Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Rombergovy křivky v díle malíře Vladislava Mirvalda

Tomáš Mayer

Bakalářská práce
2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Mayer**
Osobní číslo:	**I18163**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Rombergovy křivky v díle malíře Vladislava Mirvalda**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Český malíř Vladislav Mirvald je jedním z prvních českých autorů, který využíval při své tvorbě matematické objekty. Jeho tvorba bývá zařazována do oblasti nefigurální tvorby a informelu. Při zpracování práce bude třeba prostudovat a odhalit geometrické principy, podle kterých Vladislav Mirvald vytvářel své kontinuální, tangenciální a undulační válce. Tyto atraktivní geometrické obrazy Mirvald vytvářel pomocí statokinesigramu, který se používá pro zkoušku stability. Záznam pohybu testované osoby vytvoří tzv. Rombergovu křivku. Ta zaznamenává pohyb hlavy, když stojící člověk se zavřenýma očima vyrovnává svou polohu. Křivka byla následovně využita jako osa undulačního válce, jenž měl manifestovat organickou morfologii.

Cílem práce bude naprogramovat aplikaci, která:dokáže změřit nebo nasimulovat statokinesigram a pro získanou Rombergovu křivku vytvoří undulační válec.

Součástí práce bude uživatelská příručka.

Rozsah pracovní zprávy: **35-45**
Rozsah grafických prací: **10**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

POSPISZYL, Tomáš a Vladislav MIRVALD. Mirvald. Praha: Arbor Vitae, 2010. 160 s. ISBN 9788087164587.

SEDLÁŘ, Jaroslav. Ismy: umění 20. století. Praha: Meridian Word Press, 2014. 612 s. ISBN 9780968529355.

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem Rombergovy křivky v díle malíře Vladislava Mirvalda jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 5. 2023

Tomáš Mayer

PODĚKOVÁNÍ

Děkuji vedoucímu práce Mgr. Jaroslavu Markovi, Ph.D. za pomoc při vypracování práce.

Také děkuji rodině a přátelům za podporu při mém studiu.

ANOTACE

Bakalářská práce se zaměřuje na Mirvaldovy Rombergovy křivky. V první části je krátce popsán život a dílo Vladislava Mirvalda. V další kapitole jsou prezentovány základní ideje Rombergova testu z neurologie, který vyšetřuje stabilitu pacienta. Nejdůležitější částí práce je popis algoritmu pro matematické uchopení Mirvaldových undulačních válců, jejichž osu tvoří Rombergova křivka. Aplikace naprogramovaná v jazyku Kotlin umožňuje načtení pozic hlavy vyšetřované osoby, resp. jejich sběr, jiná aplikace naprogramovaná v jazyku C# umožňuje vykreslení Rombergovy křivky a undulačních válců, export obrázku do souboru v různých grafických formátech.

KLÍČOVÁ SLOVA

Vladislav Mirvald, Rombergův test, Rombergovy křivky, B3-splajny, zachycení pohybu

TITLE

Romberg curves in the works of painter Vladislav Mirvald.

ANNOTATION

The bachelor's thesis focuses on Mirvald's Romberg curves. The first part briefly describes the life and work of Vladislav Mirvald. The next chapter presents the basic ideas of the Romberg test from neurology, which examines the stability of a patient. The most important part of the thesis is the description of the algorithm for mathematically capturing Mirvald's undulatory cylinders, whose axis is formed by the Romberg curve. An application programmed in the Kotlin language allows for loading the positions of the examined person's head, or rather, their collection. Another application programmed in the C# language allows for the visualization of the Romberg curve and undulatory cylinders, as well as exporting the image to a file in various graphic formats.

KEYWORDS

Vladislav Mirvald, Romberg test, Romberg curves, B3-splines, motion capture.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

API	Application Programming Interface

ÚVOD

Vladislav Mirvald patří s Kamilem Linhartem a Jiřím Sýkorou mezi tři lounské osobnosti, kteří zanechali významné stopy ve výtvarném umění 20. století. Zdeněk Sýkora patří mezi pionýry užívání počítače ve výtvarné tvorbě, když ke své tvorbě linií a struktur používal sady náhodně generovaných čísel. Do děl Vladislava Mirvalda náhodnost neinfiltruje v podobě náhodně generovaných čísel ale náhodně vzniklých procesů založených na Brownově pohybu při rozpíjení se barev či jejich zmrznutí. Mirvaldova díla mají často značně geometrizující charakter. Do této skupiny patří i tzv. undulační válce, které jsou určeny trajektorií hlavy neurologem vyšetřované osoby při Rombergerově testu. Trajektorie hlavy tvoří osu válce.

Cílem této práce bude pomocí vhodných matematických nástrojů sestavit základy pro vykreslení undulačních válců a vytvořit aplikaci, která je vykreslí.

Mobilní aplikace bude vytvořena v jazyce Kotlin pro mobilní zařízení Android. Aplikace využívá lineární akcelerometr. Vývojovým prostředím pro tvorbu aplikace je Android Studio. Výstupem budou diskrétní 3D souřadnice bodů hlavy vyšetřovaného pacienta, s variabilní dobou mezi měřeními bodů.

PC aplikace bude realizována v jazyku C#, pro konstrukci Beziérova kubického splajnu bude využita interní funkce. Pro grafické zpracování bude využit framework Windows Forms, který je součástí frameworku .NET 7. Framework obsahuje knihovnu *Drawing*, která umí vykreslit Beziérovu křivku. Jako vývojové prostředí slouží Visual Studio 2022. Aplikace umožní konstrukci undulačního válce podle vložené osy.

1 ŽIVOT VLADISLAVA MIRVALDA

1.1 Dětství a dospívání

Vladislav Mirvald byl český pedagog a malíř. Narodil se v Záluží v severních Čechách 3. 8. 1921 do havířské rodiny. V této dnes již neexistující obci soužilo české a německé obyvatelstvo. Nedaleko obce se nacházel důl na hnědé uhlí Herkules, kde pracoval jeho otec. V roce 1932 začal Mirvald navštěvovat České státní reformní gymnázium reálné v nedalekém Mostě. Už v této době se věnoval malbě a kresbě. Dochovaly se jeho výkresy, náčrty a obrázky s výrazným geometrickým charakterem.

Po Mnichovské dohodě se Záluží ocitlo v protektorátu Böhmen und Mähren Protože Mirvaldův otec nechtěl opustit zaměstnání, Mirvald se rozhodl, že se sám přestěhuje do Loun, kde dokončí gymnazijní studium. V Lounech se ale ocitl bez přístřeší a bez jídla. Naštěstí mu pomohli na Okresním úřadě pro mládež, když mu zajistili bydlení a stravu v různých domácnostech.
U lounského lékaře Vlastimila Jurena si zalíbil jeho sbírku umění a obrazů. To ho směřovalo ke studiu výtvarné výchovy. Po maturitě začal pracovat na železnici. Byla to fyzicky a časově náročná práce. Na uměleckou tvorbu mu z celého týdne zbyla jenom neděle. I přesto se dochovalo mnoho přírodních motivů a studií krajin z této doby.

Jedna z nejvýznamnějších osob, se kterou se v té době setkal byl knihovník Jaroslav Janík. Janík lounské mladé umělce podporoval v uměleckých pokusech. Taktéž pořádal hudební a recitační večery, kde se probíral surrealismus. V této době seznámil s budoucími přáteli, Zdeňkem Sýkorou a Kamilem Linhartem. S oběma se Mirvald často setkával a navzájem si ovlivňovali svá díla.

Z počátku Mirvald tvořil díla ovlivněná kubismem, především kubistická zátiší. Pro Mirvalda byl kubismus vstupem do modernismu a možností, jak zobrazovat reálný svět v jednoduchých obrazech. Nebyl to jediný umělecký směr, který Mirvaldovi imponoval. Při vytváření fotokoláží vycházel, v té době,

z oblíbeného surrealismu, když stvářel „nové“ bytosti z různých částí lidského těla. Taktéž vytvořil protiválečnou koláž Stéla hrdinů. Jeho tvorba ke konci války se orientuje na realistickou krajinomalbu.

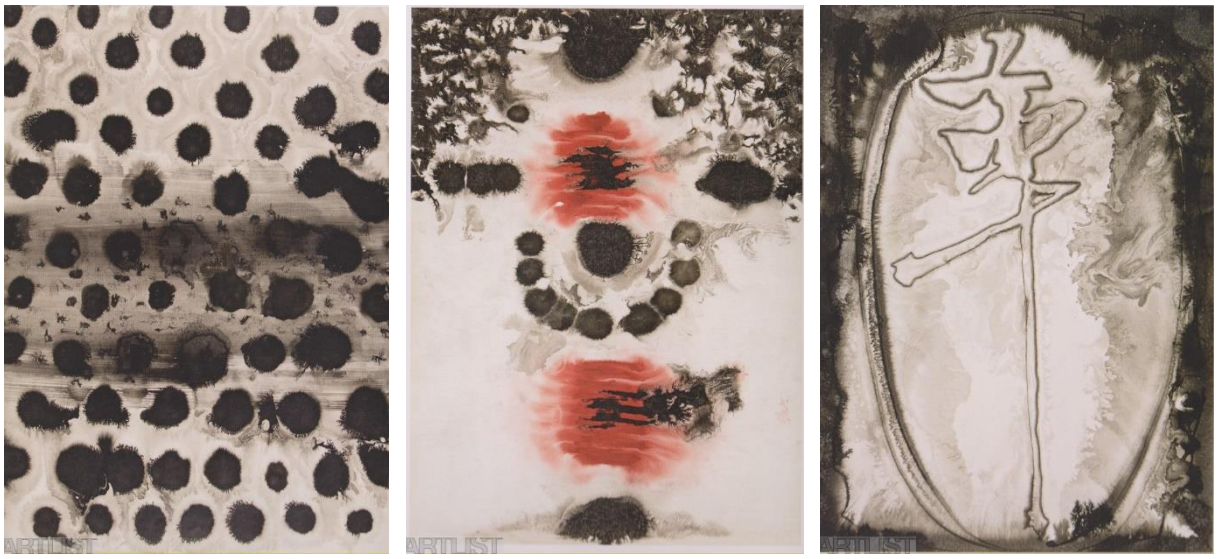
1.2 Poválečná tvorba

Ihned po válce, kdy jsou znovu otevřeny vysoké školy, se Mirvald rozhodne společně se Zdeňkem Sýkorou a Kamilem Linhartem začít studovat výtvarnou výchovu, modelování a deskriptivní geometrii na Českém učení technickém. Později bylo studium převedeno pod Pedagogickou fakultu Univerzity Karlovy. V této době se také do Čech dostávaly nové myšlenkové impulsy a navazující nové výstavy moderního umění.

V době poválečného Československa rostl vliv Sovětského svazu. To pro Mirvalda znamenalo omezení svobodného projevu. Aby si udržel nezávislost, rozhodl, že se tvorba stane privatissimem. To zapříčinilo, že se o jeho tvorbu zajímalo velmi málo lidí, ale také že nebyl ovlivňován kritiky nebo galeristy.

1.3 Kankáže

V letech 1961–1963 byl Mirvald odborný asistent na pedagogické fakultě. Zde objevil nový výtvarný postup, když narazil v Laberennově knize O původu světů kapitolu o Brownově pohybu. Pokud dá do sklenice vody malé zlomky hmoty, je vidět neustálý pohyb úlomků. S touto inspirací využil tuš k tvorbě takzvaných kankáží (Obr. 1 a). Tento styl mu vydržel po dlouhou dobu. Mirvald rád rozmýšlel, jak tento postup rozvinout. Například využíval písmové šablony nebo pracoval s teplotou. Během zimy vynesl ještě mokrou kankáž do mrazu, což způsobilo vytvoření květů (Obr. 1 b). Na tvorbě kankáží Mirvalda nejvíce fascinovala míra náhody. Nyní nevytvářel dílo jen on samotný, ale i matka příroda. Další varianta kankáží bylo využití písmových šablon (Obr. 1 c). Tím kankáže propojil s lettrismem. To je umělecký směr, kde je základním prvkem písmo.



Obr. 1 a) Kankáž, b) Zmrzláž, v) Čínský znak. Zdroj [2][2]

V polovině šedesátých let se Mirvaldova tvorba dostala i mimo úzký okruh příznivců. Přidal se k výtvarné skupině Křižovatka, kterou založil básník Jiří Kolář a teoretik Jiří Padrta. Křižovatka byla neformálním sdružením, která spojovala umělce s konstruktivním uvažováním.

1.4 Geometrická tvorba

Přibližně v polovině roku 1964 začal Mirvald pracovat s dělením obrazu pomocí narýsovaných linií. S vytvořenou strukturou polí pak následně pracoval. Po liniích se Mirvald soustředil na Cicerové kružnice (Obr. 2 a). U této tvorby je znát důraz na geometrii, ale stále Mirvald vkládá prvek náhody. To vkládalo do jeho děl nádech přírody. Tomu napomáhala i volba barev. Mirvald pracoval s principem, kdy volba dvou sousedících barev ovlivňují vnímaní barev v celém obraze.

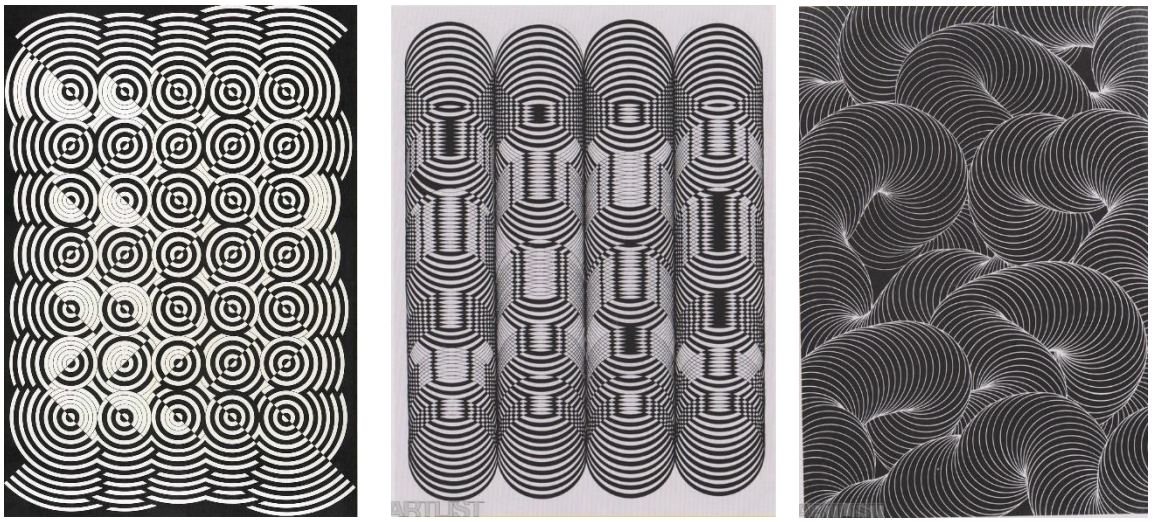
V roce 1965 Mirvald začíná pracovat s aperspektivou válců (Obr. 2 b). Pomocí nich vytváří optické efekty, kdy udržoval diváka na hranici mezi iluzí a popíráním této iluze. Jako s jeho nedávnou tvorbou, pracoval zde s geometrií. Preciznost, se kterou tvořil tato díla, se projevila hlavně na potřebném čase

k dokončení díla. Na druhou stranu to Mirvald bral jako meditativní druh aktivity.

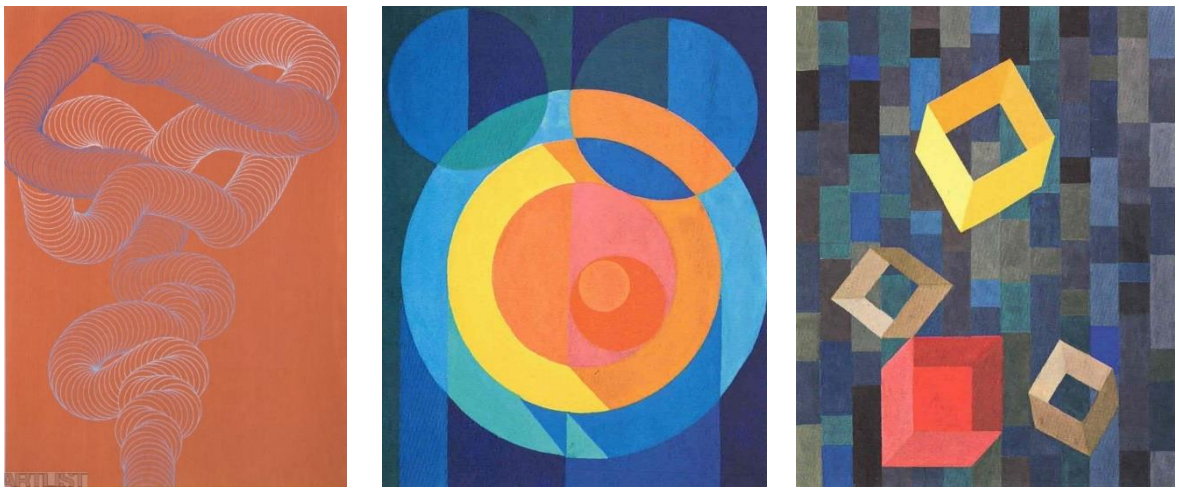
Tomuto stylu nazvaném op-art se tou dobou dařilo v reklamě, módě a filmu. Mirvald ale nechtěl, aby jeho tvorba byla s op-artem spojována, jelikož nechtěl být spojován s módním trendem.

V roce 1979 Mirvald začíná sérii Undulačních válců (Obr. 2 c). V těchto obrazech využívá válcových křivek, které rovnoměrně zaplňují obraz dle jistého pravidla. A jako v jiných obrazech, pravidlo bylo často ovlivněno různou hodnotou náhodnosti. Od roku 1986 začíná cyklus nazvaný Rombergovy křivky (Obr. 3 a). Zatímco dosud byly undulační válce tvořeny pomocí geometrických pravidel nebo náhodnými směry, v této sérii využívá Rombergova testu z neurologie. Rombergův test sloužící k vyšetření stability pacienta bude vysvětlen v další kapitole. Křivku, která vznikla v průběhu Rombergova testu, Mirvald použil jako osu undulační válce. To, jakožto v jeho minulých dílech, dávalo dotyk přírody do jeho tvorby. K další tvorbě použil moaré efekt zvětšených detailů válců. Poloměry těchto válců byly mnohdy mimo kreslící plátno.

V Mirvaldově tvorbě lze na jednu stranu vidět tendence dodávat řád a logiku, ale na druhou stranu nechává přírodní zákony ovlivnit konečný vzhled. Tato kombinace umožňuje vizualizovat přírodní síly v estetické podobě. Ke konci života se Mirvald zabýval geometrickými zátišími (Obr. 3 b, Obr. 3 c). V těch používá kombinaci rovnoběžných i kruhových tvarů. Také zde využívá více barevnosti než obvykle. Zemřel v Lounech 17. dubna roku 2003. [1], [2]



Obr. 2 a) Třícicerové oblouky, Zdroj [1], b) Aperspektiva válců c) Undulační válce. Zdroj [2]



Obr. 3 a) Rombergova křivka, Zdroj, [2] b) Kruhové zátiší, c) Zátiší s pláštěm krychlí. Zdroj [1]

2 ROMBERGŮV TEST

Rombergův test (a Rombergův příznak) je pojmenován po německém lékaři a neurologovi Moritzovi Heinrichovi Rombergovi (1795–1873). Původně byl Rombergův příznak popsán Marshallem Hallem, Moritzem Rombergem a Bernardem Branchem. Rombergův příznak je definován jako stav, kdy pacient není schopen stát bez problémů pevně na zemi, když jsou pacientovi odebrány vizuální vodítka (zavřením očí, nebo temnou místností).

2.1 Historie

V roce 1836 Marshall Hall zjistil u pacienta, který měl Tabes Dorsalis (vysychání míchy), že má problémy udržet rovnováhu ve tmě. Z tohoto příznaku v roce 1840 Moritz Romberg vytvořil test. Takřka současně Bernardus Brach zaznamenal podobný příznak u svých pacientů. Brach taktéž zjistil, že tato nestabilita není spojena se slabostí svalů. V roce 1858 Duchenne de Boulogne zaznamenal případy progresivní ataxie. Zpozoroval, že pacienti s Tabes Dorsalis mají tendenci ztrácet zrak a společně s tím se zhoršuje ataxie.

2.2 Využití

Rombergův test je běžně využíván při neurologických vyšetřeních. Poskytuje cenné informace pro hodnocení integrity míšních sloupců a je obzvláštně užitečný u pacientů s ataxií nebo jinou poruchou koordinace. Pomáhá hodnotit a potvrzovat různá neurologická onemocnění, mezi které může patřit Parkinsonova choroba, nedostatek vitamínu B12, terciální syfilis a Wernickeho encefalopatie. Taktéž je využíván pro měření kinematiky u starší populace.

2.3 Provedení testu

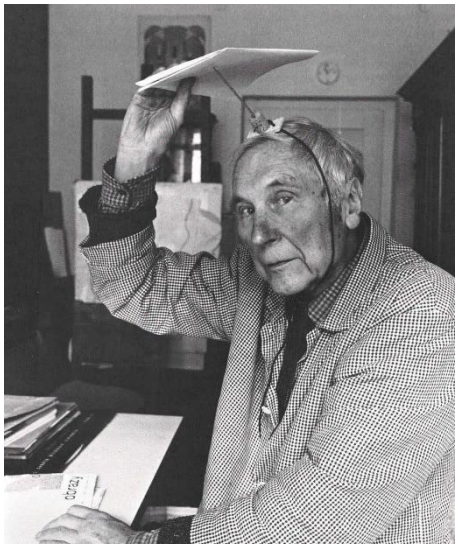
Ačkoliv je Rombergův test jednoduchý na provedení a je možné ho provést bez dodatečné asistence, je důležité dbát na bezpečnost. Pokud testovaná osoba ztratí rovnováhu, hrozí zranění dané osoby. Pozorující osoba by měla být připravena test přerušit a zabránit zranění.

Testovaná osoba se postaví bez bot s chodidly u sebe. Paže připaží nebo bude držet paže překřížené na prsou. V této první části je pozorována stabilita při otevřených očích. Ve druhé části testovaná osoba zavře oči a pokusí se udržet rovnováhu po dobu jedné minuty. Mladé dospělé osoby by měly bez problémů vydržet přes 30 sekund. Je běžné, že se testovaná osoba lehce houpe, což ukazuje snahu osoby o udržení rovnováhy. Test je označen pozitivním, pokud testovaná osoba ztratí rovnováhu. Ta je definována zvýšeným houpáním těla, pohybem nohou k zabránění pádu nebo pádem.

Tento test lze modifikovat pro zvýšení přesnosti testu. Příkladem modifikace může být stání s chodidly za sebou, stání na jedné noze nebo stání na pěnové gumě. Pozitivní výsledek může naznačit, že testovaná osoba může mít zdravotní problém spojený se ztrátou rovnováhy, ale je důležité provést další testy pro určení pravého původce. [3], [4]

3 SBĚR DAT

Pro napodobení Mirvaldových Rombergových křivek je potřeba provést Rombergův test a zaznamenat pohyb hlavy. Mirvald při tvorbě těchto děl používal jednoduchou metodu. Vertikálně připevnil tužku na vrchol hlavy a přidržoval kus papíru/plátna, tak aby tužka kreslila křivku ukazující chvění těla (Obr. 4). V této práci je tato metoda nedostatečná. K získání dat v této práci je vytvořena samostatná aplikace.



Obr. 4 Mirvald demonstruje tvorbu Rombergových křivek. Zdroj [1]

3.1 Aplikace HeadMovementCatcher

Aplikace HeadMovementCatcher je jednoduchá minimalistická aplikace pro mobilní zařízení Android. Aplikace je naprogramována v jazyce Kotlin a používá API 23, což zajišťuje kompatibilitu s operačním systémem android 6.0 Marshamallow a novější. K vývoji aplikace bylo použito Android Studio.

3.1.1 Kotlin

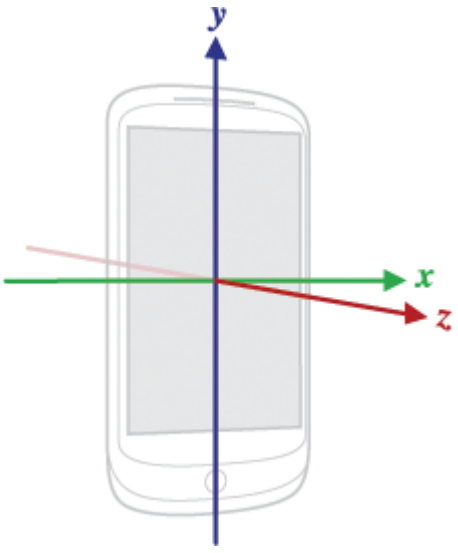
Kotlin je open-source programovací jazyk. Je staticky typovaný a podporuje objektové i funkční programování. Kotlin napodobuje syntaxi a koncepty z jiných jazyků jako je například C# nebo Java. Existuje ve variantách, které se zaměřují na Java Virtual Machine, JavaScript a nativní kód.

Kotlin je spravován organizací Kotlin Foundation, společnost založená organizacemi JetBrains a Google. Kotlin je oficiální podporovaný jazyk pro vývoj pro operační systém Android. Jednou z výhod jazyka Kotlin je Interoperabilní s jazykem Java. To dovoluje využívat oba jazyky společně. [5], [6], [7]

3.1.2 Lineární akcelerometr

K měření pohybu hlavy je využíván akcelerometr v mobilních zařízeních Android. Akcelerometr je zařízení měřící zrychlení. Zařízení Android měří zrychlení ve třech osách, dle souřadnicového systému. Souřadnice jsou odvozeny od výchozí pozice (Obr. 5). Osa *X* je horizontální pohyb, osa *Y* vertikální pohyb a osa *Z* ukazuje pohyb dopředu a dozadu.

Základní akcelerometr, kromě samotné změny rychlosti, je ovlivněn gravitační silou. Je nutné gravitační sílu vyfiltrovat z měření. Android API má již zabudované řešení ve formě abstraktního lineárního akcelerometru. Lineární akcelerometr poskytuje data bez gravitační síly v m/s^2. [8]**Chyba! Nenalezen**



Obr. 5 Souřadnicový systém. Zdroj [8]

zdroj odkazů.

3.1.3 Popis aplikace

Aplikace (Obr. 6) je navržena pro jednoduché použití. V horní části aplikace se nachází vyplnitelné položky. Změnou údajů v jednotlivých položkách lze ovlivnit měření.

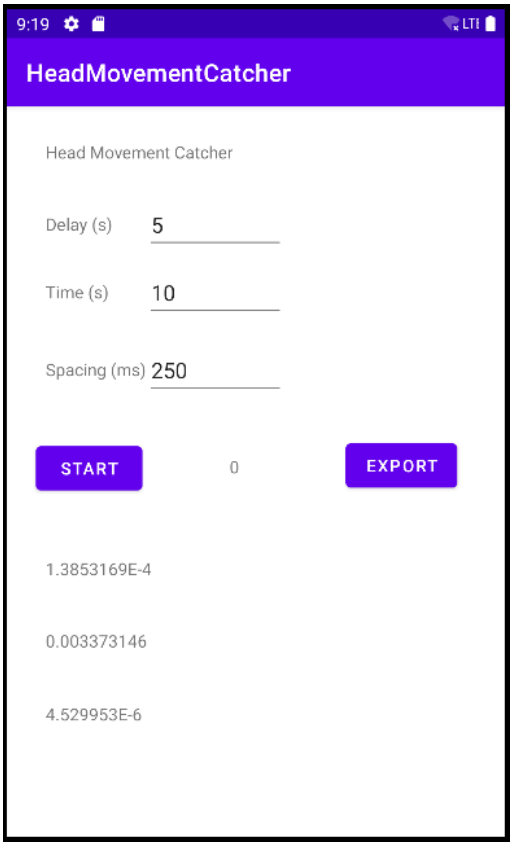
- Položka Delay udává dobu mezi stiskem tlačítka start a začátkem měření v sekundách.
- Položka Time udává dobu měření v sekundách.
- Položka Spacing udává rozptyl v milisekundách mezi při ukládání bodů

V prostřední části se nachází tlačítka „Start“ a „Export“ s číselným textem mezi nimi.

- Tlačítkem „Start“ lze začít měření
- Číslo mezi tlačítky udává počet uložených bodů v současném měření
- Tlačítko „Export“ uloží textový soubor do mobilního zařízení

V dolní části aplikace se nachází tři číselné hodnoty.

- Aktuální hodnota lineárního akcelerometru v ose X.
- Aktuální hodnota lineárního akcelerometru v ose Y.
- Aktuální hodnota lineárního akcelerometru v ose Z.



Obr. 6 Aplikace HeadMovementCatcher. Zdroj vlastní

3.1.4 Chování aplikace

Aplikace si uchovává relativní pozici od začátku posledního měření. Při změně dat v senzoru se automaticky volá metoda onSensorChanged. Uvnitř této metody se aktualizuje zobrazovaná hodnota senzorů a aktualizuje současná relativní pozice.

```
override fun onSensorChanged(event: SensorEvent?) {
    if (event?.sensor?.type==Sensor.TYPE_LINEAR_ACCELERATION){

        X = event.values[0]
        XText.text = X.toString()
        corX += X

        Y = event.values[1]
        YText.text = Y.toString()
        corY += Y

        Z = event.values[2]
        ZText.text = Z.toString()
        corZ += Z
    }
}
```

Při stisku tlačítka „Start“ se aktivuje měření. Tlačítko „Start“ se během měření deaktivuje. Hodnoty relativní pozice (corX, corY, corZ) se vynulují. Na základě vyplněných hodnot time a spacing se vytvoří CountDownTimer, který bude v pravidelném intervalu o velikosti hodnoty spacing zaznamenávat relativní pozici vzhledem k začátku měření. Hodnota zobrazující počet uložených bodů se aktualizuje podle počtu ticků.

```
fun recordMovement(){

    startButton.isEnabled = false

    var delay : Long = delayText.text.toString().toLong()
    var time : Long = timeText.text.toString().toLong()
    var spacing: Long = spacingText.text.toString().toLong()
    var con = this
    var count : Int = 0
    counterText.text = "0"

    var tempN = (time*1000/spacing).toInt()
    points = ArrayList<Point>()
    sleep(delay*1000)

    var timer: CountDownTimer = object: CountDownTimer((time*1000),
spacing){
        override fun onTick(millisUntilFinished: Long) {
            count++
            counterText.text = count.toString()
            var p: Point = Point()
            p.X = corX
            p.Y = corY
            p.Z = corZ
            points.add(p)
        }

        override fun onFinish() {
            startButton.isEnabled = true
        }
    }
    corX   = 0f
    corY   = 0f
    corZ   = 0f
    timer.start()
}
```

Při stisku tlačítka „Export“ se uživateli zobrazí aktivita pro vytvoření souboru.

```
fun exportData (){
    val intent = Intent(Intent.ACTION_CREATE_DOCUMENT).apply {
        addCategory(Intent.CATEGORY_OPENABLE)
        type = "application/txt"
        putExtra(Intent.EXTRA_TITLE, "dataPoints.txt")
    }
    startActivityForResult(intent, 1)
}
```

Po dokončení aktivity, se do vybraného souboru uloží data z posledního měření. Data se ukládají ve formátu „*X*;*Y*;*Z*“, kde každý řádek znamená jiný bod.

```
override fun onActivityResult(
    requestCode: Int, resultCode: Int, resultData: Intent?) {
    if (requestCode == 1
        && resultCode == Activity.RESULT_OK) {
        resultData?.data?.also { uri ->
            try {
                contentResolver.openFileDescriptor(uri, "w")?.use {

                    points.forEach { p ->
FileOutputStream(it.fileDescriptor).use {
                        it.write((p.X.toString()+ ";" +p.Y.toString()+ ";"
+p.Z.toString() + "\n").toByteArray())
                        }
                    }
                }
            } catch (e: FileNotFoundException) {
                e.printStackTrace()
            } catch (e: IOException) {
                e.printStackTrace()
            }
            Toast.makeText(this,"File saved", Toast.LENGTH_SHORT).show()
        }
    }
}
```

Ukázky hodnot z textového souboru:

-70.89225;-748.9291;200.40013

-67.632225;-723.292;153.77345

-54.969196;-748.7402;181.10866

Doba mezi měřeními těchto diskrétních bodů bylo 250 ms.

3.1.5 Provedení měření

Před začátkem měření je důležité zajistit bezpečnost. Pokud měřená osoba ztratí rovnováhu, hrozí zranění.

1. V aplikaci se nastaví parametry měření. Je možné ponechat výchozí nastavení.
2. Testovaná osoba se postaví stabilně na zem.
3. Stiskne se tlačítko „Start“.
4. Momentálně běží odpočet, během této doby přesuňte měřící zařízení do úrovni hlavy.

a. Postačí zařízení držet jednou rukou v požadované poloze.

5. Testovaná osoba zavře oči.
6. Testovaná osoba se pokusí udržet rovnováhu po dobu parametru time.
7. Po uplynutí doby je možno zařízením volně pohybovat bez ovlivnění měření.
8. Pro export dat do textového souboru stiskněte tlačítko „Export“.
9. Pojmenujte soubor.

4 PC APLIKACE

Aplikace je naprogramovaná v jazyce C# na architektuře .NET 6.0 pomocí frameworku Windows Forms. Aplikace je sestavena tak, aby umožňovala načtení souřadnic bodů na trajektorii určené polohou hlavy vyšetřované osoby a poté vykreslila Rombergovu křivku. Nástrojem pro její vykreslení se stanou kubické Beziérovy splajny. Samostatným problémem je konstrukce undulačního válce podle děl Vladislava Mirvalda, jehož osou je Rombergova křivka. Válec bude vytvořen pomocí projekcí kružnic v rovině kolmé na osu válce v jednotlivých disktrétních bodech daných měření polohy.

Součástí grafického rozhraní jsou možnosti pro uložení snapshot vzniklého uměleckého díla. Aplikace je vytvořena pomocí nástroje Visual Studio 2022.

4.1 C#

Jazyk C# je objektově orientovaný, silně typový jazyk. C# je založen na jazyku C a je podobný jazykům C, C++, Java a JavaScript. C# je orientovaný na komponenty. Programy v tomto jazyce běží na .NET architektuře. Architektura .NET umožňuje jazyku C# využívat mnohé knihovny, usnadňující tvorbu aplikací. [9], [10]

4.2 Popis aplikace

Aplikace (Obr. 7) se skládá ze tří částí. Na levé straně okna a zabírající většinu místa je plátno. Plátno zobrazuje křivku a kružnice/kruhy, podle vybraných parametrů. V pravé horní části je nastavení. Zde je možné vybrat způsob zadání bodů, které objekty se budou zobrazovat a jak se vykreslí kruhy/kružnice. V levé dolní části se zobrazuje seznam bodů, které určily vykreslenou křivku – osu undulačního válce.

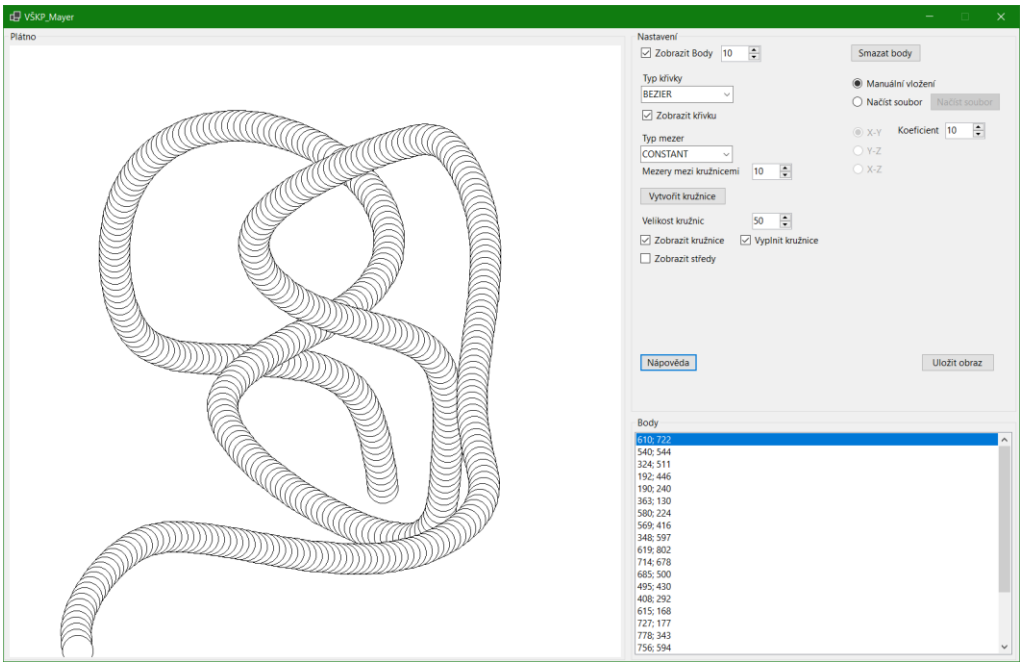
4.3 Popis nastavení

- Zobrazit body
    - Zobrazuje body jako modré kruhy, velikost kruhu je možné změnit

- Typ Křivky
  - Určuje, jak se vykreslí křivka, jsou zde dvě možnosti: LINE a BEZIER
  - LINE vykreslí úsečky mezi body, lze také nazvat Lineární Beziérovou křivkou
  - BEZIER vykreslí Beziérovu kubickou křivku mezi body
- Zobrazit křivku
  - Zobrazí černě vykreslené úsečky/křivky
- Typ mezer
  - Určuje způsob vykreslení kruhů, jsou zde dvě možnosti: CONSTANT a SPEED
  - CONSTANT vykreslí kružnice s konstantní mezerou
  - SPEED vykreslí mezi body uvedený počet kružnic, znázorňuje rychlost pohybu
  - Číslo pod tímto nastavením udává velikost mezer nebo počet kružnic
- Tlačítko Vytvořit kružnice
  - Vygeneruje kružnice podle nastavení
- Velikost kružnic
  - Udává velikost vykreslených kružnic
- Zobrazit kružnice
  - Zobrazí kružnice
- Vyplnit kružnice
  - Vyplní kružnice plnou barvou

- Zobrazit středy
    - Zobrazí červenými kruhy středy kružnic
- Tlačítko Smazat body
    - Vynuluje aktuální body, vyčistí plátno
- Manuální vložení / Načíst soubor
    - Určuje styl poskytnutí dat
    - U manuálního vložení se vkládají body klikáním na plátno
    - Načíst soubor umožňuje použít data z měření aplikace HeadMovementCatcher
    - Při načteném souboru se určuje startovní bod kliknutím na plátno.
- X-Y/Y-Z/X-Z
    - Při načteném souboru určuje, které souřadnice se používají pro zobrazení
- Koeficient
    - U načteného souboru ovlivňuje velikost křivky
- Nápověda
    - Otevře dokument s nápovědou
- Uložit obraz
    - Uloží obraz na disk

Ukázky nastavení jsou ukázány v 45.



Obr. 7 Okno aplikace. Zdroj vlastní

4.4 Body křivky

Pro vykreslení křivky je potřeba zadat body získané z polohy hlavy vyšetřované osoby. Aktuální body se ukládají do proměnné listOfPoints. Aplikace podporuje dvě možnosti. Manuální zadání bodů a načtení bodů ze souboru. Aplikace dokáže zachytit kliknutí myší na plátno (událost Click). Pokud je nastaveno manuální zadávání, je přidán do proměnné listOfPoints bod, znázorňující pozici kurzoru při kliknutí. Seznam těchto bodů udává křivku.

Pokud je nastaveno Načtení ze souboru, kliknutím na plátno určíme, kde se bude nacházet první bod křivky. Při kliknutí na tlačítko Načíst soubor se otevře dialogové okno pro výběr souboru. Je očekáván textový soubor, kde každý bod je na svém řádku ve formátu „*X*;*Y*;*Z*“. Příklad „12.053;16.16;0.005“. Načtení souboru je řešeno v metodě OpenFileDialog_FileOk.

```
private void OpenFileDialog_FileOk(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            try
            {
                StreamReader sr = new StreamReader(openFileDialog.FileName);
                string line;
```

```
                listOf3DPoints = new List<Point3D>();
                while ((line = sr.ReadLine()) != null)
                {
                    string[] cords = new string[3];
                    line = line.Replace('.', ',');
                    cords = line.Split(';');
                    Point3D p = new Point3D();
                    p.X = float.Parse(cords[0]);
                    p.Y = float.Parse(cords[1]);
                    p.Z = float.Parse(cords[2]);
                    listOf3DPoints.Add(p);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Nelze    načíst    soubor.\n\nError    message:
{ex.Message}");
                throw;
            }
            GeneratePoints(null, null);
        }
```

4.5 Zobrazení křivky

Aplikace podporuje dva typy křivek. Je-li nastavena LINE křivka, je křivka vykreslena pomocí úseček mezi body. Pokud je nastavena BEZIER křivka, je pomocí splajnové interpolace vykreslena Beziérova kubická křivka. Mezi jednotlivými body je vykreslena Beziérova kubická křivka. K hladkému slepení křivek, je potřeba využít A-rámce. K výpočtu je využita B-splajna.

Aplikace dostane body, kterými musí procházet. K vykreslení jedné Beziérovy kubické křivky jsou potřeba čtyři body: začátek křivky S0, konec křivky S3 a dva kontrolní body S1, S2 určující směr křivky. Body S0 a S3 jsou body, kterými musí procházet a jsou známé. Body S1 a S2 jsou body, které nejsou známé a je nutné je dopočítat. Prvním krokem je vypočítání B-splajnových kontrolních bodů. Vzorec pro vztah B-splajnových kontrolních bodů (Bn) a bodech, kterými prochází křivka (Sn) pro výpočet při 5 bodech.

$$\begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix} = \begin{bmatrix} (6S_1 - S_0) \\ 6S_2 \\ 6S_3 \\ (6S_4 - S_5) \end{bmatrix}$$

Pro výpočet B-splajnových kontrolních bodů je vytvořena rozšířená matice [M|S], kde M je „1 4 1 matice“ a S je matice Sn bodů. Tato matice je pomocí

Gaussových operací převedena do tvaru [I|Bn]. B pravé části matice se budou nacházet body B-splajny.

V aplikace tento výpočet obstarává metoda GetControlPointsForBezier v souboru Calculator.cs.

```
private static PointF[] GetControlPointsForBezier(BindingList<Point>
listOfPoints)
        {
            PointF[] ControlPoints = new PointF[listOfPoints.Count-2];

            int numRows = listOfPoints.Count - 2;
            int numCols = listOfPoints.Count;

            double[,] matrix = new double[numCols,numRows];

            //fill matrix (last two columns are control points based values)
            for (int y = 0; y < numRows; y++)
            {
                for (int x = 0; x < numCols; x++)
                {
                    //fill first matrix 1-4-1 diagonally
                    if (x==y)
                    {
                        matrix[x, y] = 4;
                        if (x-1>=0)
                        {
                            matrix[x - 1, y] = 1;
                        }
                        matrix[x+1, y] = 1;
                    }

                    //last two columns are point based
                    if (x==numCols-2)
                    {
                        if (y==0) // first row have different value
                        {
                            matrix[x, y] = (6 * listOfPoints[1].X) -
listOfPoints[0].X;
                        }
                        else if (y == numRows-1) // last row have different value
                        {
                            matrix[x, y] = (6 * listOfPoints[numCols-2].X) -
listOfPoints[numCols-1].X;
                        }
                        else
                        {
                            matrix[x, y] = 6 * listOfPoints[y+1].X;
                        }
                    }
                    if (x == numCols - 1)
                    {
                        if (y == 0) // first row have different value
                        {
                            matrix[x, y] = (6 * listOfPoints[1].Y) -
listOfPoints[0].Y;
                        }
                        else if (y == numRows - 1) // last row have different
value
```

```
                        {
                            matrix[x, y] = (6 * listOfPoints[numCols - 2].Y) -
listOfPoints[numCols-1].Y;
                        }
                        else
                        {
                            matrix[x, y] = 6 * listOfPoints[y + 1].Y;
                        }
                    }
                }
            }

            //row-reduce matrix

            for (int y = 0; y < numRows; y++)
            {
                // find non-zero element
                int elementRow = y;
                while (elementRow < numRows && matrix[y, elementRow] == 0)
                {
                    elementRow++;
                }

                // no element position found
                if (elementRow == numRows)
                {
                    continue;
                }

                // swap rows if element position is zero
                if (elementRow != y)
                {
                    for (int x = y; x < numCols; x++)
                    {
                        double temp = matrix[x, y];
                        matrix[x, y] = matrix[x, elementRow];
                        matrix[x, elementRow] = temp;
                    }
                }

                // change the element to 1
                double element = matrix[y, y];
                for (int x = y; x < numCols; x++)
                {
                    matrix[x, y] /= element;
                }

                // create zeroes on each row
                for (int yy = 0; yy < numRows; yy++)
                {
                    //skip element row
                    if (yy == y)
                    {
                        continue;
                    }
                    double factor = matrix[y, yy];
                    for (int x = y; x < numCols; x++)
                    {
                        matrix[x, yy] -= factor * matrix[x, y];
                    }
                }
            }
```

```
            //last two columns have control points
            for (int y = 0; y < numRows; y++)
            {
                ControlPoints[y] = new PointF(Convert.ToSingle(matrix[numCols -
2,y]), Convert.ToSingle(matrix[numCols - 1, y]));
            }

            return ControlPoints;
        }
```

Dalším krokem je vypočet kontrolních bodů Beziérovích křivek. Kontrolní body Beziérových křivek odpovídají bodům v první a druhé třetině úseček mezi B-splajnovými body.

Body Bi jsou kontrolní body B-splajny

Body Si jsou body, kterými má procházet křivka

B0 = S0 & Bn = Sn

Mějme kontrolní body Ci;1 a Ci;2

Ci;1 = Bi + (Bi+1-Bi) / 3

Ci;2 = Bi + 2(Bi+1-Bi) / 3

V aplikaci se o výpočet kontrolních bodů stará metoda GetBezierPoints v souboru Calculator.cs. Samotná Beziérova křivka je poté vykreslena pomocí metody DrawBeziers, která je obsažena v knihovně .NET.

4.6 Polohy kružnic

K zobrazení kružnic je potřeba znát, kde jsou středy kružnic. Uživatel může ovlivnit vzdálenost mezi jednotlivými kružnicemi. Má na výběr 2 způsoby mezer. U způsobu CONSTANT je aplikována konstantní mezera mezi všemi kružnicemi. Velikost mezery je dána v pixelech. Výpočet pro generovaní středů kružnic obstarává metoda GetPointsOnLine v souboru calculator.cs.

```
Point point = new Point();
                        double distance;
```

```
                        //check distance
                        while (true)
                        {
                            distance = Math.Sqrt(Math.Pow(nextPoint.X -
lastPoint.X, 2) + Math.Pow(nextPoint.Y - lastPoint.Y, 2));
                            if (currentSpace < distance)
                            {
                                //We are inside the line
                                break;
                            }
                            else
                            {
                                //The line is too short -> move to next point
                                //check if next point exists
                                if (points.Count > index)
                                {
                                    lastPoint = points[index - 1];
                                    nextPoint = points[index];
                                    index++;
                                    //Shorten the space
                                    currentSpace = currentSpace - distance;
                                }
                                else
                                {
                                    //no point exist -> Leave
                                    yield break;
                                }

                            }
                        }

                        //Find the point
                        double ratio = currentSpace / distance;

                        point.X = Convert.ToInt32((1 - ratio) * lastPoint.X +
ratio * nextPoint.X);
                        point.Y = Convert.ToInt32((1 - ratio) * lastPoint.Y +
ratio * nextPoint.Y);

                        yield return point;
                        lastPoint = point;
```

currentSpace = spacing;

Tato metoda se používá jak u křivky typu LINE, tak i křivky typu BEZIER. U křivky typu BEZIER je ale proveden krok navíc, kde se v programu BEZIER křivka převede na sérii úseček (křivku typu LINE).

```
        internal  static  IEnumerable<Point>  GetPointsOnCurve(BindingList<Point>
listOfPoints, int spacing, SpacingType spacingType)
        {
            //Convert to lines -> then use the previous method

            GraphicsPath path = new GraphicsPath();

            path.AddCurve(listOfPoints.ToArray());
```

```
            Matrix mx = new Matrix();
            mx.Translate(0, 1);

            path.Flatten(mx, 0.1f);

            BindingList<Point> linePoints = new BindingList<Point>();
            foreach (var pointF in path.PathPoints)
            {
                linePoints.Add(Point.Round(pointF));
            }

            return GetPointsOnLine(linePoints, spacing, spacingType);
    }
```

Pokud uživatel vybral metodu SPEED, jsou kružnice vygenerovány jinak. Mezi každými dvěma zadanými body se vygeneruje zadaný počet kružnic. V měření pomocí aplikace HeadMovementCatcher se body ukládají konstantně po *x* milisekundách. Z toho lze odvodit, že pokud jsou body blíže u sebe, byl v tomto úseku pohyb hlavy rychlejší než v úseku, kde jsou body dál od sebe. Toto zobrazení může znázornit proměnlivou rychlost pohybu hlavy během měření.

Pokud máme křivku typu LINE, je výpočet jednoduchý. Jednotlivé úsečky se rozdělí podle počtu kružnic.

```
while (true)
                    {
                        Point point = new Point();

                        double distance;
                        distance = Math.Sqrt(Math.Pow(nextPoint.X - lastPoint.X,
2) + Math.Pow(nextPoint.Y - lastPoint.Y, 2));
                        double part = distance / (spacing+1);
                        for (int i = 0; i < spacing; i++)
                        {
                            double ratio = (part*(i+1)) / distance;

                            point.X = Convert.ToInt32((1 - ratio) * lastPoint.X +
ratio * nextPoint.X);
                            point.Y = Convert.ToInt32((1 - ratio) * lastPoint.Y +
ratio * nextPoint.Y);

                            yield return point;
                        }

                        if (points.Count > index)
                        {
                            lastPoint = points[index - 1];
                            nextPoint = points[index];
                            index++;
                            yield return lastPoint;
```

```
                    }
                    else
                    {
                        //no point exist -> Leave
                        yield break;
                    }

                }
```

Pokud je nastavena křivka typu BEZIER, je k výpočtu použit vzorec Beziérovy kubické křivky:

$$C(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} P_i$$

$$= (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

P_0, P_1, P_2, P_3 jsou body označující Beziérovu křivku,

t je parametr s hodnotou od 0 do 1.

K dosažení vybraného počtu kružnic na křivce stačí najít násobky hodnoty t.

```
internal static IEnumerable<Point> GetPointsOnBezier(PointF[] points, int
spacing)
        {
            yield return Point.Round(points.First());
            double currentSpace = spacing;
            Point lastPoint = Point.Round(points[0]);
            Point controlPoint1 = Point.Round(points[1]);
            Point controlPoint2 = Point.Round(points[2]);
            Point nextPoint = Point.Round(points[3]);
            int index = 4;
            double ratio = 1 / (Convert.ToDouble(spacing)+1);

            while (true)
            {
                Point point = new Point();
                //find new center
                for (int i = 0; i < spacing; i++)
                {
                    double t = ratio * (i + 1);
                    point.X = Convert.ToInt32(Math.Pow(1 - t, 3) * lastPoint.X +
3 * t * Math.Pow(1 - t, 2) * controlPoint1.X + 3 * t * t * (1 - t) *
controlPoint2.X + t * t * t * nextPoint.X);
                    point.Y = Convert.ToInt32(Math.Pow(1 - t, 3) * lastPoint.Y +
3 * t * Math.Pow(1 - t, 2) * controlPoint1.Y + 3 * t * t * (1 - t) *
controlPoint2.Y + t * t * t * nextPoint.Y);

                    yield return point;

                }

                //move to next segment
                if (points.Length > index)
```

```
            {
                lastPoint = Point.Round(points[index - 1]);
                controlPoint1 = Point.Round(points[index]);
                controlPoint2 = Point.Round(points[index + 1]);
                nextPoint = Point.Round(points[index + 2]);
                index+=3;
                yield return lastPoint;
            }
            else
            {
                //no point exist -> Leave
                yield break;
            }

        }
    }
```

ZÁVĚR

Cílem této práce bylo vytvořit aplikaci pro napodobení Mirvaldových obrazů vycházejících z Rombergových křivek. V úvodu práce byl shrnut život a dílo českého pedagoga a malíře Miroslava Mirvalda ve snaze demonstrovat, jak významné bylo v jeho tvorbě kombinování geometrického řádu a logiky s náhodností reálného světa. Jedním z jeho projektů bylo využití Rombergových křivek pro konstrukci undulačních válců.

Pomocí vytvořené mobilní aplikace bylo možné zaznamenat pohyb hlavy při Rombergově testu. Pomocí těchto dat bylo možné v PC aplikaci zkonstruovat Rombergovu křivku. K vytvoření křivky byla použita interpolace diskrétních měřených bodů kubickou Beziérovou křivkou. Následovala konstrukce undulačního válce tvořeného kružnicemi v rovině kolmé k Rombergově křivce.

Téma jsem zvolil, protože mě zaujala kombinace umění, matematiky, náhodnosti a počítačové grafiky.

POUŽITÁ LITERATURA

[1] POSPISZYL, Tomáš. Vladislav Mirvald. [V Řevnicích]: Arbor vitae, c2010. ISBN 978-80-87164-58-7.

[2] Vladislav Mirvald. Artlist: Centrum pro současné umění Praha [online]. Praha: Centrum pro současné umění Praha, c2006–2023 [cit. 2023-05-12]. Dostupné z: https://www.artlist.cz/vladislav-mirvald-108623/

[3] Khasnis A, Gokula R M. Romberg's test. J Postgrad Med [serial online] 2003 [cit. 2023-05-12]. Dostupné z: https://www.jpgmonline.com/text.asp?2003/49/2/169/894

[4] Forbes J, Munakomi S, Cronovich H. Romberg Test. [Upraveno 12. února 2023]. StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; Leden 2023-. Dostuné z: https://www.ncbi.nlm.nih.gov/books/NBK563187/

[5] Kotlin overview. Developers android [online]. Mountain View, Kalifornie, USA: Google, © 2023 [cit. 2023-05-13]. Dostupné z: https://developer.android.com/kotlin/overview

[6] HELLER, Martin. What is Kotlin? The Java alternative explained. InfoWorld [online]. 2022 [cit. 2023-05-12]. Dostupné z: https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html

[7] Kotlin (programovací jazyk). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-13]. Dostupné z: https://cs.wikipedia.org/wiki/Kotlin_(programovac%C3%AD_jazyk)

[8] Kotlin overview. Sensor Coordinate System [online]. Mountain View, Kalifornie, USA: Google, © 2023 [cit. 2023-05-13]. Dostupné z: https://developer.android.com/kotlin/overview

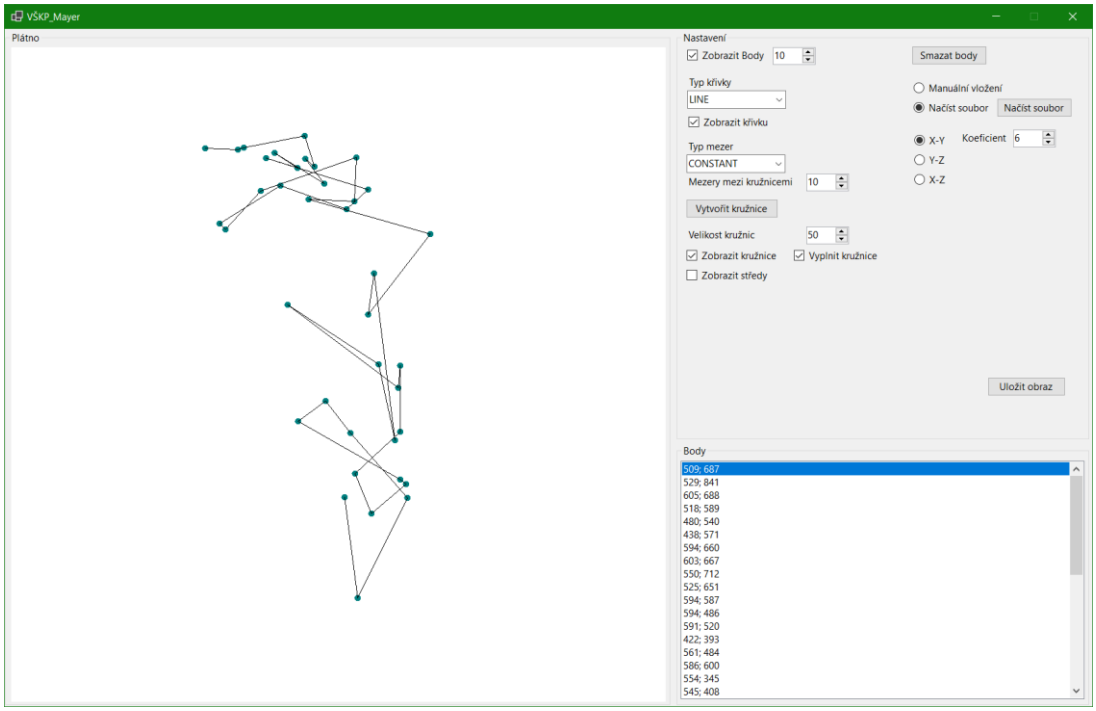
[9] A tour of the C# language. Microsoft Learn [online]. Redmond, Washington, USA: Microsoft Corporation, c2023 [cit. 2023-05-13]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[10] C Sharp. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-13]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp

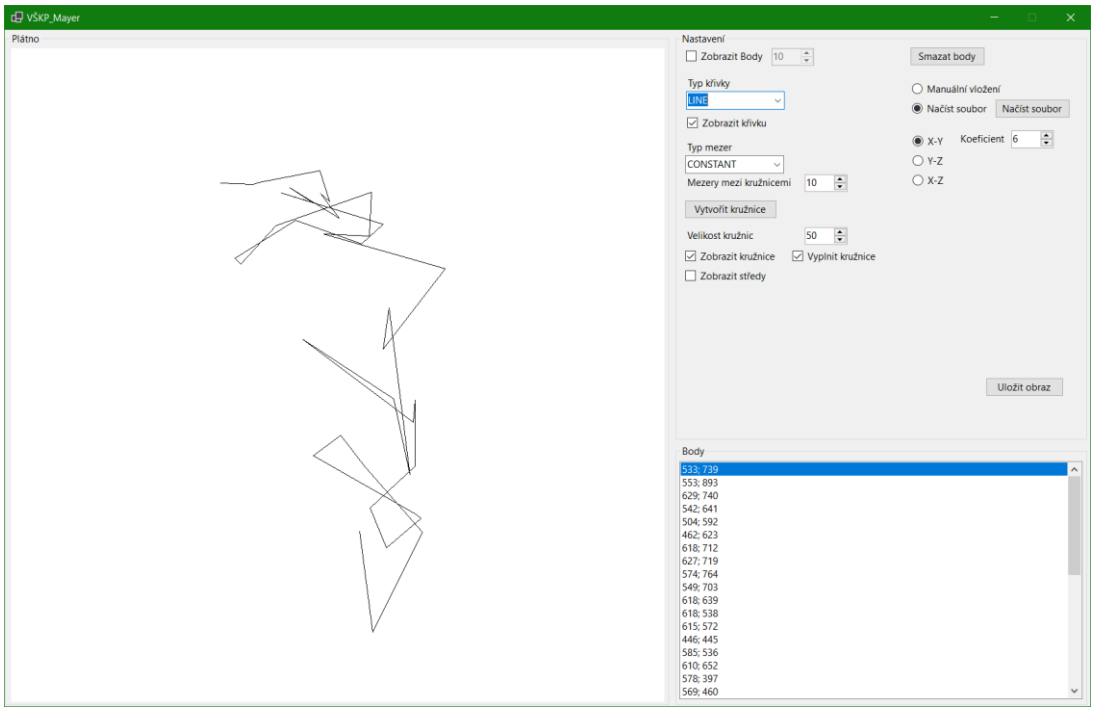
PŘÍLOHY

PŘÍLOHA A – ÚKÁZKY NASTAVENÍ APLIKACE

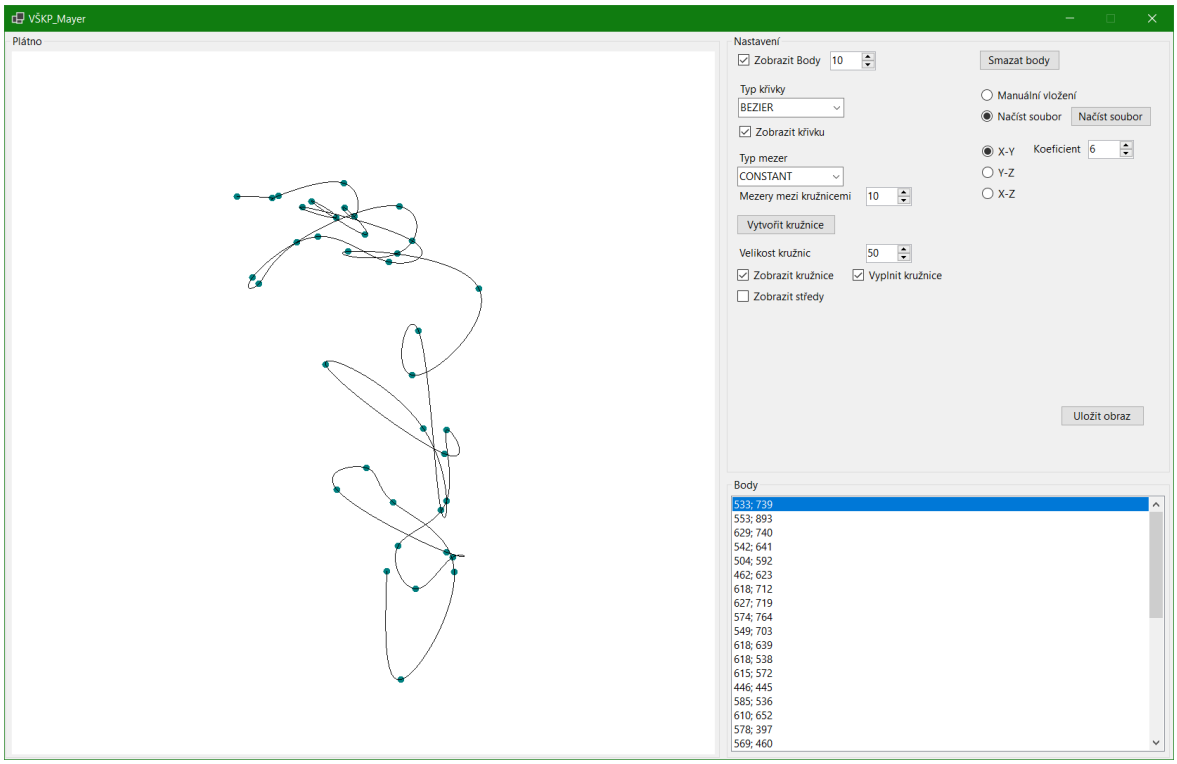
Kolekce obrázků zobrazující různá nastavení a jak mění výsledný obraz.



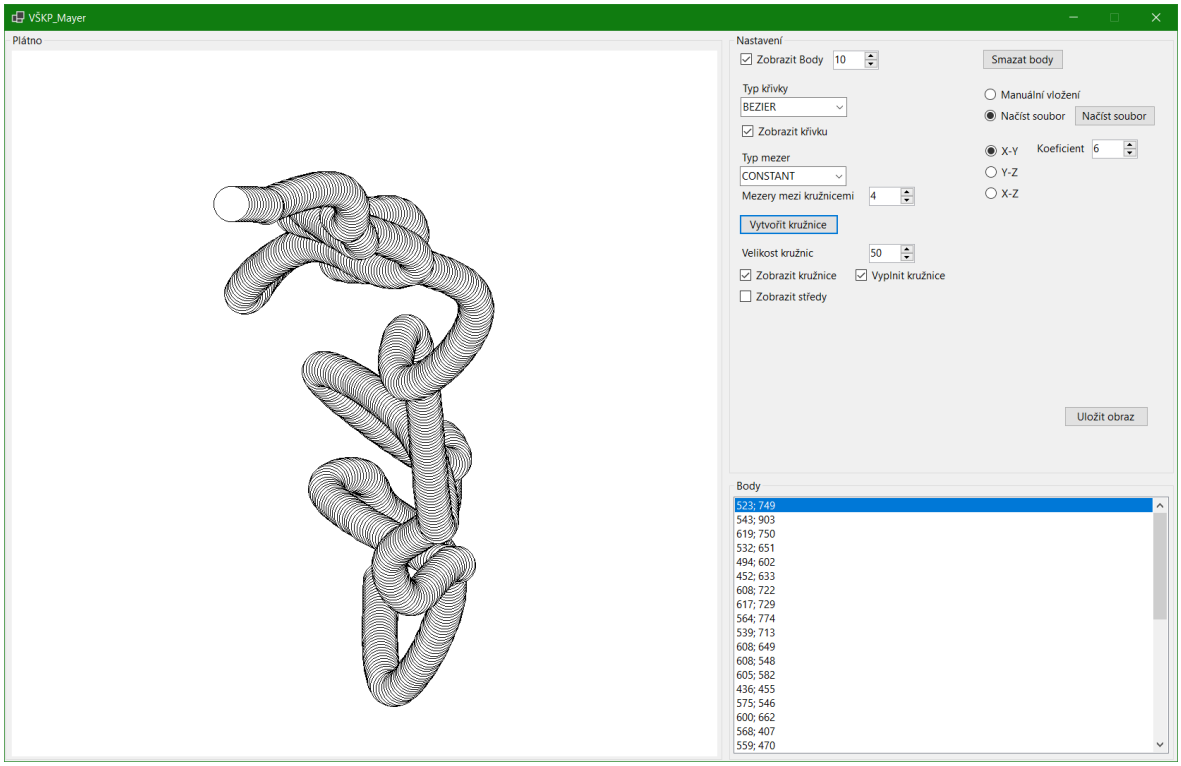
Obr. 8 LINE křivka se zobrazenými body. Zdroj vlastní



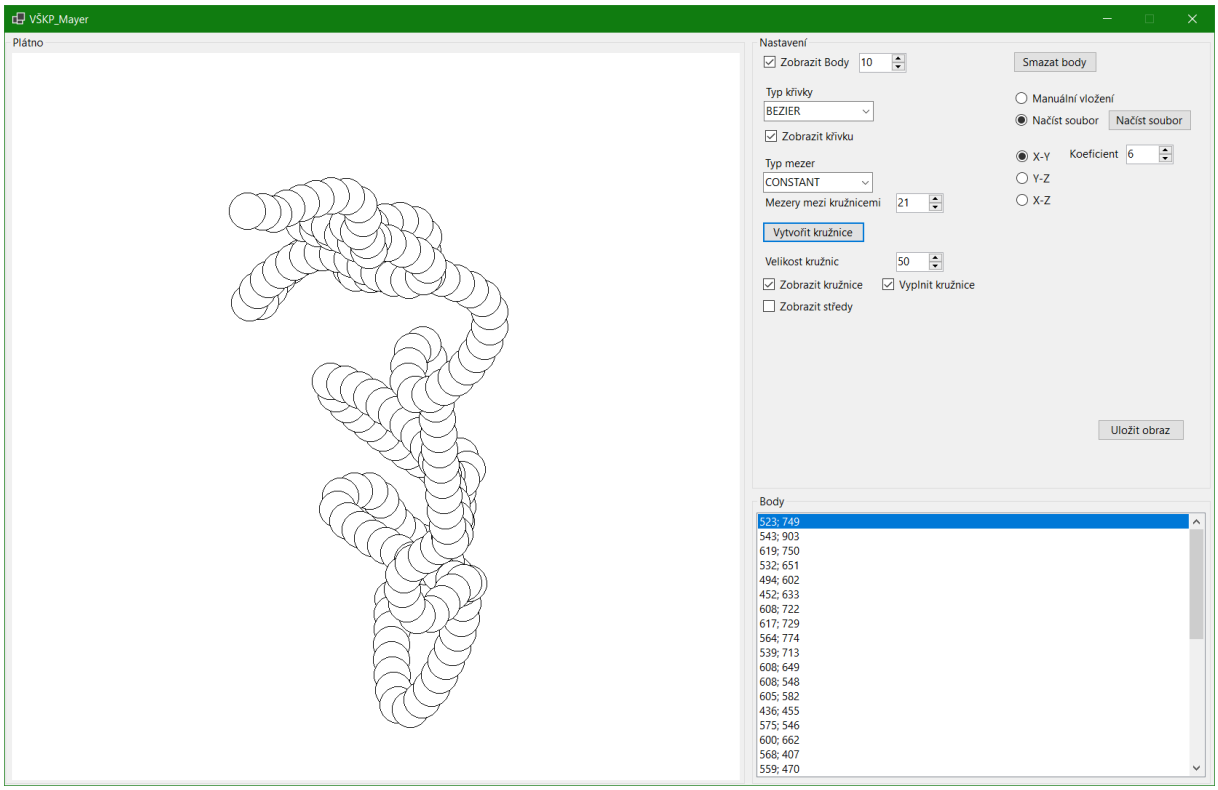
Obr. 9 LINE křivka bez zobrazených bodů. Zdroj vlastní



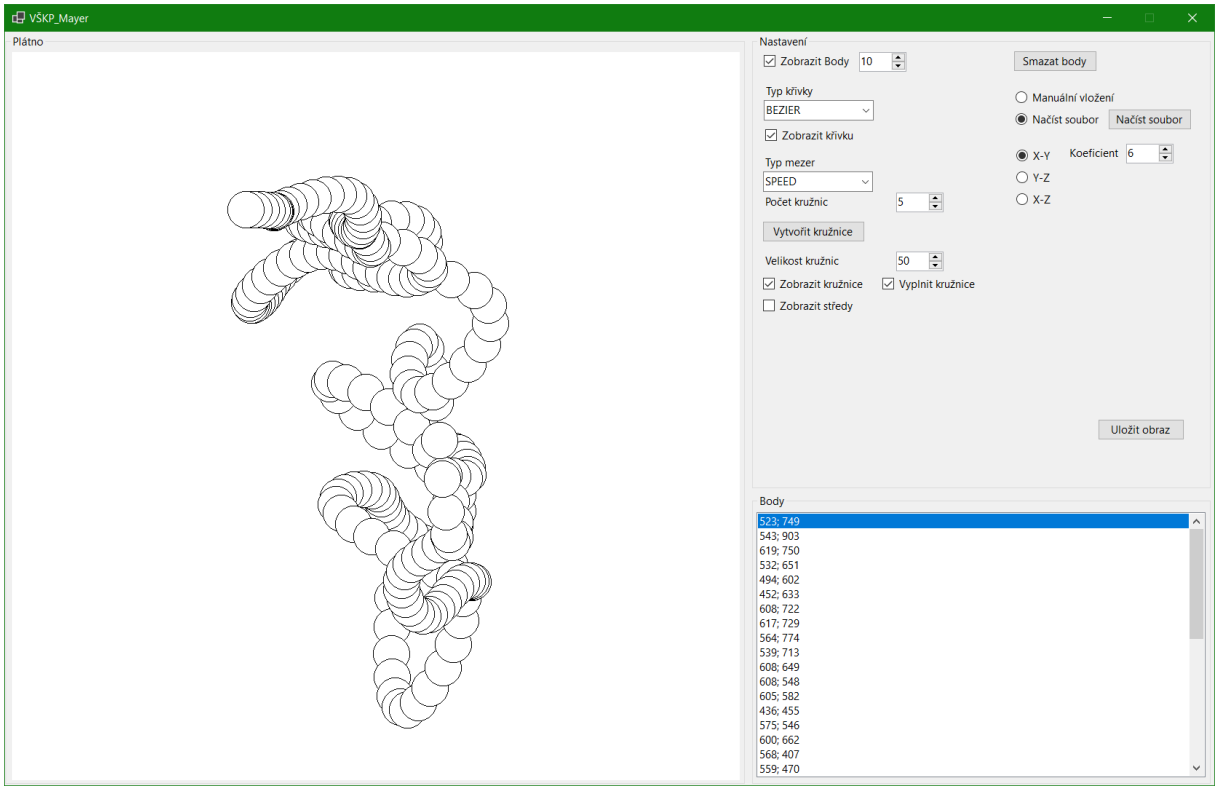
Obr. 10 BEZIER křivka se zobrazenými body. Zdroj vlastní



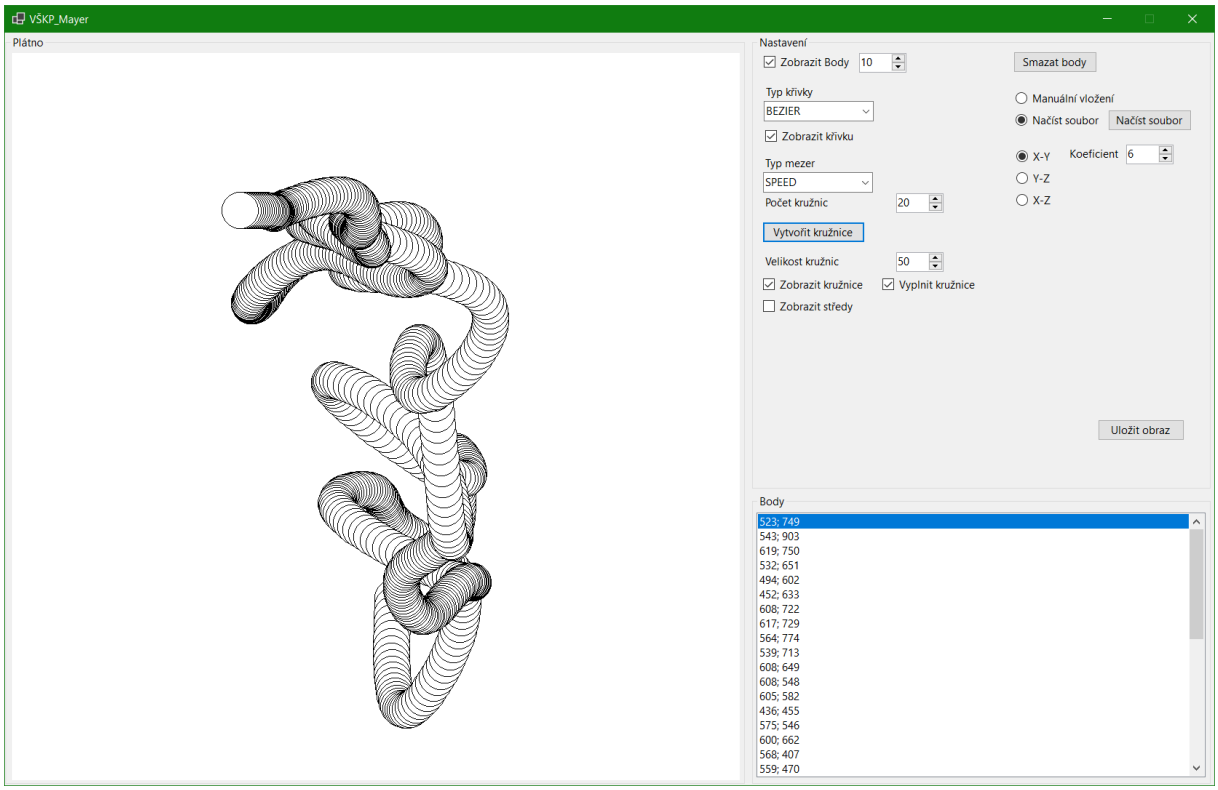
Obr. 11 CONSTANT mezery o vzdálenosti 4. Zdroj vlastní



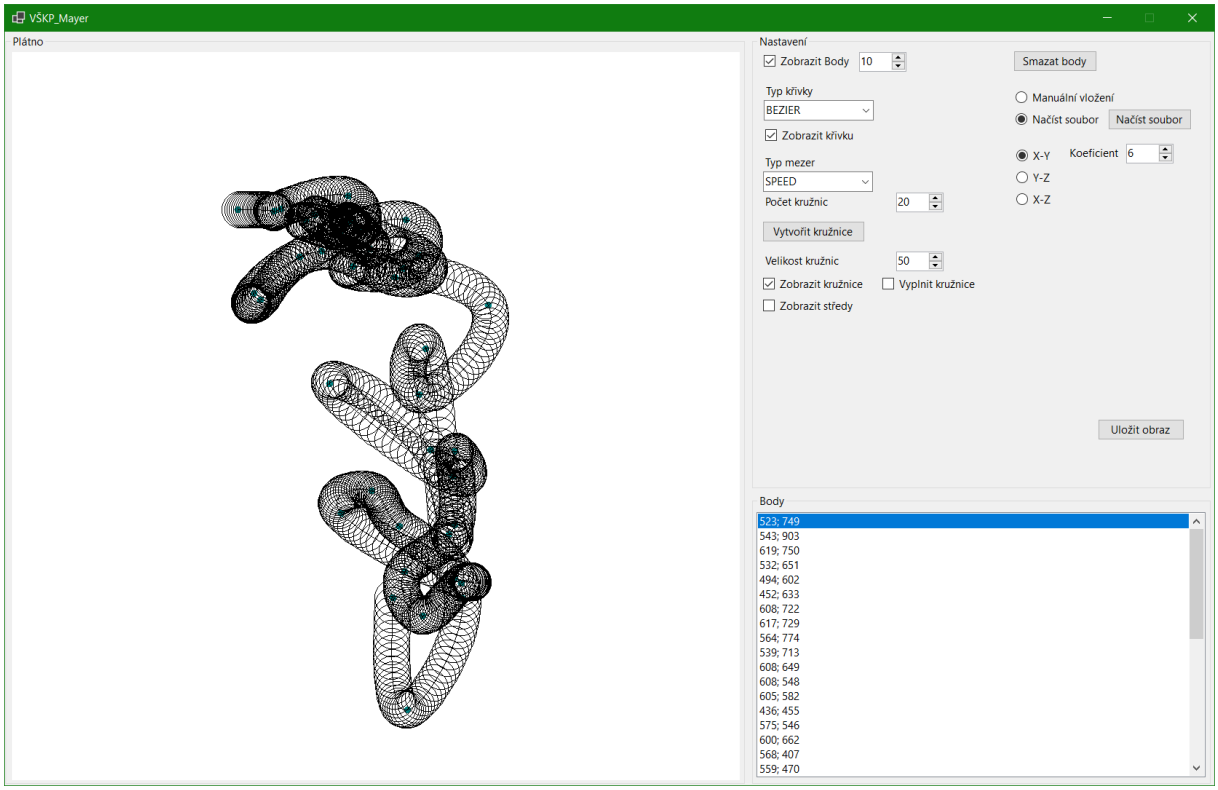
Obr. 12 CONSTANT mezery o vzdálenosti 21. Zdroj vlastní



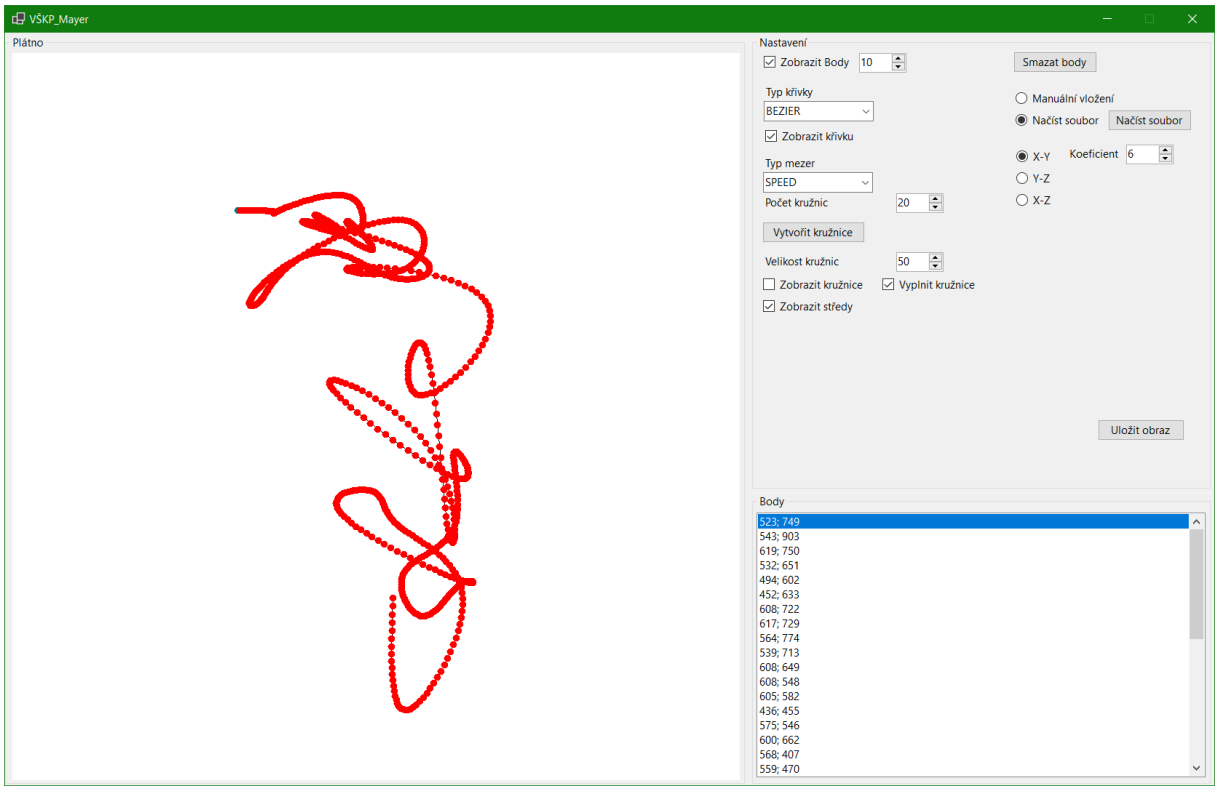
Obr. 13 SPEED mezery o počtu 5. Zdroj vlastní



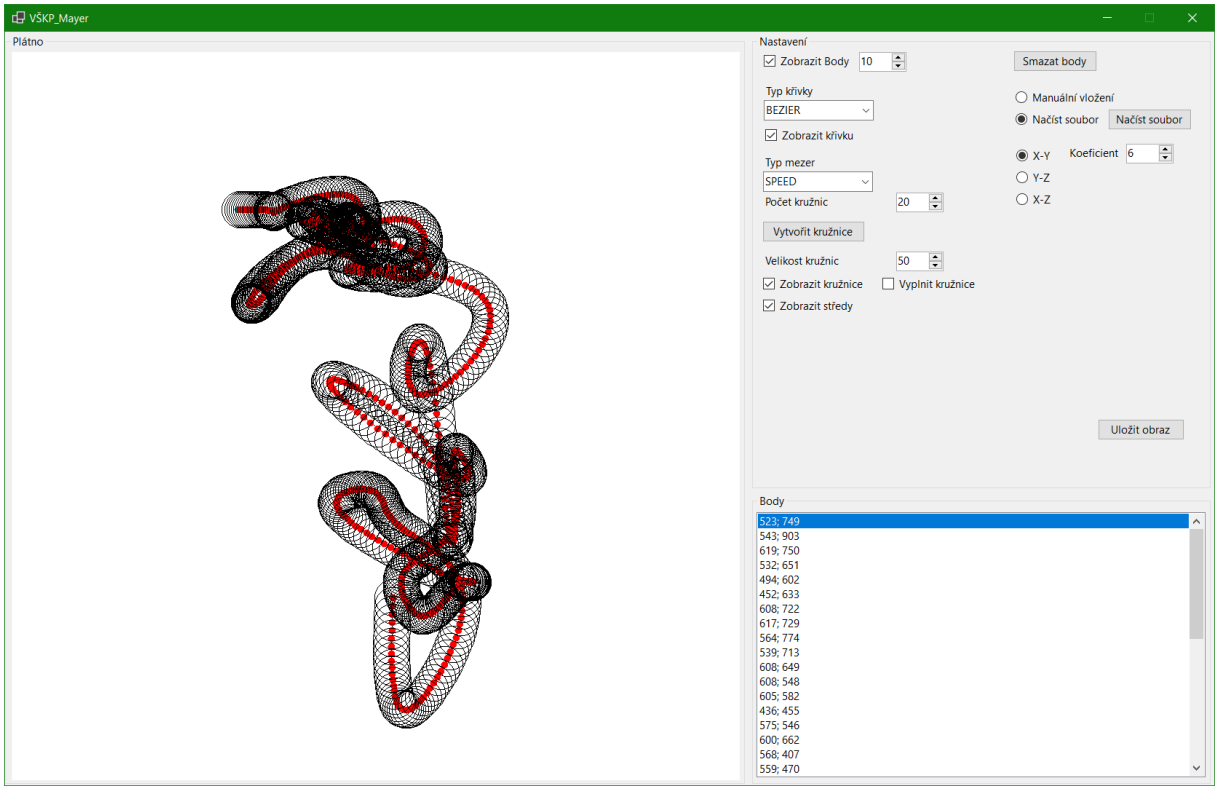
Obr. 14 SPEED mezery o počtu 20. Zdroj vlastní



Obr. 15 Zobrazení bez vyplněných kružnic. Zdroj vlastní

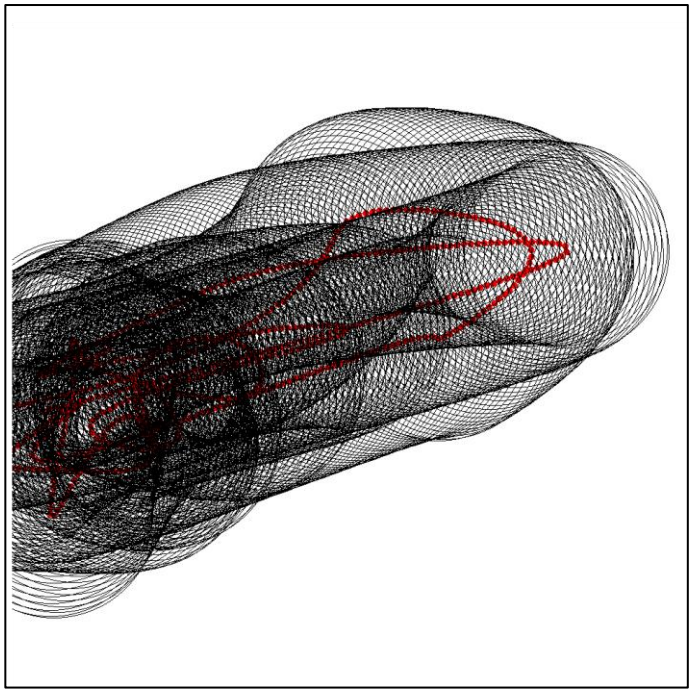


Obr. 16 Zobrazení středů kružnic. Zdroj vlastní

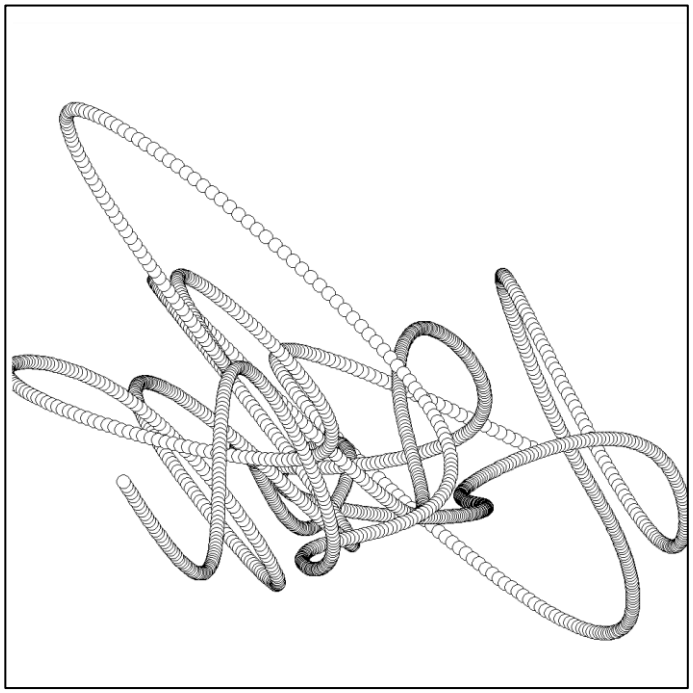


Obr. 17 Kombinace zobrazení středů kružnic a nevyplnění kružnic. Zdroj vlastní

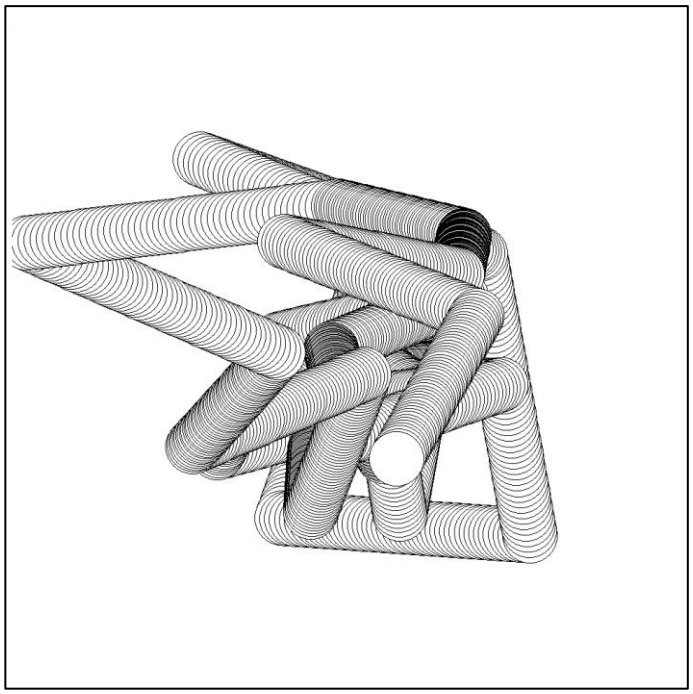
PŘÍLOHA B – VYGENEROVANÉ UNDULAČNÍ VÁLCE



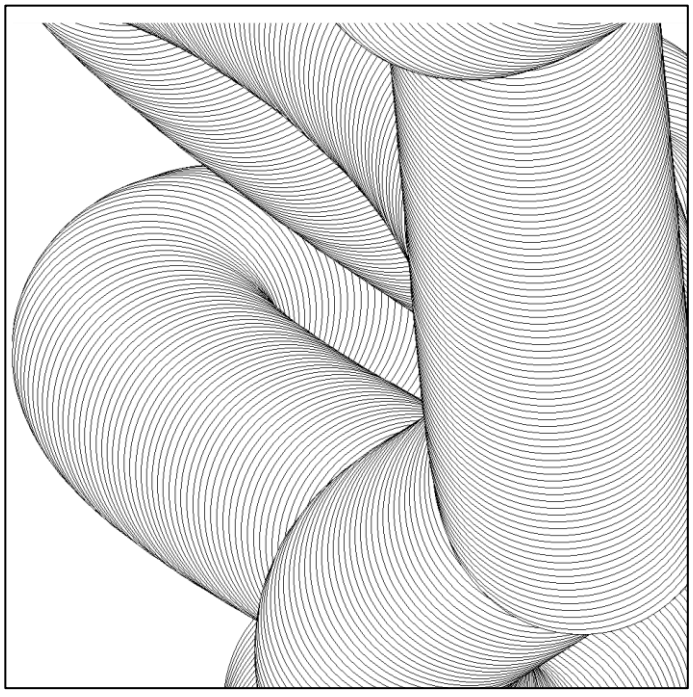
Obr. 18 Zobrazení středů, nevyplněné kružnice. Zdroj vlastní



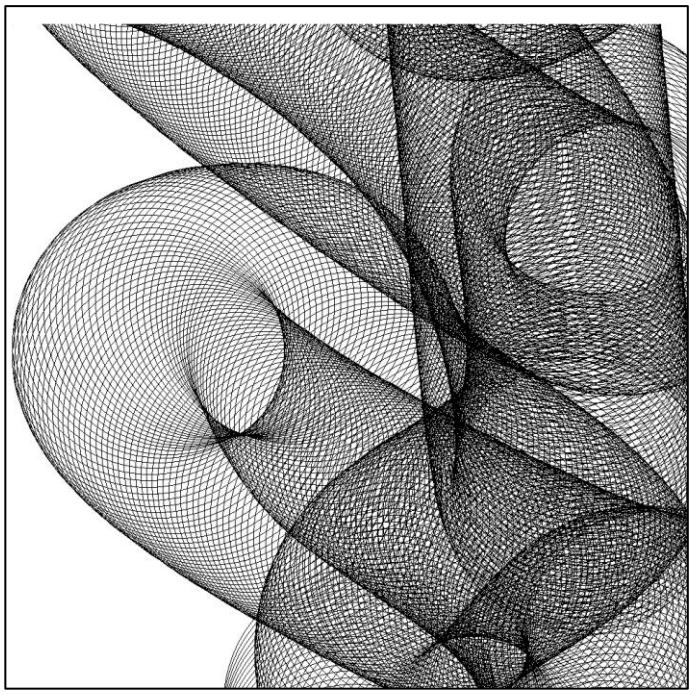
Obr. 19 SPEED mezery, velikost kružnic 10. Zdroj vlastní



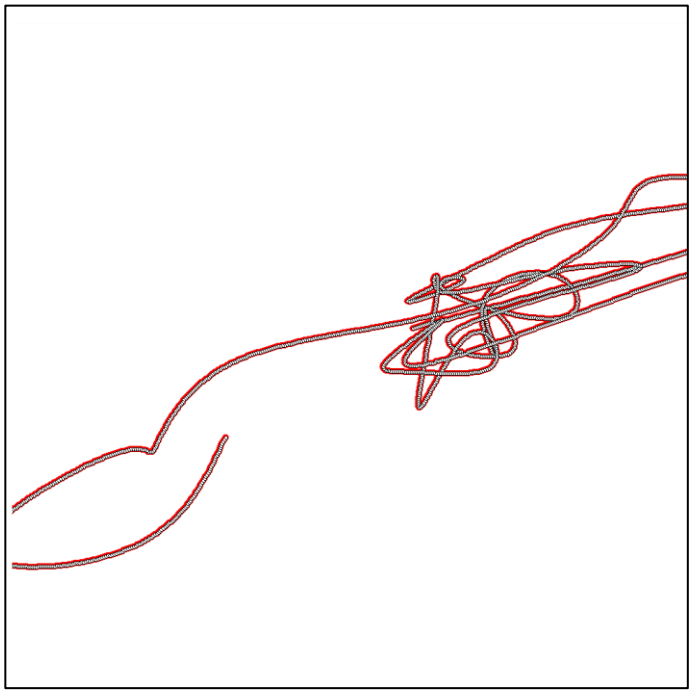
Obr. 20 LINE křivka. Zdroj vlastní



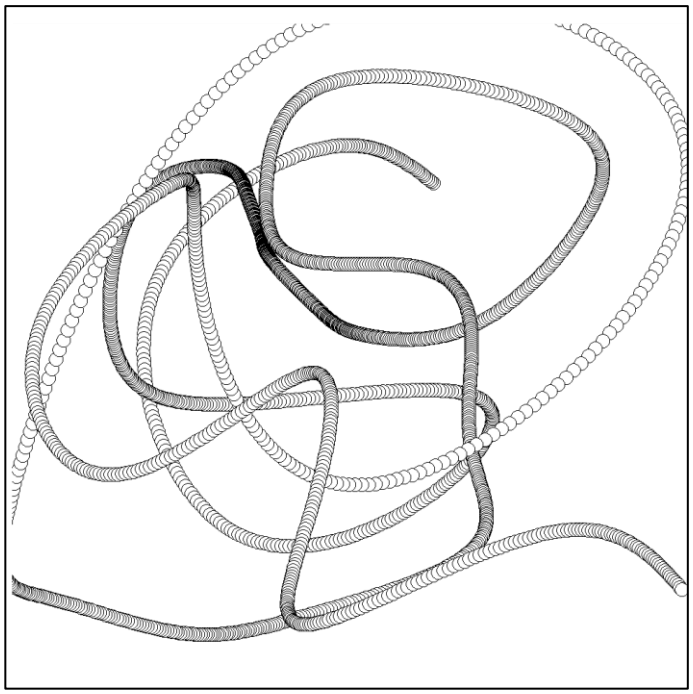
Obr. 21 Velikost kružnic 500. Zdroj vlastní



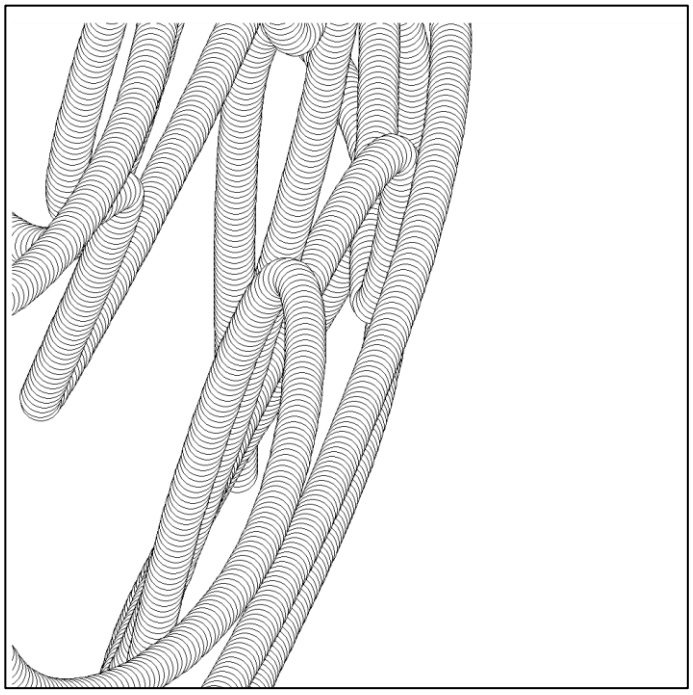
Obr. 22 Velikost kružnic 500, nevyplněné kružnice. Zdroj vlastní



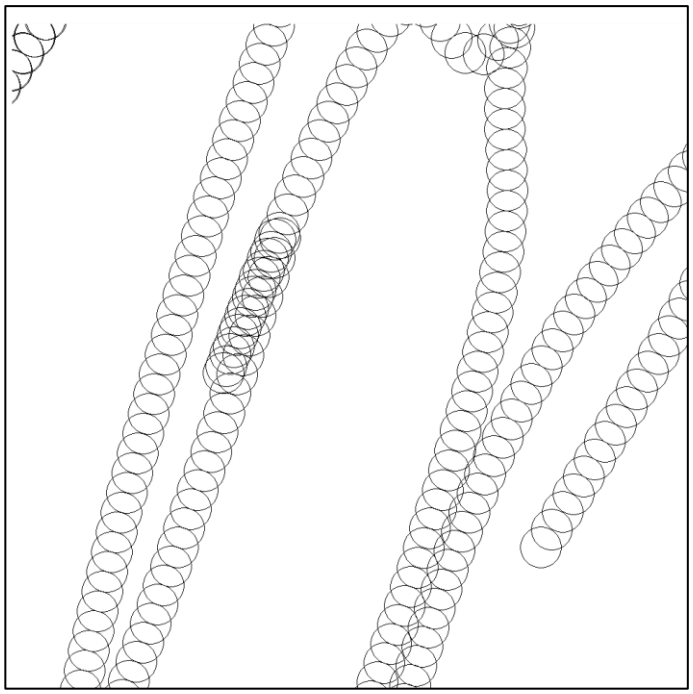
Obr. 23 Velikost kružnic 5, zobrazené středy kružnic. Zdroj vlastní



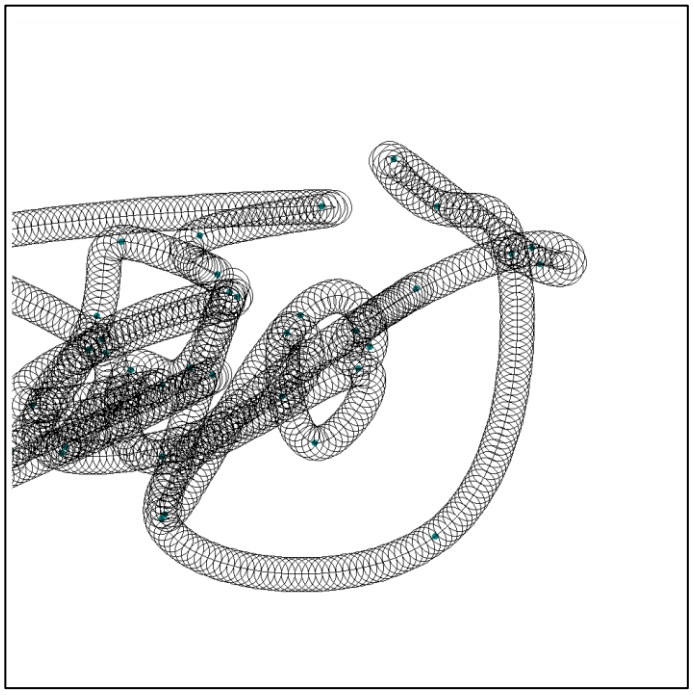
Obr. 24 SPEED křivka. Zdroj vlastní



Obr. 25 CONSTANT mezery. Zdroj vlastní



Obr. 26 CONSTANT mezery o velikosti 30, velikost kružnic 60. Zdroj vlastní



Obr. 27 Nevyplněné kružnice, zobrazená křivka, zobrazené body. Zdroj vlastní