umožňuje změnu hodnoty v *data* a zároveň informuje všechny komponenty, které dané hodnoty sledují. Kromě ukládání nových dat umožňuje třída *Store* i přidání posluchačů, kteří jsou upozorněni na změny hodnot. Díky této funkci je možné zajistit, že když dojde ke změně například připojeného zařízení, všechny součásti aplikace, které na této informaci závisí, budou automaticky aktualizovány.

Pro vymazání všech uložených dat je v třídě *Store* definována funkce *empty*. Tato funkce je klíčová při odpojení micro:bitu, protože zajistí, že v aplikaci nebudou zůstávat zastaralé informace o dříve připojeném zařízení.

2.2.2 Protokol komunikace

Pro komunikaci mezi počítačem a robotem se používá **UART protokol**, který umožňuje odesílání textových příkazů. Jelikož micro:bit má omezenou velikost přenášených dat, je nutné zprávy rozdělovat na menší části, aby bylo možné je správně přenést.

Aby bylo možné poslat delší řetězec příkazů micro:bitu, je nutné ho rozdělit na menší bloky. To je úkolem funkce *splitTextByBytes*, která převádí vstupní text na sekvenci částí nepřesahujících stanovenou velikost v bajtech. Funkce používá objekt *TextEncoder* k převodu textu na bajty. Pro každý znak ve vstupním textu se kontroluje, zda by jeho přidání do aktuálního bloku nepřekročilo limit. Pokud ano, aktuální blok se uloží do pole *chunks* a začne se vytvářet nový blok.

```
function splitTextByBytes(text: string, maxBytes: number =
500): string[] {
    const encoder = new TextEncoder();
    const chunks: string[] = [];
    let currentChunk = "";
    for (const char of text) {
        const currentBytes =
encoder.encode(currentChunk).length;
        const charBytes = encoder.encode(char).length;
if (currentBytes + charBytes > maxBytes) {
```

```
            chunks.push(currentChunk + "\n");
            currentChunk = char;
        } else {
            currentChunk += char;
        }
    } if (currentChunk) {
        chunks.push(currentChunk + "\n");
    }
    return chunks;
}
```

Každý blok je zakončen znakem *\n*, což zajistí, že micro:bit správně interpretuje jednotlivé zprávy.

Funkce *sendArray* je hlavní částí komunikace s micro:bitem. Nejprve ověří, zda je zařízení správně připojeno a zda je dostupná služba *uartService*, která umožňuje odesílání dat. Pokud je spojení dostupné, data se převedou do formátu vhodného pro odeslání. Každý příkaz je reprezentován jako pole čísel v textové podobě, která je následně spojena do jednoho řetězce.

Vzhledem k limitům přenosu přes Bluetooth se tento řetězec rozdělí na menší části pomocí výše popsané funkce *splitTextByBytes*. Velikost jednotlivých bloků je nastavena na 18 bajtů. Než se začne s odesíláním dat, pošle se do micro:bitu signál, který informuje zařízení o začátku přenosu. Micro:bit během procesu odesílání odpovídá různými zprávami, které indikují stav přenosu. Pro zachycení těchto zpráv se nastaví událost *receiveText*, která reaguje na přijatá data. Pokud zpráva začíná znakem *%dr*, znamená to, že se micro:bit snaží vykreslit určitou část dat. Na základě této informace se vypočítá postup vykreslování v procentech a aktualizuje se zobrazení.

```
if (detail.startsWith("%dr")) {
    const drawn = detail.slice(3);
    const x = (parseInt(drawn) / (outputCommands.length)) *
100;
    document.querySelector(`div[data-index="${drawn -
1}"]`)?.classList.remove(Styles["dactive"]);
    document.querySelector(`div[data-index="${drawn -
1}"]`)?.classList.add(Styles["dcomplete"]);
    document.querySelector(`div[data-
index="${drawn}"]`)?.classList.add(Styles["dactive"]);
    setProgress(x);
```

```
}
```

Dále se reaguje na konkrétní stavové zprávy:

- *%rsta* – potvrzuje zahájení nahrávání
- *%rend* – potvrzuje dokončení nahrávání
- *%dsta* – značí začátek kreslení
- *%dend* – informuje o dokončení kreslení

Následuje samotné odeslání všech bloků textu. Každý blok je postupně posílán přes *uartService.sendText*. Odesílání se provádí v cyklu, který zajišťuje, že každý blok je odeslán samostatně. Po odeslání všech bloků se pošle speciální znak *#\n*, který informuje micro:bit o ukončení přenosu.

2.2.3 Převod tvarů na cesty

Součástí převodu z SVG dat na Turtle graphics je také důležité převést tvary, jako například kruhy, obdélníky a podobné. Tyto prvky nemají atribut *d*, a proto je nelze rovnou zpracovat stejně jako *path* element.

Abychom je mohli použít v našem systému, musí se převést na odpovídající cesty. Každý prvek je nejprve zkontrolován, zda není součástí *<symbol>*, protože symboly nechceme zahrnovat do výstupních dat. Samotný převod na *path* se provádí pomocí funkce *shapeToPath*, která je součástí knihovny *svg-path-commander*. Tato funkce převede všechny základní tvary do jednotného formátu *path*, který můžeme snadno zpracovat. Výsledný *path* se pak zpracuje stejně jako každý jiný – získá se jeho *d* atribut a uloží se do pole *pathData*. Nakonec funkce vrátí pole *pathData*, které obsahuje všechny extrahované cesty v jednotném formátu.

2.2.4 Zobrazení cesty robota v reálném čase

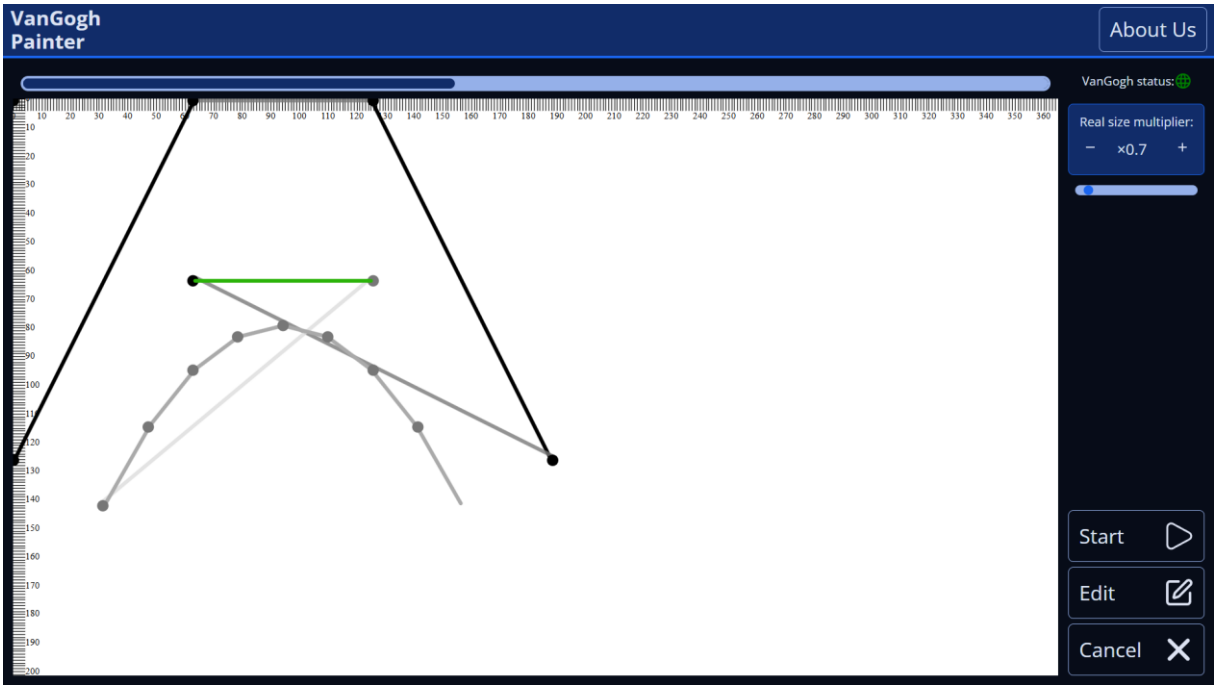
Funkce *calc()* je funkce, která zpracovává vstupní pole číselných hodnot a převádí je na geometrické prvky. Tato funkce přijímá dvourozměrné pole čísel jako vstupní parametr a vrací dva výsledky – pole úhlů a pole čar. Na začátku funkce jsou inicializovány základní proměnné pro sledování pozice a orientace – souřadnice začínají v bodě *[0, 0]* a počáteční úhel je nastaven na 0 stupňů. Dále je definována proměnná *pendown*, která sleduje, zda je pero v kreslicí poloze nebo ne.

V hlavní smyčce funkce se prochází každý příkaz ze vstupního pole a provádí se odpovídající akce podle číselného kódu příkazu. Pokud je kód příkazu 5, pero se zvedne a nebude kreslit. Naopak když je kód 4, pero se položí a začne kreslit. Příkazy s kódem 3 a 2 slouží k rotaci – příkaz 3 otáčí doprava o zadaný úhel, zatímco příkaz 2 otáčí doleva. Při každé rotaci se do pole úhlů přidá nový záznam obsahující hodnotu rotace, aktuální pozici a index příkazu.

Nejdůležitější je příkaz s kódem 1, který představuje pohyb vpřed o určitou vzdálenost. Při tomto příkazu se do pole čar přidá nový segment s informacemi o tom, zda pero kreslí, o úhlu rotace, délce čáry a aktuální pozici. Následně se aktualizují souřadnice pomocí trigonometrických funkcí, které zohledňují aktuální úhel a vzdálenost pohybu.

Druhá funkce je implementována jako posluchač událostí pro UART službu, která zpracovává textové zprávy přijaté od robota. Tato funkce je klíčová pro sledování postupu kreslení v reálném čase. Když přijde zpráva začínající řetězcem *"%dr"*, funkce extrahuje informaci o aktuálně kresleném příkazu a vypočítá procento dokončení celého kreslení. Tato informace je použita k aktualizaci uživatelského rozhraní – funkce odstraňuje a přidává CSS třídy k příslušným elementům, čímž **vizuálně označuje aktivní a dokončené části kresby.**

Funkce také reaguje na specifické stavové zprávy od robota. Zpráva *"%rsta"* označuje začátek nahrávání dat do robota, zatímco *"%rend"* signalizuje dokončení nahrávání. Podobně *"%dsta"* označuje začátek kreslení a *"%dend"* jeho dokončení. Při těchto událostech funkce aktualizuje indikátor průběhu a vypisuje stavové informace do konzole.



Obrázek 2 Stránka s mapováním cesty

2.2.5 Škálování

Tato funkce slouží k úpravě velikosti cesty (*path*) ve formátu SVG tak, aby odpovídala požadované šířce. Při tomto procesu se zachovává poměr stran a relativní poloha prvků v cestě. Funkce přijímá dva parametry:

- *path* – obsahuje definici cesty ve formátu SVG.
- *targetWidth* – požadovaná šířka výsledné cesty.

Cílem je upravit souřadnice všech bodů v cestě tak, aby odpovídaly novému rozměru, přičemž se zachová původní poměr stran.

Vstupní cesta je nejprve rozdělena na jednotlivé příkazy a jejich číselné argumenty pomocí regulárního výrazu. V SVG cestách se používají různé příkazy, které určují tvar křivky. Každý příkaz může obsahovat několik číselných argumentů reprezentujících souřadnice bodů.

Regulární výraz */([MLHVCSQTAZmlhvcsqtaz])([^MLHVCSQTAZmlhvcsqtaz]*)/g* slouží k rozdělení cesty na jednotlivé příkazy a jejich argumenty. Pro každý nalezený příkaz se extrahují číselné argumenty, které jsou převedeny na pole čísel (*parseFloat*). Aby bylo možné určit měřítko, je nejprve nutné zjistit rozsah souřadnic (*xMin*, *xMax*, *yMin*, *yMax*) Každý příkaz obsahující souřadnice (*M*, *L*, *T*, *C*, *S*, *Q*, *A*) je analyzován a jeho souřadnice jsou porovnány s aktuálním minimem a maximem. Speciální případy, jako *H* (horizontální čára) a *V* (vertikální čára), jsou zpracovány samostatně. Na konci této části se získají minimální a maximální hodnoty souřadnic v ose X a Y.

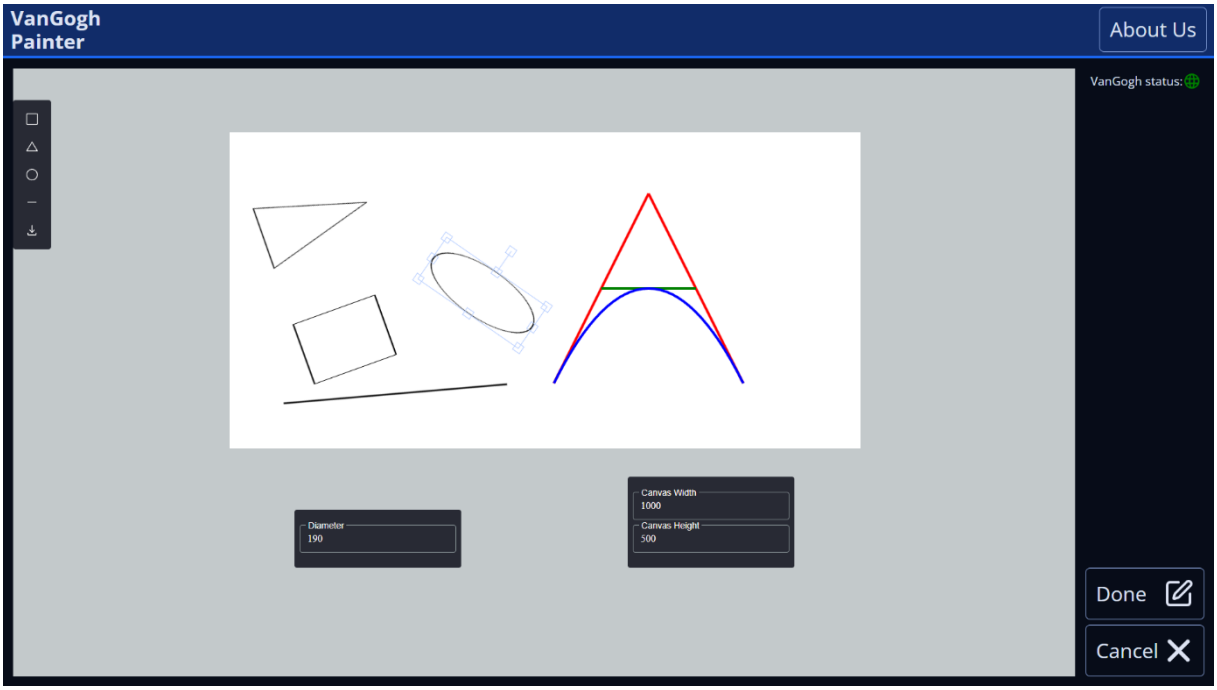
Po získání šířky (*currentWidth = xMax - xMin*) se vypočítá měřítko (*scale = targetWidth / currentWidth*), které se použije k úpravě souřadnic. Pokud je šířka 0, funkce varuje uživatele (*console.warn*) a vrací původní cestu.

Kromě škálování je potřeba také zajistit, aby nový SVG začínal v bodě *(0,0)*. Toho se dosáhne odečtením *xMin* a *yMin* od všech souřadnic. Každý příkaz je převeden na novou hodnotu podle vypočteného měřítka:

- Souřadnice se vynásobí měřítkem a přepočítají tak, aby začínaly od *(0,0)*.
- Zachovává se struktura původní cesty (např. *C* obsahuje šest hodnot, *Q* čtyři atd.).
- Čísla jsou zaokrouhlena na dvě desetinná místa (*toFixed(2)*), aby byla výstupní data čitelná a kompaktní.

2.2.6 Editor SVG souborů

SVG editor, vytvořen pomocí knihovny React Sketch Canvas,



Obrázek 3 Editor SVG

2.3 Popis použití

Po načtení webu je možnost kliknout na tlačítko „Connect“, které otevře dialogové okno s dostupnými Bluetooth zařízeními. Po připojení micro:bitu s webem je uživatel automaticky přesměrován na */Upload*, kde má možnost nahrát SVG soubor. Po nahrání je opět automaticky přesměrován, tentokrát na */Painter*.

Zde se vykreslí cesta, kterou robot po spuštění objede. Světle šedé cesty nevykresluje a tmavě šedé vykresluje. Cesta, kterou momentálně projíždí je vyznačena zelenou barvou a hotové jsou vyznačeny černě. Když práci dokončí, zastaví se na konci trasy. Součástí této stránky je také pravítko podél pole, ukazující velikost obrazce a nastavení pro změnu velikosti výtvoru a možnost přiblížení a oddálení obrazce pro lepší viditelnost na webové aplikaci.

Také je přítomno tlačítko pro jednoduché úpravy SVG souboru. Při kliknutí na tlačítko se web přesměruje na */Editor*. Zde je možné přidávat jednoduché tvary, jako jsou obdélníky, kruhy a čáry a manipulovat s jejich základními vlastnostmi, nebo stáhnout SVG soubor z editoru na uložiště počítače.

3 Problémy

3.1 Bluetooth připojení

Jedním z největších problémů při implementaci bylo navázání spojení mezi webovou aplikací a micro:bitem pomocí technologie Bluetooth Low Energy (BLE). I přes opakované úpravy kódu a kontrolu nastavení se nedařilo připojení stabilně navázat, což komplikovalo testování i další vývoj aplikace.

První krok při řešení problému spočíval v kontrole správné inicializace Bluetooth adaptéru v prohlížeči a ověření, zda micro:bit vysílá BLE signál a je v režimu párování. Pro testování byly použity různé prohlížeče, především Google Chrome a Microsoft Edge, kvůli jejich široké podpoře. Přestože micro:bit byl aplikací rozpoznán, připojení se buď ihned přerušilo, nebo vůbec neproběhlo.

Další analýza naznačila, že problém může souviset s omezeními Web Bluetooth API, která se mohou lišit podle verze prohlížeče a operačního systému. Ukázalo se, že některé systémy vyžadují speciální oprávnění k přístupu k BLE zařízením, což mohlo být jedním z důvodů nefunkčního připojení. Kromě toho byla provedena kontrola firmwaru micro:bitu, protože správná verze softwaru je klíčová pro podporu BLE komunikace.

Nakonec se problém podařilo vyřešit díky doporučení vedoucího práce, který navrhl drobné úpravy v kódu a konfiguraci micro:bitu. Po těchto změnách a dodatečném ladění BLE komunikace se podařilo připojení úspěšně navázat na Windows 11, ale pro Windows 10 bylo rozhodnuto, na základě ukončení podpory systému v blízké době, že rozšíření funkčnosti na tento systém je kontraproduktivní a tudíž neuskutečněn.

Tento problém ukázal, že práce s BLE technologií může být velmi nepohodlná a vyžaduje přesné nastavení jak softwaru, tak samotného zařízení. Kromě správné inicializace je nutné brát v úvahu také verzi prohlížeče, oprávnění systému a kompatibilitu firmwaru.

3.2 Editor SVG

Při implementaci editoru SVG souborů se objevil zásadní problém, protože dostupná řešení byla velmi omezená a většina z nich nevyhovovala požadavkům na integraci do webové aplikace. Výběr vhodného editoru provázelo několik komplikací, které ztěžovaly jeho přímé využití v projektu.

Jednou z hlavních překážek při vývoji aplikace byla nekompatibilita části dostupných SVG editorů s Reactem (například „SVGEdit“), který byl použit jako hlavní framework. Aby byly tyto editory použitelné, byly vyžadovány rozsáhlé úpravy kódu, což by mohlo narušit architekturu celé aplikace. Dalším problémem byla nízká modularita některých nástrojů, což znemožňovalo snadné přidávání funkcí nebo jejich efektivní přizpůsobení („svgEditor“). Kromě toho byla řada editorů zastaralá, chyběla jim kvalitní dokumentace nebo už nebyly aktivně vyvíjeny, což výrazně omezovalo jejich využitelnost v moderní webové aplikaci (například „SVGEdit-React“). Editor doporučený vedoucím práce byl splňoval téměř všechny požadavky, ale na konec bylo zjištěno, že do tohoto editoru nelze vkládat již vytvořené SVG soubory a výsledek tohoto editoru je příliš objemný pro micro:bit, což způsobí jeho samovolné odpojení a resetování („React Sketch Canvas“).

Po zvážení všech těchto problémů se jako nejlepší řešení ukázalo vytvořit vlastní SVG editor. Tento přístup umožnil mít plnou kontrolu nad jeho implementací a upravit ho přesně podle potřeb projektu. Základem se stal online návod, který poskytl základní strukturu editoru, ale bylo nutné ho dále upravovat a optimalizovat, aby byl plně funkční. Výsledný editor byl navržen tak, aby byl kompatibilní s Reactem, umožňoval plynulé zpracování SVG souborů a snadno se integroval do celého systému aplikace, i když nebyl tak kvalitní jako „React Sketch Canvas“.

Celý proces ukázal, že hotová řešení nejsou vždy nejlepším řešením a někdy je efektivnější vytvořit vlastní nástroj, který přesně odpovídá požadavkům projektu. Tento přístup umožnil nejen lepší kontrolu nad výsledkem, ale také zajistil budoucí rozšiřitelnost a flexibilitu, což usnadní další vývoj aplikace.

3.3 Práce s SVG

Během vývoje projektu bylo při zpracování SVG souborů náročné extrahovat pouze požadované části. Jedná se o atribut *d*, který definuje tvar křivky. Bylo nutné implementovat filtr, který vybere pouze nezbytné prvky a odstraní nadbytečné.

Dalším problémem bylo vykreslování těchto upravených dat na plátno v rámci webové aplikace. Jejich dynamická úprava a zobrazení vyžadovaly přesné propojení s grafickým rozhraním. Při nesprávném nastavení se mohly jednotlivé prvky zobrazovat nesprávně, docházelo k deformacím nebo nesprávné interpretaci souřadnic.

Třetí výzvou bylo správné škálování vektorových prvků. Jednou z hlavních výhod SVG je možnost libovolného zvětšování a zmenšování objektů bez ztráty kvality, avšak při manipulaci s atributem *viewBox* a jednotlivými transformacemi bylo nutné zajistit, aby se změna velikosti neprojevila negativně na celkovém rozložení prvků. Nevhodná úprava těchto parametrů mohla způsobit nežádoucí změny v proporcích nebo posunutí grafiky mimo požadovanou oblast zobrazení.

Řešení těchto problémů vyžadovalo důkladné pochopení struktury SVG, správné implementace algoritmů pro výběr a zpracování prvků a optimalizaci vykreslování na webové platformě. Po úpravách bylo nakonec dosaženo správného zpracování SVG souborů tak, aby byly plně funkční a odpovídaly požadovaným kritériím projektu.

4 Testování

V průběhu vývoje projektu bylo testování prováděno průběžně, aby bylo možné včas odhalit a opravit případné chyby. Tento přístup umožnil minimalizovat vznik kritických problémů a zajistit, že jednotlivé funkce fungují podle očekávání. Každá implementovaná část byla pečlivě ověřována, čímž se předešlo nežádoucím chybám, které by mohly ovlivnit celkovou stabilitu systému.

Největší pozornost byla věnována testování funkcí souvisejících se škálováním, mapováním trasy a připojením robota. Tyto části vyžadovaly opakované úpravy a ladění, protože se ukázalo, že i malé nepřesnosti mohly vést k nesprávnému zobrazování nebo nefunkčnosti některých prvků.

Po dokončení klíčových funkcí byl proveden beta-testing, jehož cílem bylo ověřit, zda je webová aplikace intuitivní a snadno ovladatelná. Web byl zpřístupněn několika uživatelům, kteří měli za úkol samostatně projít jeho rozhraním a využít nabízené funkce. Tento test byl úspěšný, protože uživatelé byli schopni projít celou aplikací bez větších problémů, což potvrdilo její použitelnost a správnou implementaci jednotlivých částí. Menší problémy byly pouze s responzivitou webu na různých platformách a podobné záležitosti.

Na základě zpětné vazby od testerů byly provedeny drobné úpravy (zejména srovnání několika vzhledových nesrovnalostí), které přispěly ke zlepšení uživatelského zážitku. Díky testování v průběhu celého vývoje bylo možné vytvořit stabilní a funkční aplikaci, která splňuje požadované technické i uživatelské nároky.

Závěr

Vytvoření webu bylo ve výsledku úspěšné s funkčním komunikací mezi webovou aplikací a robotem, intuitivním a přívětivým UI, funkčním mapováním trasy, škálováním a základním editorem přímo v prohlížeči. Největší výzvy tkvěly v samotném připojení robota, při manipulaci s SVG soubory a implementací vhodného editoru pro SVG soubory.

Plán se zásadně musel změnit hlavně při implementaci SVG editoru, kde původní plán byl najít již hotovou knihovnu s editorem a vložit ji do projektu, ale žádná nevyhovovala požadavkům anebo nebyla kompatibilní. Tudíž místo toho byla vytvořena primitivnější verze editoru, podle již existujících materiálů, která byla následně upravena podle potřeby.

Projekt se dá dále rozvíjet, převážně v ohledu hlubšího rozšíření editoru, nebo umožnění nahrání jiných formátů vektorové grafiky, místo pouze SVG souborů. Vylepšení lze určitě uplatnit například v ohledu vyšší flexibility možností přibližování a oddalování plátna, nebo optimalizaci samotného kódu. Udržitelnost by měla být dostatečná na dohlednou dobu.

Seznam zkratek a odborných výrazů

HTML
HyperText Markup Language – značkovací jazyk používaný pro tvorbu webových stránek.

React
Open-source JavaScriptová knihovna pro tvorbu uživatelských rozhraní.

SVG
Scalable Vector Graphics – vektorový formát pro 2D grafiku založený na XML.

BLE
Bluetooth Low Energy – bezdrátová komunikační technologie s nízkou spotřebou energie.

UART
Universal Asynchronous Receiver-Transmitter – protokol pro sériovou komunikaci mezi zařízeními.

IDE
Integrated Development Environment – integrované vývojové prostředí.

UI/UX
User Interface / User Experience – návrh uživatelského rozhraní a uživatelského zážitku.

API
Application Programming Interface – rozhraní pro komunikaci mezi softwarovými komponentami.

DOM
Document Object Model – objektový model pro manipulaci s HTML a XML dokumenty.

Web Bluetooth API
Technologie umožňující webovým aplikacím komunikovat s BLE zařízeními.

WebStorm
Vývojové prostředí od JetBrains optimalizované pro JavaScript a React.

Micro:bit
Programovatelný mikropočítač používaný převážně pro výuku programování.

Micro:bit Web Bluetooth
Knihovna umožňující komunikaci mezi Micro:bitem a webovou aplikací přes BLE.

Micro:bit Web Components
Webové komponenty usnadňující interakci s Micro:bitem.

VanGogh Extension
Rozšíření pro Micro:bit umožňující práci s grafikou a LED displejem.

Figma
Cloudový nástroj pro návrh uživatelského rozhraní.

Turtle Graphics
Grafický systém pro vykreslování obrazců pomocí příkazů robota.

Protokol komunikace
Soubor pravidel určujících způsob výměny dat mezi zařízeními.

IoT
Internet of Things – síť fyzických zařízení vybavena elektronikou a jsou schopna si vyměňovat data.

Seznam obrázků

Obrázek 1 Úvodní stránka 6
Obrázek 2 Stránka s mapováním cesty 11
Obrázek 3 Editor SVG 13

Použité zdroje

1. **JetBrains s.r.o.** WebStorm. *JetBrains.* [Online] [Citace: 14. únor 2025.] https://www.jetbrains.com/webstorm/.
2. **Meta Platforms, Inc.** React. *React.* [Online] [Citace: 14. únor 2025.] https://react.dev/.
3. **Moran, Rob.** microbit-web-bluetooth. *Github.* [Online] 6. květen 2023. [Citace: 2. leden 2025.] https://github.com/thegecko/microbit-web-bluetooth.
4. —. microbit-web-components. *Github.* [Online] 6. květen 2023. [Citace: 5. leden 2025.] https://github.com/thegecko/microbit-web-components.
5. **Grochocki, Sebastian.** FabricJs 6 and React Tutorial | Adding Shapes to Canvas - Part 1. *YouTube.* [Online] 16. srpen 2024. [Citace: 18. únor 2025.] https://www.youtube.com/watch?v=eSiEBH7D1mM&list=PLOmd6EbLLA_oLtJ9howoPC01788f1dtEz&index=4.
6. **Lancaster University.** Bluetooth UART Service. *micro:bit runtime.* [Online] [Citace: 24. únor 2025.] https://lancaster-university.github.io/microbit-docs/ble/uart-service/.
7. **NORDIC Semiconductors.** Nordic UART Service (NUS). *NORDIC Semiconductors.* [Online] 10. březen 2025. [Citace: 10. březen 2025.] https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/libraries/bluetooth/services/nus.html.
8. **Rowbitt, Mike.** UART. *BBC micro:bit MicroPython.* [Online] 2016. [Citace: 9. březen 2025.] https://microbit-micropython.readthedocs.io/en/v1.0.1/uart.html.
9. **microbit-cz.** pxt-vangogh-extension. *Github.* [Online] 24. červen 2023. [Citace: 11. prosinec 2024.] https://github.com/microbit-cz/pxt-vangogh-extension.
10. **Wikipedia Foundation, Inc.** Figma. *Wikipedia The Free Encyclopedia.* [Online] 26. únor 2025. [Citace: 9. březen 2025.] https://en.wikipedia.org/wiki/Figma.
11. **microbit-cz.** pxt-vangogh-SVG. *Github.* [Online] 24. červen 2024. [Citace: 11. prosinec 2024.] https://github.com/microbit-cz/pxt-vangogh-SVG/blob/main/algorithm/converter.js.
12. **Syahrasyad, Nanda.** Understanding SVG Paths. *Not a Number.* [Online] 17. červenec 2023. [Citace: 26. prosinec 2024.] https://www.nan.fyi/svg-paths.

A. Seznam přiložených souborů

- https://van-gogh-painter-web.vercel.app
- https://github.com/pslib-cz/MP2024-25_Holy-Jan_VanGogh-Painter

Na přiloženém datovém nosiči se nacházejí následující soubory a složky:

- **MP2025-Holý-Jan-P4A-VanGogh_Painter.docx** – editovatelná verze dokumentace maturitní práce
- **MP2025-Holý-Jan-P4A-VanGogh_Painter.pdf** – tisknutelná verze dokumentace maturitní práce
- **MP2024-25_Holy-Jan_VanGogh-Painter** – složka se zdrojovými kódy