ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vomastek** Jméno: **Michal** Osobní číslo: **435318**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačová grafika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Nástroj pro výuku základních křivek

Název diplomové práce anglicky:

A Tool for Teaching Fundamental Curves

Pokyny pro vypracování:

Diplomová práce se zabývá tvorbou aplikace na podporu výuky základních křivek používaných v počítačové grafice. Toto téma je pro studenty obtížné a cílem aplikace je téma studentům co nejvíce zpřístupnit.
Na základě rešerše, provedené v rámci semestrálního projektu, konkretizujte detailní požadavky na výukovou aplikaci tak, aby pokryla požadavky předmětu Počítačová grafika (PGR) [1]. Navrhněte strukturu aplikace a metodou user centered design ji implementujte jako webovou aplikaci v jazyce JavaScript.
Soustřeďte se zejména na interaktivitu a názornost aplikace, součástí aplikace bude tutoriál s úkoly pro studenty, jehož prostřednictvím se naučí aplikaci ovládat a plynule přejdou k jednotlivým křivkám. Dbejte na snadnou rozšiřitelnost o další křivky, úlohy a tutoriály.
Vytvořte úvodní tutoriál (ovládání aplikace) a sadu tutoriálů pokrývající křivky vyučované v předmětu PGR.
Navrhněte program tak, aby zároveň umožňoval generování jednoduchých úkolů, na kterých si student při samostudiu vyzkouší, zda probíranou látku pochopil.

Seznam doporučené literatury:

[1] Materiály k předmětu Programování grafiky, DCGI FEL ČVUT, Praha.
[2] Žára, J. a kol.: Moderní počítačová grafika, Computer Press, Brno, 2004.
[3] Linkeová, I.: Základy Počítačového modelování křivek a ploch, skripta ČVUT FS, 2008. (kap. 1 a 2)

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Petr Felkel, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **13.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

Ing. Petr Felkel, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

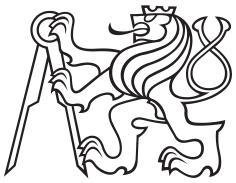
III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Nástroj pro výuku základních křivek

Bc. Michal Vomastek

Vedoucí: Ing. Petr Felkel, Ph.D.
Studijní program: Otevřená informatika
Obor: Počítačová grafika
Květen 2019

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Petru Felkelovi, Ph.D. za konzultace a čas, který mi věnoval. Dále bych chtěl poděkovat svým rodičům, prarodičům, sestře a přátelům za veškerou podporu. Můj velký dík patří také všem účastníkům uživatelského průzkumu a testování.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 20. května 2019

.....

Abstrakt

Diplomová práce se zabývá tvorbou aplikace na podporu výuky základních křivek používaných v počítačové grafice. Toto téma je pro studenty obtížné a cílem aplikace je téma studentům co nejvíce zpřístupnit. Aplikace byla z důvodu dostupnosti navrhnuta jako webová stránka v jazyku JavaScript. Mezi diskutované a implementované křivky patří Coonsova kubika, NURBS, Bézierova křivka, Hermitovské kubiky a Explicitní křivka. Těmto křivkám byly vytvořeny tutoriály, které znázorňují vlastnosti křivek. V tutoriálech je možné měnit křivkám jejich parametry a také souřadnice i počet řídících bodu. Aplikace umožňuje vykreslovat nejen křivky, ale i bázové funkce a jejich derivace.

Klíčová slova: křivky, grafika, web, JavaScript, výuková, aplikace

Vedoucí: Ing. Petr Felkel, Ph.D.

Abstract

This thesis deals with the creation of an application to support the teaching of basic curves used in computer graphics. This topic is difficult for students and the goal of the application is to make it as accessible to students as possible. The application was designed as a JavaScript website for availability purposes. Curves discussed and implemented here include Coons cubic, NURBS, Bezier curve, Hermit cubic, and Explicit curve. Tutorials have been created for these curves to show curve properties. It is possible to change the parameters, the coordinates and the number of control points of curves in the tutorials. The application allows to draw not only curves but also base functions and their derivatives.

Keywords: curves, graphics, web, JavaScript, educational, application

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Křivky mají v počítačové grafice velké využití. Používají se ve 2D i 3D a to například pro definování fontů, určení dráhy pohybujících objektů, vytvoření tvarů těles a další [2]. Pro různé případy jsou vhodné různé křivky. Ve vektorové grafice jsou nejčastěji používané Bézierovy křivky, u kterých uživatel zadává body a tečny v těchto bodech. V případě ploch, nebo u složitějších tvarů křivek (úsečky, elipsy, kružnice) se ale nejčastěji používají obecnější NURBS křivky.

Křivky jsou rozděleny na dva základní druhy a to interpolační, kde křivka prochází všemi body a aproximační. Z interpolačních křivek je často používaná například Catmull-Rom křivka. Z aproximačních je nejznámější Bézierova křivka. Její velkou výhodou je možnost jejího adaptivního vykreslování. Křivku je možné rozdělit na dvě další křivky bez změny tvaru křivky. Takto je možné rekurzivně dělit křivku až do určitého splnění parametru (například křivka je již téměř úsečkou). Tento proces se využije při vykreslování, kde se křivka vykresluje za pomoci úseček. Její tvar je tedy aproximován. Velkou výhodou celého tohoto procesu je také to, že může být prováděn na grafické kartě za pomocí tesselačních shaderů. Využití najde i v počítačových hrách, kde se kvůli velkým nárokům na vykreslování vykreslují vzdálenější objekty od kamery s méně detaily (tzv. level of detail). Takto je možné vykreslit více vzdálenějších objektů a při jejich vzdálenosti téměř neovlivnit jejich vzhled. U Bézierovy křivky může být dělení křivky závislé na vzdálenosti od kamery. V případě, že by křivka byla dále od kamery, vykreslila by se s méně přímkami. Tento algoritmus je obecně známý pod názvem De Casteljau, pojmenovaným po jeho autorovi.

Na křivky je kladeno několik požadavků [1][3]. Jedním z požadavků je například možnost snadné spočítání derivace. Ty se použijí například pro určování směru a rychlosti objektu pohybujícího se po křivce. Častým požadavkem je, aby nedocházelo k oscilaci křivky. Malá změna pozic bodů nemá vyvolat velkou změnu tvaru křivky. Dalším požadavkem je, aby se při změně pozice jednoho bodu nezměnil tvar celé dlouhé křivky, ale aby se provedly pouze lokální změny. To je dosaženo za pomocí skládání křivek z částí (segmentů). Mezi segmenty se navíc definují spojitosti a to geometrická a parametrická. Dalšími požadavky je často i to, aby byla křivka schopná reprezentovat různé varianty tvarů. Na to je velmi vhodná NURBS křivka.

Kapitola 2
Rešerše

Tato kapitola je rozdělena na dvě části. Sekce 2.1 rozebírá jednotlivé studijní materiály týkající se křivek. V druhé sekci 2.2 se porovnávají existující aplikace, které používají křivky, nebo jsou přímo určeny na výuku křivek.

2.1 Studijní materiály

V této kapitole jsou rozebrány jednotlivé studijní materiály, které se týkají křivek.

2.1.1 Přednášky předmětu PGR

Výuky křivek se v předmětu PGR týkají dvě přednášky, které jsou dostupné na stránkách předmětu a to jak v české, tak v aktuálnější anglické verzi [1][3]. V průběhu chodu předmětu se přednášky neustále vylepšovaly. Autory těchto přednášek jsou Ing. Petr Felkel, Ph.D. a Ing. Jaroslav Sloup.

První přednáška týkající se křivek začíná motivací. Zmiňuje se zde například použití křivek v animacích, vektorové grafice, při tvorbě modelů, nebo při vytváření fontů. Dále se vysvětlují typické požadavky na křivky, mezi které patří například invariance vůči transformacím, snadné počítaní derivací, nebo hladkost a spojitost křivek. Přednáška pokračuje typy jednotlivých křivek a způsoby reprezentace křivek (explicitní, implicitní a parametrická). Po vysvětlení problematiky reprezentací křivek se navazuje na dělení křivek na segmenty, kde se vzápětí podrobně zmiňují druhy spojitostí (parametrická a geometrická). Dále se probírají interpolační a aproximační křivky. Z interpolačních křivek je například velmi podrobně vysvětlena Fergussonova křivka a z aproximačních Beziérova křivka včetně názorných obrázků. V případě Béziera se také vysvětluje algoritmus De Casteljau, používaný při adaptivním vykreslování. Poslední aproximační křivkou, která se také v přednášce velmi diskutuje je Coonsova. Přednáška je zakončena použitím derivací křivek a vysvětlením definicí ploch pomocí křivek.

Druhá přednáška je zaměřena hlavně na aproximační křivky. Nejprve se vysvětluje význam a rozdíl mezi Spline křivkami a B-Spline. Dále se prezentuje maticová notace pro definování křivek a hodnoty prvků matic pro různé druhy křivek (Fergussonova, Bézierova, Coonsova). Přednáška

pokračuje vysvětlováním bázových funkcí pro B-Spline. Dále je vysvětlen uzlový vektor, pro pochopení rozdílu o proti uniformní a neuniformní kubické B-Spline křivce. Poté následuje vysvětlení vah bodů pro křivky a algoritmus Cox–De Boor, který je používaný při konstrukci bázových funkcí. Po této vysvětlené látce se již přechází na podrobné vysvětlení uniformní B-Spline křivky a vliv opakovaní hodnot v uzlovém vektoru. Zbytek přednášky se podrobně zabývá křivkami NURBS. V přednášce se vysvětluje i praktické použití NURBS při tvorbě povrchů a obalových ploch.

2.1.2 Lubovo místo

Lubovo místo je webová stránka[1], na niž je diplomová práce Lubomíra Alexandra, zabývající se křivkami v počítačové grafice. Výhodou této práce je vytvořená aplikace s různými typy křivek a jejich ukázkami, které se nachází u jednotlivých témat křivek. Tato aplikace je hodnocena v kapitole 2.2.4. Nevýhodou je aktuálně špatně nastavené kódování znaků webové stránky, které je nutno ve webovém prohlížeči nastavit na *windows-1250*.

Práce se skládá celkem z dvanácti kapitol. První kapitola definuje křivky, zabývá se dělením křivek na interpolační a aproximační a vysvětluje spojitost křivek (geometrickou a parametrickou). Dále se práce věnuje již konkrétním křivkám. Druhá kapitola obsahuje Lagrangeovou interpolační křivkou, která je doprovázena i appletem. Další kapitoly obsahují Fergusonovu křivku, Catmull-Rom, Bézierovy kubiky (včetně algoritmu De Casteljau), Coonsovu křivku, B-Spline křivky a NURBS. Obsah je tedy velmi podobný vyučovaným křivkám v předmětu PGR. Nevýhodou je přebírání části obsahu bez uvádění zdrojů. Použitá literatura je sice vypsána v samostatné kapitole, ale u přebíraného textu není zřejmé, ke kterému zdroji patří.

2.1.3 Přednášky ostatních škol

Přednášky, které se zde rozebírají, jsou ze školy MIT[4][5] a Stanford[6][7] a porovnávají se s přednáškami PGR z kapitoly 2.1.1. Přednášky, které se týkají grafických křivek jsou celkem dvě a to jak na škole MIT, Stanfordu tak i na ČVUT v předmětu PGR. Na každé škole tvoří tyto přednášky pouze část výuky. Zbylé přednášky se týkají dalších témat počítačové grafiky a jsou veřejně dostupné[2] [3]. Některá témata přednášek byla společná na všech školách. Mezi ně patří například maticová notace, spojitost křivek, algoritmus De Casteljau a Bézierova křivka.

Obsah přednášek je na škole MIT[4][5] velmi podobný přednáškám v předmětu PGR. První přednáška začíná motivací a dále pokračuje Bézierovou křivkou, algoritmem De Casteljau a maticovou notací. Druhá přednáška obsahuje Bernsteinovy polynomy, derivace křivky, spojitost křivek, B-Spline, převod mezi Bézierovou a B-Spline křivkou (a i naopak), NURBS a zbytek

[1] http://www.hyperkrychle.cz/curves/obsah.html

[2] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/

[3] https://web.stanford.edu/class/cs248/lectures.html

přednášky je věnován povrchům a tělesům modelovaných za pomocí křivek. Rozdíl o proti přednáškám z předmětu PGR je hlavně v množství a podrobnosti vyučovaných křivek. V přednáškách MIT se nenachází Hermitovy křivky a Catmull-Rom. B-Spline křivky i NURBS se vysvětlují podrobněji v předmětu PGR, ale povrchům a tělesům se více věnují přednášky MIT.

Přednášky ze Stanfordu[6][7] obsahují různé druhy interpolací používané v počítačové grafice (SLERP, interpolace nejbližších sousedů a lineární interpolaci), Hermitovy křivky, maticovou notaci, spojitosti křivek, Catmull-Rom, Bézierovu křivku a algoritmus De Casteljau. O proti přednáškám PGR zde není například B-Spline, NURBS a také plochy a tělesa definovaná za pomocí křivek. Výhodou je, že se přednášky podrobně věnují interpolacím.

2.1.4 Tutoriály na internetu

Křivkám je na internetu věnováno velké množství tutoriálů. Nejčastěji se jedná o Bézierovu křivku. Jedním z tutoriálů je video [8]. V tomto videu autor vysvětluje Bézierovu křivku, kterou zároveň i implementuje v jazyku JavaScript. Výhodou je názornost tutoriálu a dostupný zdrojový kód, ale nevýhodou je, že se nevěnuje křivce podrobněji, jako například knihy z kapitoly 2.1.5.

Dalším zdrojem, který se věnuje křivkám je web [9]. Tento web se zaměřuje hlavně na Bézierovu křivku. Velkou výhodou webu je velké množství podrobných tutoriálů, které jsou doprovázené interaktivními aplikacemi. Tyto aplikace jsou funkční přímo na webu a jsou umístěny u jednotlivých vysvětlovaných témat, ke kterým se vztahují.

Tutoriál, který se nachází na webu [10] se zaměřuje na křivku NURBS. Pro křivku vysvětluje potřebný teoretický základ. Vysvětlování je doprovázeno zdrojovým kódem v jazyku C#, v kterém autor naprogramoval i aplikaci celého tutoriálu a umístil ji na web ke stažení.

Dalším webem, který se týká křivek je [11]. Zde se používají interaktivní Java applety. Applety, které se zde nachází se zaměřují například na algoritmus De Casteljau, Bézierovu křivku, B-Spline a povrchy. Nevýhodou webu je nutnost nainstalované Javy pro spouštění appletů.

2.1.5 Knihy s grafickými křivkami

Grafické křivky jsou často vysvětlovány v knihách týkající se moderní počítačové grafiky. Jednou z takových knih je například [2]. V této knize jsou vysvětleny základy křivek, mezi které patří definice křivek, maticová notace, vlastnosti a spojitosti křivek a dále následují již konkrétní křivky. První křivky jsou Hermitovské, dále Bézierovy, Coonsovy, B-Spline a NURBS. V knize se také velmi podrobně vysvětlují plochy a tělesa definované za pomocí křivek.

Podobnou knihou je i [12]. Tato kniha se však zaměřuje výhradně na křivky a plochy. Velkou výhodou je, že pro znázornění křivek a ploch používá program Rhino. V něm je například ukázána Fergusonova kubika, Bézierova křivka, Coonsova kubika, Coonsova bilineární plocha, Bézierova plocha, nebo

Uniformní ukotvená plocha. Další velkou výhodou této knihy je, že se zabývá i vlastnostmi a vztahy mezi jednotlivými křivkami a plochami.

Další knihou, která se týká moderní počítačové grafiky a obsahuje grafické křivky je [13]. V této knize se nachází několik typů křivek, mezi které patří Bézierovy, Hermitovské, B-Spline, NURBS, Coonsovy a Catmull-Rom. V knize se vysvětlují také plochy definované za pomocí křivek. Tato kniha se však v porovnání s knihou [2] věnuje křivkám méně detailně.

Další knihou, která se týká křivek je [14]. Tato kniha se zaměřuje výhradně na křivky. Křivky, které jsou v knihách [2] a [13], jsou vysvětleny i zde a však podrobněji. Například pro Bézierovu křivku, která je v grafice velmi častá, je zde samostatná kapitola. Kniha obsahuje i více metod pro dělené povrchy než v knize [2].

Další knihou, která se věnuje pouze křivkám je [15]. Tato kniha se zaměřuje hlavně na křivku NURBS, kterou vysvětluje velice podrobně. V úvodu se zde ale vysvětluje i Bézierova křivka. Velkou výhodou této knihy je mimo jejího podrobného výkladu také velké množství pseudokódů, které usnadňují pochopení i případnou implementaci křivek.

2.2 Existující aplikace

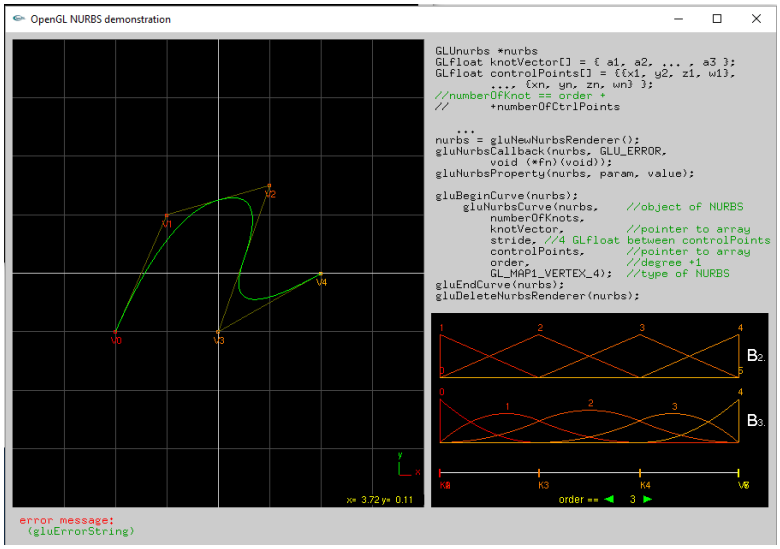
V této kapitole se porovnávají jednotlivé existující aplikace na práci s křivkami. Aplikace jsou hodnoceny podle kriterií, mezi které patří: *dostupnost*, *názornost*, *interaktivita* a *správnost*.

U kritéria *dostupnosti* je bráno v potaz, co vše musí mít uživatel nainstalováno (knihovny, Java virtual machine atp.), aby mohl aplikaci spustit, nebo zda je možné získat zdrojové kódy pro vlastní kompilaci aplikace. V *názornosti* pro výuku je nejvíce zohledněn poskytnutý grafický výstup aplikace, pomocí kterého uživatel pozoruje problematiku křivek. Mezi to patří například barevné rozlišování jednotlivých napojovaných segmentů křivek, aby bylo zřejmé, kde část křivky začíná a končí. Dále také jestli se zobrazuje konvexní obálka křivky, nebo jestli je možné zobrazit polynomy křivky. V *interaktivitě* je kladen důraz na možnosti editace křivek, jako je například přidání a odebrání bodů, editace vah, nebo uzlového vektoru. Poslední kritériem je *správnost*, kde se hodnotí samotný běh aplikace a její funkčnost. Tedy zda se aplikace chová podle předpokládaných vzorců křivek, jestli je aplikace deterministická, bez chybových výpisů, nebo bez nuceného ukončení systémem (například SEGV).

2.2.1 Modelář

Modelář je jedna z aplikací, která je používaná při výuce PGR a je volně dostupná na stránkách předmětu[4] včetně zdrojového kódu. Aplikace pracuje pouze s křivkami NURBS. Jejím autorem je Václav Gassenbauer a byla vytvořena v roce 2005. Vzhledem k tomu, že se jedná o binární aplikaci,

[4] `https://cent.felk.cvut.cz/courses/PGR/lectures.html`



Obrázek 2.1: Aplikace modelář

je nutné mít pro její běh dynamické knihovny. Mezi ně patří glut32.dll, ale i příslušná DLL C++ redistributable, podle použitého kompilátoru při sestavování programu. Výhodou v *dostupnosti* aplikace je její zdrojový kód a tedy i možnost oprav chyb aplikace či její rozšíření. Nezkušený uživatel by však nemusel zvládnout kompilaci aplikace, nebo by nedůvěřoval cizímu kódu, případně samotné binární aplikaci a byl by tím odrazen od spuštění.

Z pohledu *názornosti* je možné zobrazit váhy v jednotlivých bodech a hodnoty uzlového vektoru. Aplikace zobrazuje polynomy křivky, ale bylo by vhodné, aby bylo názornější, který bod patří ke kterému polynomu. Dále nejsou rozlišeny jednotlivé segmenty křivky. V aplikaci také není zřejmá pozice jednotlivých bodů. Ukázka aplikace je na obrázku 2.1.

Aplikace umožňuje několik *interakcí*. Je možné změnit pozice bodů, přidat další body, odebírat body, změnit stupeň křivky a měnit uzlový vektor. Aplikace také umožňuje změnit váhu bodů kliknutím kolečka myši na bod a následným táhnutím myši. Tento způsob interakce ale není zřejmý. Výhodou aplikace jsou dostupná dema pro kružnice, která lze načíst z menu, které se vyvolá kliknutím pravým tlačítkem myši do prostoru kreslené křivky. Nevýhodou v rámci interaktivity je změna uzlového vektoru pomocí táhnutí myši, protože nelze zadat přesně požadované číselné hodnoty.

V případě *správnosti* aplikace je nejzávažnější nedeterminismus aplikace, kdy se při změně stupně a jeho následném vrácení zpět na původní hodnotu změní průběh křivky. Další chybou je nevydělení součtem vah a tím špatné určení pozice bodu v prostoru při průchodu grafickými programy (*shaders*). Tato chyba byla již opravena.

2.2.2 B-spline

Tato aplikace umožňuje vykreslovat a editovat neuniformní kubický B-spline a je dostupná na stránkách předmětu PGR. Vzhledem k tomu, že se jedná

Obrázek 2.2: Aplikace B-spline

o Java applet, tak je pro její běh nutné mít stažený appletviewer[5], který je součástí JDK[6]. Aplikace se spustí pomocí programu appletviewer s parametrem URL adresy, kde se nachází příslušný applet. V místě adresy je nutné mít i příslušný jar soubor, který obsahuje třídu Bspline, se spustitelným kódem pro applet. Nevýhodou v *dostupnosti* tedy je, že pokud uživatel nemá nainstalovanou Javu, nemůže aplikaci spustit.

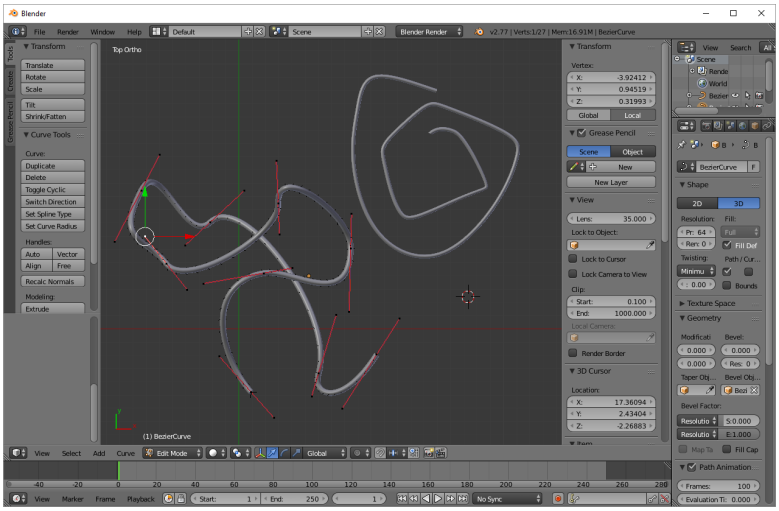
Z pohledu *názornosti* je dobré, že aplikace barevně rozlišuje jednotlivé segmenty křivky. Dále zobrazuje grafy polynomů křivky, které jsou také barevně odlišeny. Velkou výhodou je i velké množství ukázkových dem s různými křivkami, které se spustí při startu aplikace. Nevýhodou je, že není možné zobrazit pozice řídicích bodů a ani hodnoty uzlového vektoru.

Aplikace umožňuje z pohledu *interaktivity* měnit pozice řídicích bodů křivky kliknutím a následným táhnutím myši. Bod, který je vybrán pro přesun, je vždy nejbližší od místa kliknutí. Dále je také možné měnit uzlový vektor pomocí bodů zobrazených nad polynomy křivky, jak je znázorněno v ukázce na obrázku 2.2. Velkou nevýhodou je, že není možné mazat body a ani vytvářet nové body v již spuštěné aplikaci. V případě, že by bylo nutno přidat nebo odebrat body, musel by uživatel upravit soubor s křivkami, který se načítá při startu aplikace. Uživatel je tedy odkázán pouze na dostupná dema uložená v souboru. Vhodné by také bylo, aby mohl uživatel zadávat přímo v aplikaci přesné hodnoty uzlového vektoru a mohl tím například nastavit sousední hodnoty vektoru na naprosto stejnou hodnotu.

Z pohledu *správnosti* aplikace je hlavním problémem responzivita. Velikost okna aplikace je možné měnit jak do šířky tak do výšky, ale samotná vykreslovaná křivka nezmění svojí velikost. Mohlo by se tedy stát, že na příliš velkých obrazovkách by zobrazovaná křivka nebyla téměř vidět, nebo v opačném případě při malém rozlišení by se zobrazovala jen část křivky.

[5] `https://docs.oracle.com/javase/7/docs/technotes/tools/windows/appletviewer.html`

[6] Java Developer Kit



Obrázek 2.3: 3D modelovací program Blender

2.2.3 Blender

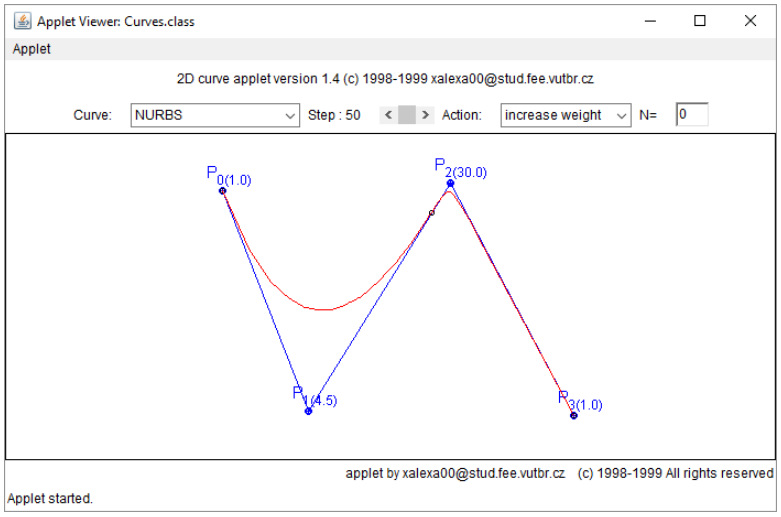
Blender je 3D modelovací program, který je dostupný zcela zdarma[7]. Program funguje na operačních systémech Windows, Linux i MacOS. Výhodou toho programu je i jeho *dostupnost* jako open source. Obsahuje mnoho funkcionalit, jako je například raytracing, vlastní engine, práce s animacemi a další. Pro tuto práci je bráno v potaz pouze používání křivek. Program umožňuje používat Bézierovy křivky, ale i NURBS. Křivky je zde možné používat pro vytváření modelů, ale i pro vytvoření dráhy, po které se může pohybovat například kamera. Ukázka programu se nachází na obrázku 2.3.

Z pohledu *názornosti* je největším problémem, že uživatel nevidí polynomy křivky. Dále nejsou barevně odlišeny jednotlivé segmenty a zdá se, jako by se při přidání bodů stále jednalo pouze o jednu křivku. Pro výuku by bylo také vhodné, aby se v případě křivek vykreslovala i konvexní obálka. Výhodou jsou například dostupné varianty kružnice a to jak pro Bézierovu křivku, tak pro NURBS, u kterých si může uživatel prohlédnout parametry vygenerovaných bodů (pozice a případně váhy).

V rámci *interakce* je velkou nevýhodou, že pro práci s křivkami je nutné, aby uživatel ovládal základní funkce programu. Mezi to patří například znalost módů (edit mode a object mode) a ovládání pohybu kamery v 3D prostoru za pomocí kláves a myši. V programu je možné křivkám měnit pozice řídících bodů, přidat i odebrat bod a v případě NURBS i změnit váhu bodu. Dále je možné křivce určit počet úseček, pomocí kterých se má vykreslit.

V rámci *správnosti* se nenašla během testování žádná chyba. Díky tomu, že je program dostupný jako open source, je možné případné chyby ve zdrojovém kódu opravit, nebo je nahlásit komunitě.

[7] `https://www.blender.org/download/`



Obrázek 2.4: Ukázka appletu Lubova místa

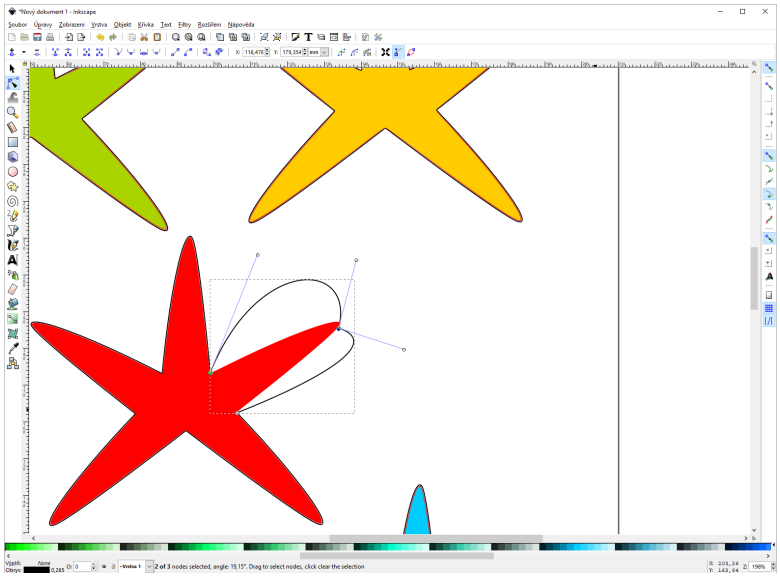
2.2.4 Applet Lubova místa

Lubovo místo je webová stránka s appletem, zaměřená na výuku křivek. Obsah této stránky je popsán v kapitole 2.1.2. Pro spouštění appletu je nutné mít nainstalovaný JDK (Java Developer Kit), v kterém se nachází program appletviewer, pomocí kterého se applet spouští. Nevýhodou v *dostupnosti* je tedy nutnost nainstalované Javy. Ukázka appletu je vidět na obrázku 2.4.

Z pohledu *názornosti* je největším problémem, že aplikace nezobrazuje polynomy křivky a uzlový vektor v případě NURBS. Výhodou je, že Lubovo místo nabízí applet s velkým množstvím různých typů křivek. Mezi ně patří například více druhů kružnic pomocí NURBS, Bézierovy racionální křivky, Catmull-Rom, Coonsovy křivky, Lagrangeovou interpolační křivku a Fergusonovu křivku. Dále je dobré, že je možné zjistit číselné souřadnice a váhy jednotlivých bodů.

V rámci *interakce* je dobré, že je možné měnit bodům pozice a váhu, nebo přidávat a odebírat křivkám body. Zda se budou měnit pozice bodů, nebo přidávat body, se ovlivní výběrem položky v menu. Operace se pak provádí kliknutím myši. Dále je možné měnit počet úseček, z nichž se má křivka vykreslit.

V rámci *správnosti* je největší problém s překreslováním v aplikaci, kdy se při každém kliknutí a táhnutí myši v appletu vykreslí pouze bílé pozadí a teprve až chvíli poté se vykreslí křivka. Díky tomu dochází například při přesouvání bodu k problikávání a ztrácí se tím informace, jak se křivka při přesouvání bodu postupně měnila. Dále také není vhodné, aby bylo vykreslování křivek omezeno maximálním počtem úseček, ale aby se počet úseček adaptivně měnil podle potřeby. Problém je také i u křivky NURBS, kdy se po přidání dalšího bodu do ukázkového dema přestane celá křivka vykreslovat.



Obrázek 2.5: Ukázka programu Inkscape

2.2.5 Inkscape

Inkscape je volně dostupný program, který je určen pro kreslení vektorové grafiky. Stáhnout lze z oficiálních stránek[8] a to pro Linux, Windows i MacOS. Ukázka programu je vidět na obrázku 2.5. Program umožňuje pracovat s celou řadou primitiv a vektorových objektů. Mezi ně patří například i Bézierovy křivky, na které se zaměřuje zbylá část hodnocení vhodnosti pro výuku.

Z pohledu *názornosti* je největším problémem pro pochopení a výuku křivek, že je uživatel odstíněn od principů křivek. Segmenty křivek nejsou barevně odlišeny a díky tomu se zdá, jako by se vždy jednalo pouze o jednu celistvou křivku. Barevné odlišení segmentů by však nebylo vhodné pro grafiky při kreslení za pomoci barevných křivek a je tedy zřejmé, proč něco takového není implementováno. Pro výuku křivek by bylo také vhodné, aby se zobrazovaly polynomy křivek a konvexní obálka křivek.

V rámci *interakce* je dobré, že uživatel může snadno přidávat křivkám body, měnit tečnu v bodě, posouvat již přidané body, nebo odstranit vybraný bod. Velmi užitečná je také možnost vytvoření bodu na křivce bez změny tvaru křivky dvojitým kliknutím na křivku. V takovém bodě lze opět změnit tečnu. Tímto způsobem se nejprve nadefinuje hrubý tvar křivky a teprve až poté jednotlivé detaily.

V rámci *správnosti* nebyla odhalena žádná chyba. V případě, že by byla, je důležité, že celý program je dostupný jako open source s velkou komunitou, která pomáhá chyby řešit a tím je zajišťována funkčnost i rozšiřování programu.

[8] `https://inkscape.org`

Kapitola 3

Uživatelský výzkum

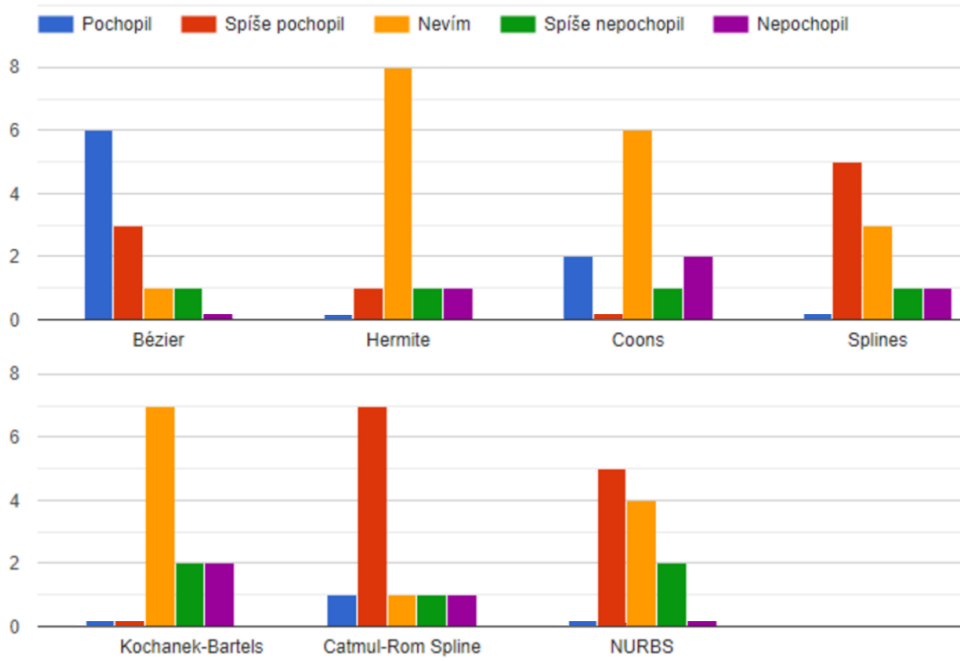
Uživatelský výzkum byl proveden za pomocí elektronického dotazníku, který byl vytvořen Google formulářem. Hlavním cílem bylo zjistit, zda studentům předmětu PGR dělají křivky problémy a také o které konkrétní křivky se jedná. Dalším cílem bylo zjistit, jaký typ výukové aplikace by si studenti spustili na svých počítačích.

Dotazník byl rozdělen do dvou hlavních sekcí. První sekce obsahovala otázky pouze pro studenty, kteří měli již zapsaný předmět PGR. První otázka byla na známku, kterou z předmětu získali. Další otázky se týkaly již křivek, a to zda studentům dělaly křivky problémy a následně vyplnění, jaké z křivek jim dělaly problémy. Mezi druhy křivek patřila Bézierova křivka, Coonsova, Hermitovské křivky, Splines, Kochanek-Bartels, Catmull-Rom a NURBS. Dále byla otázka, zda si student zkusil nějakou z aktuálně dostupných výukových aplikací na výuku křivek, jestli mu pomohly k pochopení křivek a o které aplikace se jednalo. Sekce končila otázkou, co by chtěl student změnit na výuce křivek.

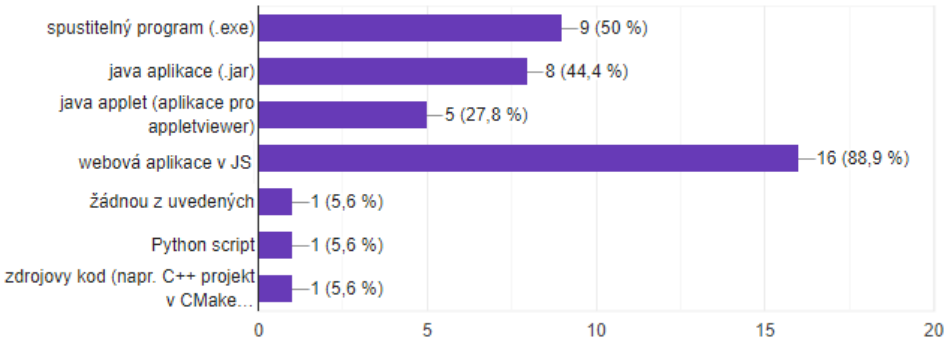
Druhá sekce cílila na všechny dotazované studenty a zjišťovala cílové platformy a typ aplikace, která by studentům nejvíce vyhovovala. První otázka byla, který typ aplikace by si student spustil na svém počítači. Mezi možnostmi byl spustitelný program (.exe), Java aplikace, Java applet, webová aplikace napsaná v jazyku JavaScript, žádný z uvedených a volitelné pole pro vyplnění vlastního návrhu na typ aplikace. Další otázky byly na zjištění, zda by si student spustil aplikaci na svém telefonu a jestli by tuto platformu upřednostnil před osobním počítačem.

Dotazníku se zúčastnilo celkem 18 studentů. 11 studentů (61%) mělo již zapsaný předmět PGR, 1 student (5.6%) si bude teprve předmět zapisovat a zbylých 6 studentů (33.3%) nemělo předmět zapsaný a nebudou si ho zapisovat. Ze studentů, kteří měli již předmět zapsaný, mělo známku A 9.1%, B 36.4%, C 27.3%, D 18.2%, E 9.1% a F 0%. Dále 50% studentů odpovědělo, že spíše měli problém s pochopením křivek a zbylých 50% odpovědělo, že spíše ne. Které druhy křivek studenti pochopili, je znázorněno na obrázku 3.1a. Na otázku, zda si studenti vyzkoušeli nějakou z dostupných výukových aplikací, odpovědělo 81.8% že ne a 18.2% odpovědělo že neví. Dále v druhé sekci odpovědělo ze všech účastníků dotazníku, že by 83.3% studentů upřednostnilo aplikaci pro osobní počítač před mobilní aplikací, 11.1% by ji neupřednostnili

a 5.6% neví. 38.9% by si spustilo mobilní aplikaci na svém zařízení, 27.8% ne a 33.3% neví. To, jaké typy aplikací by si byli studenti ochotni spustit na svém počítači je vidět na obrázku 3.1b. Všechny vyplněné otázky jsou dostupné v příloze B.



(a) : Odpovědi pochopení křivek.



(b) : Odpovědi přípustných typů aplikace.

Obrázek 3.1: Odpovědi studentů na otázky z dotazníku.

Kapitola 4

Požadavky na aplikaci

Požadavky na aplikaci byly vyhodnoceny na základě dat z elektronického dotazníku probíraném v kapitole 3 a rešerše z kapitoly 2. Jedním z požadavků je dostupnost. Z dotazníku vyplynulo, že by si 88,9% studentů spustilo webovou aplikaci psanou v JavaSriptu. Problémem doposud používaných výukových aplikací je ten, že u většiny z nich se jedná o Java applet, který by si spustilo pouze 27.8% studentů, nebo o spustitelný program (.exe), který by si spustilo pouze 50% studentů. Nejvhodnější by tedy byla webová aplikace psaná v JavaScriptu. Výhodou webové aplikace může být i to, že je možné ji spouštět ve webových prohlížečích na mobilních zařízení a uspokojit tím tak 11.1% studentů, kteří by raději upřednostnili mobilní aplikaci před osobním počítačem. Na to však musí být webová aplikace připravena a nepoužívat například pouze klávesy na provádění operací, protože by se nemusely na mobilním zařízení vůbec nacházet. Vzhledem k výsledku v dotazníku je však hlavním cílem, aby byla aplikace funkční pro osobní počítač.

Další z požadavků na aplikaci je interaktivnost. Uživatel by měl být schopen měnit pozice bodů, aby bylo zřejmé, jak se tím ovlivní průběh křivky. Dále také přidávat nové body, odebírat body a v případě NURBS měnit také uzlový vektor a měnit bodům váhu. V rámci názornosti by mělo být zřejmé, jaké jsou jednotlivé hodnoty uzlového vektoru, pozice a váhy bodů, jak vypadají polynomy křivky a kde začínají a končí jednotlivé segmenty křivky. Vhodné by také bylo, aby se aplikace snadno rozšiřovala o další křivky. V případě JavaScriptu ve webovém prohlížeči, kde je uživatelům dostupný zdrojový kód, by bylo možné tohoto cíle dosáhnout.

Kapitola 5
Křivky

5.1 Úvod do křivek

Následující kapitola vysvětluje základy křivek. Mezi ně patří například reprezentace křivek, maticová notace, spojitost křivek a další. Informace pro tuto část byly získány ze zdrojů [2][1][3].

5.1.1 Reprezentace křivek

Křivky jsou reprezentovány rovnicemi, které mohou být parametrické, explicitní, nebo implicitní. Každá z reprezentací je vhodná pro jiné situace, nejčastěji se však používá parametrické vyjádření.

$$y = f(x) \tag{5.1}$$

$$z = f(x, y) \tag{5.2}$$

V případě explicitního vyjádření 5.1 je x nezávislá proměnná, která se při vykreslování iterativně mění a na základě definované funkce $f()$ získáme hodnotu závislé proměnné y. Tyto dvě proměnné reprezentují souřadnice v 2D prostoru. Nevýhodou této reprezentace je to, že křivka je funkcí a proto není možné, aby souřadnici x odpovídaly dvě proměnné y. V případě explicitního vyjádření 5.2 je navíc použita další proměnná pro reprezentaci 3D prostoru. Výsledkem tohoto vyjádření již ale není křivka, ale plocha. V případě, že by jsme požadovali křivku, bylo by nutné použít průsečík rovin.

$$0 = F(x, y) \tag{5.3}$$

$$0 = x^2 + y^2 - 9 \tag{5.4}$$

Implicitní vyjádření 5.3 má nezávislé proměnné x i y. Používá se zejména při testování vzájemné polohy. Když by jsme použili například implicitní rovnici kružnice 5.4, tak pro bod by jsme byli schopni zjistit, zdali je uvnitř, nebo vně pouhým porovnáním výsledného znamínka. Pro $F(x, y) < 0$ je bod uvnitř v případě $F(x, y) > 0$ je bod vně a pro $F(x, y) = 0$ je bod na hranici.

Stejně jako v případě explicitního vyjádření tak i u implicitního vyjádření se při přidání další souřadnice pro 3D vytváří plocha. Požadujeme-li křivku, je nutné použít průsečík rovin. Vykreslení takové křivky by bylo možné například metodou sledování paprsku, kde by se testovala vzájemná poloha přímky a implicitní rovnice.

$$x = x(t), y = y(t), z = z(t) \tag{5.5}$$

$$x = x(t, u), y = y(t, u), z = z(t, u) \tag{5.6}$$

Křivky jsou nejčastěji vyjadřovány za pomocí parametrické rovnice 5.5. Na základě nezávislého parametru t, představující pohyb po křivce v čase, je možné získat bod na křivce. Vykreslování probíhá iterativně od počátku intervalu parametru t až do jeho konce (často od 0 do 1). Tuto reprezentaci křivky je možné použít v libovolné dimenzi. V případě, že by jsme chtěli vytvářet plochy, bylo by nutné použít další nezávislý parametr, tak jak je znázorněno v rovnici 5.6. Výhodou je také to, že na rozdíl od implicitního vyjádření nemusí být křivka funkcí.

5.1.2 Maticová notace

$$\begin{aligned} Q(t) &= \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \\ a_i &= \begin{bmatrix} a_{xi} \\ a_{yi} \\ a_{zi} \end{bmatrix} \\ Q(t) &= \sum_{i=0}^{n} a_i t^i \end{aligned} \tag{5.7}$$

Parametrické vyjádření používají i polynomiální křivky, které jsou definovány rovnicí $Q(t)$ v 5.7. Proměnná n se označuje jako stupeň křivky. V případě $n + 1$ se hovoří o řádu křivky. Problémem takového vyjádření je neintuitivní zadávání parametrů vektorů a_i. Z tohoto důvodu se používá maticová notace, rozdělená na tři části. Tato notace je znázorněna rovnicí 5.8 a 5.9 pro křivky stupně tři.

$$Q_i(t) = GMT \tag{5.8}$$

$$Q_i(t) = \begin{bmatrix} p_{i-3} & p_{i-2} & p_{i-1} & p_i \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{5.9}$$

První část notace je vektor geometrických podmínek $\mathbf{G}$, obsahující jednotlivé parametry měnící tvar křivky, mezi které nejčastěji patří řídící body, nebo tečné vektory. Druhá část je bázová matice $\mathbf{M}$. První řádek bázové matice

určuje průběh, jak moc se bude brát v potaz první geometrická podmínka v čase t. Stejně to platí i pro druhý řádek a druhou geometrickou podmínku. Třetí částí je vektor $\mathbf{T}$ závislý na čase. V případě, že vektor $\mathbf{T}$ zderivujeme $\begin{bmatrix}3t^2 & 2t & 1 & 0\end{bmatrix}^T$, získáme tak rovnici pro výpočet první derivace křivky.

$$Q_i(t) = \begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix} \begin{bmatrix}b_{11} & b_{12} & b_{13} & b_{14}\\ b_{21} & b_{22} & b_{23} & b_{24}\\ b_{31} & b_{32} & b_{33} & b_{34}\\ b_{41} & b_{42} & b_{43} & b_{44}\end{bmatrix}^T \begin{bmatrix}p_{i-3}\\ p_{i-2}\\ p_{i-1}\\ p_i\end{bmatrix} \tag{5.10}$$

Možností jak notaci zapsat je více. Jednou z variant je například otočení pořadí prvků vektoru $\mathbf{T}$ a otočení pořadí sloupců matice $\mathbf{M}$. V předmětu PGR se používá notace z rovnice 5.9. V knize Moderní počítačová grafika [2] se ale používá notace z rovnice 5.10. Po roznásobení však vede každá z variant na stejný výsledek.

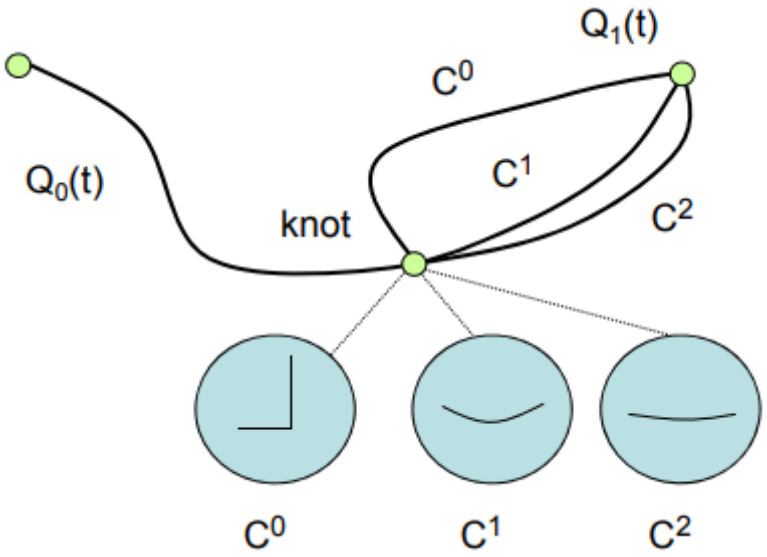
5.1.3 Spojitost křivek

Na křivky je často kladeno několik požadavků, aby se v praxi daly snadno používat. Jedním z požadavků je, aby se při změně řídícího bodu změnila jen část křivky a ne zcela celá křivka. Toho je dosaženo za pomocí napojování více křivek, o kterých se hovoří jako o segmentech, které definují jednu celou křivku.

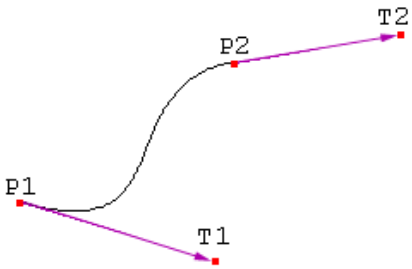
U křivek rozlišujeme dva druhy spojitosti a to geometrickou $\mathbf{G}$ a parametrickou $\mathbf{C}$. Křivka je C^n spojitá, jestli že má všechny derivace podle t v rozsahu 0 až n spojité. V případě G^n stačí, aby všechny derivace podle t byly v rozsahu 0 až n lineárně závislé. Z tohoto vztahu tedy platí, že je-li křivka C^n spojitá, pak je i G^n spojitá. Opačně to platit ale nemusí. Význam těchto spojitostí je ten, že při G^0 a C^0 navazují segmenty na sebe, tedy koncový bod prvního segmentu a počáteční bod dalšího segmentu má stejnou pozici, ale směr se v tomto bodě může skokem změnit. V případě G^1 je díky derivacím zaručeno, že se směr nezmění skokem, ale rychlost se může skokem změnit. V případě C^1 se skokem nezmění ani rychlost. Obdobně to platí i pro druhý řád spojitosti reprezentující akceleraci, ale i další řády. Ukázka $\mathbf{C}$ spojitostí je na obrázku 5.1.

5.2 Hermitovské kubiky

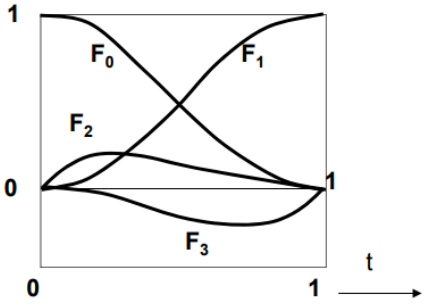
Hermitovské kubiky se používají pro hladkou interpolaci mezi body. Jak již z názvu kubik vyplývá, jedná se o křivky třetího stupně (čtvrtého řádu). Z toho důvodu se tyto polynomiální křivky často popisují maticovou notací, která je vysvětlena v kapitole 5.1.2. Geometrické podmínky se skládají z dvou bodů a dvou tečných vektorů v těchto bodech.



Obrázek 5.1: Porovnání C^0, C^1 a C^2 spojitostí. Zdroj [1].



(a) : Hermitova kubika. Zdroj [16].



(b) : Bázové funkce. Zdroj [1].

Obrázek 5.2: Hermitova kubika a její polynomy.

$$Q_i(t) = \begin{bmatrix} p_i & p'_i & p_{i+1} & p'_{i+1} \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ 1 & -2 & 1 & 0 \\ -2 & 3 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{5.11}$$

Jedna z možných maticových notací této kubiky je znázorněna v rovnici 5.11. Další častou variantou jsou prohozené geometrické podmínky p'_i a p_{i+1} a tím i prohozený druhý a třetí řádek matice $\mathbf{M}$. Výsledek je však naprosto stejný. Ukázka této křivky se nachází na obrázku 5.2a a její bázové funkce jsou na obrázku 5.2b. Bázové funkce *F2* a *F3* se použijí pro tečné vektory a funkce *F0* s *F1* pro řídící body.

První tečný vektor znázorňuje, jakým směrem a rychlostí se vstupuje do prvního počátečního bodu. Druhý tečný vektor znázorňuje, jakým směrem a rychlostí se vystupuje z druhého koncového bodu. Díky této vlastnosti se velmi jednoduše navazují další tyto křivky (segmenty) na sebe. Lze to provést tak, že tečný vektor a bod konce předchozí křivky se použije jako

počátek další křivky. V případě takového skládání je nutné zajistit, aby se při vykreslování vybraly v daném čase správné řídící body. Možné je například to, že celá křivka bude definována v čase t od *0* do *počtu segmentů*. První segment bude v čase t od 0 do 1, druhý od 1 do 2 a tak dále. Každý segment je pak řízen geometrickými podmínkami $p_{\lfloor t \rfloor}$, $p'_{\lfloor t \rfloor}$, $p_{\lfloor t \rfloor+1}$, $p'_{\lfloor t \rfloor+1}$. Důležité je ještě posunout čas t, protože bázové funkce této křivky jsou od 0 do 1. To se provede například takto: $t - \lfloor t \rfloor$. Další možnou variantou je ponechání rozsahu křivky od 0 do 1 a učení geometrických podmínek a nového času pro konkrétní segment na základě toho, v jaké části se t nachází. Pro dva segmenty a čas $t = 0.75$ by se křivka nacházela v polovině druhého segmentu.

$$p'_i = \frac{p_{i+1} - p_{i-1}}{2} \tag{5.12}$$

Mezi podmnožinu Hermitovských kubik spadá křivka Catmull-Rom. Hlavní změnou je ta, že uživatel již nenastavuje tečné vektory mezi napojovanými segmenty, ale tyto tečné vektory se vypočítají automaticky. Provede se tak za pomocí poloviny vektoru získaného z předchozího a následujícího bodu právě tom bodě, kde tečnu hledáme. Výpočet tečny je v rovnici 5.12.

$$p'_i = (1 - t)\frac{p_{i+1} - p_{i-1}}{2} \tag{5.13}$$

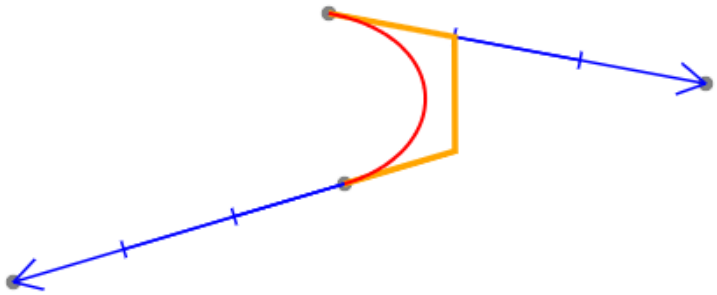
Další podobnou kubikou je Cardinal spline. Výpočet tangenty je znázorněn rovnicí 5.13 a provádí se téměř stejně jako v případě Catmull-Rom, ale navíc je možné tuto tangentu přeškálovat za pomocí parametru t (tension). V případě že je $t = 0$, bude opět výsledkem Catmull-Rom.

$$p'_i = \frac{(1-t)(1-c)(1+b)}{2}(p_i - p_{i-1}) + \frac{(1-t)(1+c)(1-b)}{2}(p_{i+1} - p_i) \tag{5.14}$$

Mezi podmnožinu Hermitovských kubik spadá dále i křivka Kochanek-Bartels. Výpočet tangent je znázorněn rovnicí 5.14. Tato křivka tentokrát obsahuje hned tři parametry a to t (tension), c (continuity) a b (bias). Nastavením všech parametrů na 0 získáme křivku Catmull-Rom. Nastavením $b = 0$ a $c = 0$ získáváme Cardinal spline. Výhodou této křivky je, že parametry b a c můžeme ovládat váhu mezi vektory předchozího a následujícího bodu.

Výhoda křivek Catmull-Rom, Cardinal spline a Kochanek-Bartels je ta, že uživatel specifikuje pouze body, kterými křivka prochází a případné úpravy tvary křivky provede za pomocí parametrů, místo nastavování tangent.

Hermitovské kubiky je možné vzájemně převádět mezi Bézierovou kubikou. V případě, že chceme převést Hermitovskou kubiku na Bézierovu, tak Bézierova křivka bude začínat a končit v počátečním a koncovém bodě Hermitovské křivky a zbylé dva řídící body jsou určeny za pomocí posunutí o vynásobený příchozí tečný vektor jednou třetinou a stejně tak i pro odchozí tečný vektor, který se ale navíc musí otočit na druhou stranu. Tento převod je ukázán na obrázku 5.3.



Obrázek 5.3: Převod mezi Hermitovskou a Bézierovou kubikou.

5.3 Bézierova křivka

Bézierova křivka je jedna z nejznámějších křivek. Byla pojmenována po Pierre Bézierovi, který ji zveřejnil v roce 1962 a používal ji ve firmě Renault. Křivku však používal již v roce 1959 Paul de Casteljau ve firmě Citroen. Ten ale křivku nepublikoval.

$$Q(t) = \sum_{i=0}^{n} p_i B_{i,n}(t) \tag{5.15}$$

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^i \tag{5.16}$$

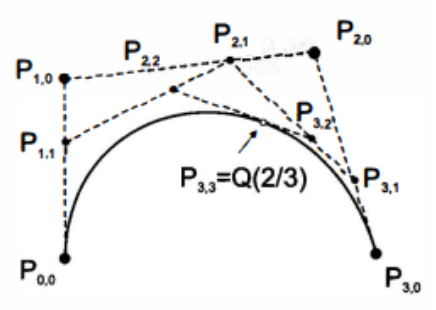
$$B'_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t)) \tag{5.17}$$

$$Q'(t) = \sum_{i=0}^{n} p_i B'_{i,n}(t) \tag{5.18}$$

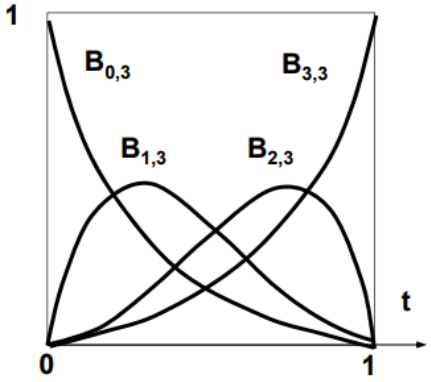
$$B''_{i,n}(t) = n(B'_{i-1,n-1}(t) - B'_{i,n-1}(t)) \tag{5.19}$$

Křivku je možné definovat pomocí vzorce 5.15, kde proměnná n značí stupeň křivky. Základem Bézierovy křivky jsou Bernsteinovy polynomy (viz vzorec 5.16), které publikoval Sergei Natanovich Bernstein v roce 1912. Pro křivku stupně tři, se nachází tyto polynomy na obrázku 5.4b. Protože součet Bernsteinových polynomů je vždy jedna a to na celém intervalu křivky t od 0 do 1, je díky tomu zaručeno, že se křivka bude nacházet v konvexní obálce řídicích bodů. Derivace polynomů lze vypočítat podle vzorce 5.17 a díky tomu lze získat i tečný vektor křivky, jak je znázorněno ve vzorci 5.18. Podobně jako lze získat první derivaci polynomu ze znalosti vyššího a nižšího stupně, lze získat stejně i druhou derivaci, známe li první derivace vyššího a nižšího stupně Bernsteinova polynomu (viz rovnice 5.19).

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t) \tag{5.20}$$



(a) : Algoritmus De Casteljau. Zdroj [2].



(b) : Bernsteinovy polynomy. Zdroj [1].

Obrázek 5.4: Bézierova kubika.

$$P_{i,n}(t) = (1-t)P_{i-1,n-1}(t) + tP_{i,n-1}(t) \tag{5.21}$$

Bernsteinovy polynomy je také možné definovat rekurzivně podle rovnice 5.20. Důkaz uvádí zdroj [17]. Toho využívá algoritmus De Castejau, který je schopen křivku dělit v čase t a tím i efektivně vykreslovat. Algoritmus je znázorněn na obrázku 5.4a a vzorci 5.21. Chceme li křivku rozdělit na dva Béziery, tak první Bézier bude mít řídící body $P_{0,0}$, $P_{1,1}$, $P_{2,2}$ ... $P_{n,n}$ a druhý Béziér $P_{n,n}$, $P_{n,n-1}$, $P_{n,n-2}$ ... $P_{n,0}$, kde proměnná n značí stupeň křivky. Takové dělení lze provést v určitém čase t. Křivku můžeme rekurzivně dělit až do určité splněné podmínky. V případě, že křivkou by byla již téměř přímka, můžeme tuto přímku vykreslit a můžeme ukončit dělení. Takto lze křivku efektivně vykreslit.

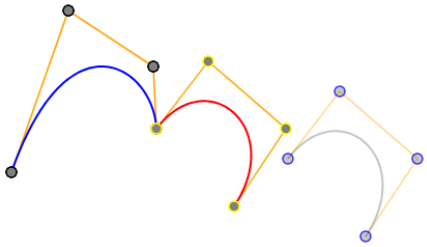
$$Q(t) = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{5.22}$$

$$Q_0 = P_n \tag{5.23}$$

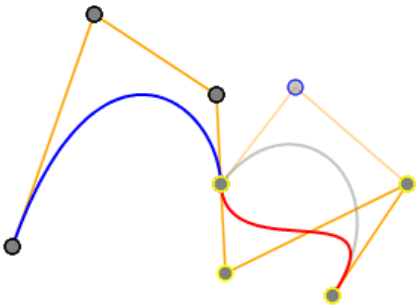
$$Q_1 = P_n + (P_n - P_{n-1}) \tag{5.24}$$

$$Q_2 = P_{n-2} + 4(P_n - P_{n-1}) \tag{5.25}$$

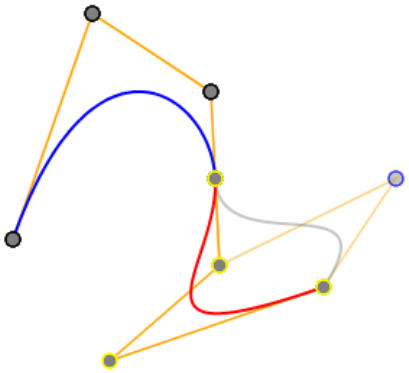
Křivku je možné používat s různými stupni. Nejčastěji se používá křivka stupně tři označovaná jako Bézierova kubika. Bézierova kubika má v případě maticové notace podobu vzorce 5.22. Její geometrické podmínky tvoří řídící body. Problémem Bézierovy křivky je ten, že každý z bodů ovlivňuje průběh celé křivky. Tento problém se řeší skládáním několika Bézierových křivek (segmentů), které se na sebe napojují. V takovém případě se řeší i spojitosti



(a) : Bézierovy kubiky. Spojitost C^0



(b) : Bézierovy kubiky. Spojitost C^1



(c) : Bézierovy kubiky. Spojitost C^2

Obrázek 5.5: Spojitost Bézierových kubik.

napojovaných křivek, o které se píše již v kapitole 5.1.3. Navazování křivek se věnuje například zdroj [18], kde vysvětluje navazování na dvou křivkách, kde první má řídící body $P_0, P_1 ... P_n$ a druhá křivka má řídící body $Q_0, Q_1 ... Q_n$. Spojitost C^0 i G^0 zaručíme tak, že počáteční bod druhé křivky nastavíme na stejnou pozici jako koncový bod první křivky (vzorec 5.23 a obrázek 5.5a). Pro spojitost C^1 je navíc požadováno, aby obě křivky měly stejné první derivace v bodě napojení. Toho lze dosáhnout změnou pozice bodu $Q1$ podle vzorce 5.24 viz obrázek 5.5b. Obdobně to platí i pro G^1 spojitost, kde stačí aby derivace křivek byly lineárně závislé. V případě, že by jsme požadovali navíc i C^2 spojitost, měnila by se tentokrát pozice bodu $Q2$ podle vzorce 5.25, jak je znázorněno na obrázku 5.5c.

5.4 Coonsova kubika

$$Q(0) = \frac{P_0 + 4P_1 + P_2}{6} \tag{5.26}$$

$$Q(1) = \frac{P_1 + 4P_2 + P_3}{6} \tag{5.27}$$

Coonsova kubika je polynomiální aproximační křivka stupně tři. Ukázka této křivky je na obrázku 5.6a. Důležitou vlastností je, že křivka začíná a

končí v takzvaném antitěžišti. Počátek je v jedné třetině těžnice vedené z bodu P_1 do úsečky P_0, P_2 (viz vzorec 5.26). Konec je v jedné třetině těžnice vedené z bodu P_2 do úsečky P_1, P_3 (viz vzorec 5.27). V případě, že opakujeme body, tedy platí že $P_0 = P_1 = P_2$, tak křivka začíná v bodě P_0 a končí v jedné šestině úsečky P_0P_3 [2].

$$Q_i(t) = \begin{bmatrix} p_i & p_{i+1} & p_{i+2} & p_{i+3} \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{5.28}$$

$$\begin{aligned} Q_i(0) &= \frac{P_i + 4P_{i+1} + P_{i+2}}{6} \\ Q_i(1) &= \frac{P_{i+1} + 4P_{i+2} + P_{i+3}}{6} \end{aligned} \tag{5.29}$$

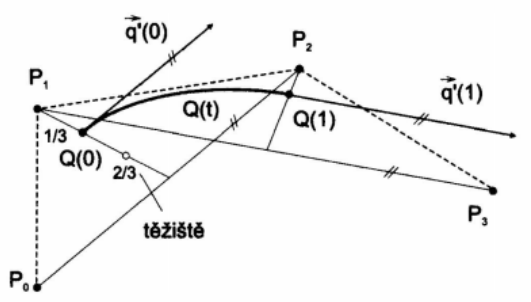
Křivku je možné zapisovat maticovou notací, jak je znázorněno rovnicí 5.28. Geometrickou podmínku tvoří řídicí body. Proměnná i použitá ve vzorci značí číslo segmentu. Napojování segmentů je v případě Coonsovy křivky velmi snadné. Máme li segment definovaný body P_0, P_1, P_2, P_3 a chceme li přidat další segment, stačí zadat další bod P_4. Nový segment je pak určen body P_1, P_2, P_3, P_4. Začátek a konec segmentu je pro obecný segment i určen rovnicí 5.29.

Plynulé navazování segmentů pouze posunutím bodů a přidáním nového bodu ve vektoru geometrických podmínek je dosaženo díky bázovým polynomům. Ty se nachází na na obrázku 5.6b. Při napojování pak dochází k tomu, že například bod P_0, který byl původně svázán s polynomem $BS_3(t)$ plynule přejde v dalším segmentu na $BS_2(t)$, v dalším segmentu na $BS_1(t)$ a v posledním segmentu na $BS_0(t)$. Před ani po těchto segmentech se bod P_0 výpočtů neúčastní. Při vykreslování je pak nutné určit, které body se v čase t výpočtu účastní. Definujeme li křivku v čase t na intervalu od 0 do $n-3$, kde n je počet řídicích bodů, pak se výpočtu účastní body $P_{\lfloor t \rfloor}$, $P_{\lfloor t \rfloor+1}$, $P_{\lfloor t \rfloor+2}$, $P_{\lfloor t \rfloor+3}$. V každém segmentu se pak čas t musí přepočítat, aby byl dodržen interval polynomů od 0 do 1. To lze provést například takto: $t - \lfloor t \rfloor$.

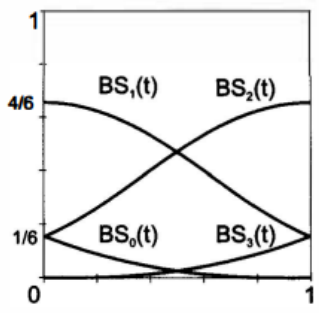
5.5 NURBS

NURBS nebo li Non-uniform rational B-spline je často používaná křivka z důvodu její variability. Za pomocí této křivky je možné definovat například Bézierovy kubiky, Coonsovy kubiky, ale lze pomocí ní vytvořit i kružnici a obdélník. Podrobněji se této křivce věnují zdroje [15][19][2].

$$\begin{aligned} Q(u) &= \sum_{i=0}^{n} p_i R_{i,n}(u) \\ R_{i,p}(u) &= \frac{w_i N_{i,p}(u)}{\sum_{j=0}^{n} w_j N_{j,p}(u)} \end{aligned} \tag{5.30}$$



(a) : Coonsova kubika. Zdroj [2].



(b) : Polynomy Coonsovy kubiky. Zdroj [2].

Obrázek 5.6: Coonsova kubika.

Křivka je definována rovnicí 5.30. Proměnná n v rovnici značí stupeň křivky, proměnná u představuje čas a $R_{i,p}(u)$ se označuje jako racionální bázová funkce. Křivka požaduje aby bylo zadáno $n+1$ bodů, tedy p_0, p_1, p_2 ... p_n. Každý tento bod mám mít definovanou váhu. Ta je ve vzorci označována jako w_i. Důležitou funkcí je také bázová funkce $N_{i,p}(u)$, kterou lze spočítat za pomocí algoritmu Cox De Boor.

$$N_{i,0}(u) = \begin{cases} 1, & t_i \leq u < t_{i+1} \\ 0, & jinak \end{cases}$$
$$N_{i,p}(u) = \frac{u - t_i}{t_{i+p} - t_i} N_{i,p-1}(u) + \frac{t_{i+p+1} - u}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(u) \tag{5.31}$$

Algoritmus Cox De Boor[20] je vyjádřen rovnicí 5.31. Proměnné t_i jsou hodnoty takzvaného uzlového vektoru. Jestliže indexujeme uzlový vektor od nuly, pak křivka začíná v čase t_n a končí v čase t_c. Je požadováno, aby uzlový vektor byl velikosti $n + 1 + c$, kde c je počet bodů. Hodnoty vektoru jsou neklesající $t_0 \leq t_1 \leq t_2 \leq ... \leq t_{n+c}$. V případě, že by nastala nedefinovaná situace, například to že by bylo děleno nulou, je výsledkem této operace 0. Důležitou vlastností je, že polynomy generované tímto algoritmem jsou na celém intervalu v rozsahu 0 až 1. U křivky NURBS pak hodnoty uzlového vektoru ovlivňují tvar (průběh) celé křivky.

$$Q'(u) = \sum_{i=0}^{n} p_i N'_{i,n}(u)$$
$$N'_{i,p}(u) = \frac{p}{t_{i+p} - t_i} N_{i,p-1}(u) - \frac{p}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(u) \tag{5.32}$$

V případě, že je nutné znát derivaci křivky NURBS, lze ji získat za pomocí vzorce 5.32. Důkaz správnosti je uveden ve zdroji [21], který i uvádí, že derivace je například nutné znát při fyzikálních výpočtech jako je například analýza zatížení budov.

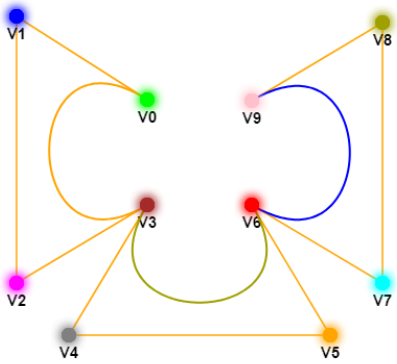
Jak již bylo řečeno, křivka NURBS znamená Non-uniform rational B-spline. Zobecňuje tedy křivku *B-spline* (Coonsův kubický B-spline). *Rational* se používá pro označení, že se jedná o křivku, která používá váhy bodů (body mají homogenní souřadnice). A *Non-uniform* značí, že hodnoty uzlového vektoru nemusí být od sebe stejně vzdáleny.

Jestliže chceme pomocí křivky NURBS definovat Coonsův kubický B-spline, tak stačí, aby body měly váhu 1, stupeň křivky byl nastaven na 3 a uzlový vektor měl od sebe stejně vzdálené hodnoty. Například {0,1,2,3,4,5,6,7,8,9} u křivky s šesti zadanými body. Zde je již vidět velká výhoda NURBS. Protože se racionální báze počítají vždy pro každý bod, nemusí se zde řešit, v jakém segmentu se při vykreslování nacházíme a které body se účastní výpočtu, tak jak se řešilo v případě Coonsovy kubiky v kapitole 5.4.

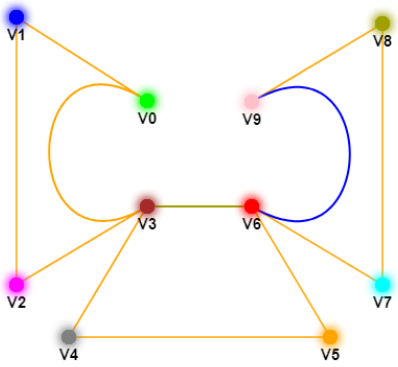
Podobně jako lze použít křivku NURBS na definování Coonsovy kubiky, lze ji také použít i na Bézierovy křivky. Taková křivka je stupně 3. Významný vliv má zde opakování hodnot uzlového vektoru. Jak již bylo řečeno v kapitole 5.3, křivka začíná a končí v prvním a posledním řídícím bodě. Toho lze u křivky NURBS dosáhnout opakováním hodnot uzlového vektoru. V případě křivky stupně 3 stačí v hodnotách uzlového vektoru $t_1, t_2, ..., t_{n+c-1}$ zopakovat třikrát stejnou hodnotu a dojde k tomu, že křivka bude procházet řídícím bodem. Například pro 10 bodů může být uzlový vektor {1,1,1,1,2,2,2,3,3,3,4,4,4,4}. Polynomy pro takový uzlový vektor jsou na obrázku 5.7c. Křivka je znázorněna na obrázku 5.7a. V případě, že by jsme u těchto kubik chtěli spojitost C^1, stačilo by již pozměnit korespondující body, jak je vysvětleno v kapitole 5.3. V případě, že by jsme bodům $V4$ a $V5$ nastavili nulovou váhu, vznikla by mezi body $V3$ a $V6$ úsečka, jak je znázorněno na obrázku 5.7b. Cesta po takové úsečce by však nebyla lineární. Pokud by jsme chtěli pomocí NURBS vytvořit obdélník, bylo by to možné provést právě pomocí nastavování nulových vah a nebo pomocí nastavení stupně křivky na 1.

Další výhodou křivky NURBS je, že lze pomocí ní definovat kružnici. Jednu z možných variant uvádí zdroj [2]. Křivce nastavuje stupeň 2. Uzlový vektor na hodnoty {0,0,0,1,1,2,2,3,3,4,4,4}. Bodů používá celkem 9, které umisťuje postupně do tvaru čtverce a to nejen do rohů, ale i mezi rohy. Poslední bod je navázán na pozici prvního bodu. Rohové body mají váhu $\frac{\sqrt{2}}{2}$ a ostatní váhu 1.

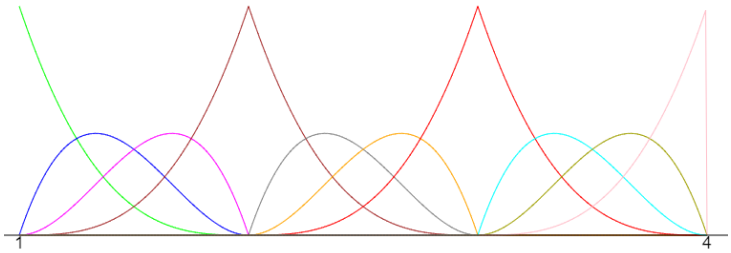
Nevýhodou křivky NURBS je, že je časově náročnější na výpočet, protože se pro každý bod počítá racionální báze. Dalším problém je, že může nastat přerušení křivky, které by mělo být při vykreslování ošetřeno. Stane se tak například když nastavíme řídícímu bodu, kterým křivka prochází, nulovou váhu. V takovém případě lze křivku považovat za přerušenou a skokem přechází na další segment.



(a) : NURBS definující Bézierovy kubiky.



(b) : NURBS a nulové váhy bodů $V4, V5$.



(c) : Bázové polynomy křivky NURBS.

Obrázek 5.7: NURBS definující Bézierovy kubiky.

Kapitola 6

Návrh aplikace

6.1 Požadavky

- Funkční požadavky
  - aplikace bude křivky vykreslovat
  - křivkám bude možné měnit parametry (například: stupeň křivky, uzlový vektor, tension atp.)
  - bude možné vizualizovat bázové funkce křivek
  - aplikace bude umět segmenty křivky barevně odlišit, aby bylo zřejmé, že se z nich křivka skládá
  - bude možné přidávat, mazat a přesouvat řídící body křivky
- Nefunkční požadavky
  - bude se jednat o webovou aplikaci
  - aplikace bude napsána v jazyku JavaScript
  - dostupnost - aplikaci bude možné provozovat i offline (lokálně) z webového prohlížeče, bez přítomnosti serveru
  - rozšiřitelnost - díky polymorfismu bude možné přidat další nové křivky, úlohy a tutoriály
  - udržitelnost - aplikace bude mít navrhnuto rozhraní (pro vykreslování), které bude voláno jednotlivými ukázkami úloh

6.2 Technologie aplikace

6.2.1 Web

Vzhledem k tomu, že se jedná o webovou aplikaci, bude nutné použít HTML (Hypertext Markup Language) a také CSS (Cascading Style Sheets) pro definování vzhledu webové stránky. Z HTML se použijí i standardní input elementy, pomocí nichž se budou zpracovávat vstupy od uživatele. Mezi takové použité elementy bude patřit jistě checkbox, number, radio a range.

V případě běžného vývoje webové stránky se často dělí web na takzvaný front-end a back-end. Za front-end se považuje klientská část (webový prohlížeč) a za back-end server, který zpracovává klientské požadavky. Často se na straně serveru používají další programovací jazyky, jako například PHP nebo Java, které na základě požadavků odesílají data klientovi. Výhodou je, že tímto způsobem je možné zpracovávat operace jako jsou například registrace, přihlášení, zpracování formulářů a další. Další výhodou je, že nad samotnými jazyky existují různé implementace, urychlující vývoj webu, označované jako framework. Mezi často používané patří například JavaServer Faces[1], Symfony[2], Nette[3], Zend Framework[4].

Cílem této aplikace je studentům nástroj na výuku křivek co nejvíce zpřístupnit. Z toho důvodu se nebude používat běžná back-end část, která by potřebovala, aby byl zpřístupněn server. Web bude možné používat i lokálně z webového prohlížeče bez přístupu na internet. Pro splnění požadavků totiž není nutná žádná z funkcionalit běžné back-end části. Běh aplikace tedy zajišťuje klientská část za pomocí jazyku JavaScript (JS). U takového přístupu je při vývoji webu nevýhodou, že se při více HTML stránkách opakují stejné části, jako je například navigace. V případě změny odkazu v navigaci, se pak musí pozměnit všechny webové stránky. Běžně tento problém řeší framework, který používá šablony, z kterých se poskládají jednotlivé HTML stránky. Zde tuto funkcionalitu bude zajišťovat JavaScript, který bude generovat navigaci na každé HTML stránce. V případě změny navigace ve scriptu se změna projeví všude.

Protože bude web dostupný i se svými scripty, bude cílem zachovat jejich čitelnost. K tomu bylo zvoleno použití JavaScript classes, které jsou zahrnuty v ECMAScript 2015, pomocí nichž se vytváří třídy. Tuto funkcionalitu umožňuje téměř každý webový prohlížeč. Díky třídám se kód zpřehlední, protože styl zápisu těchto tříd je velmi podobný jako v ostatních programovacích jazycích, na rozdíl od dřívějšího JS prototypového zápisu.

6.2.2 Renderer

Možností jak znázornit křivky je více. Jedna z nich by byla za použití pouze výpisu aktuálního bodu v zadaném čase. Takový přístup ale není pro běžného uživatele vhodný. Uživatel by pozoroval průběh za pomocí iterativní změny času. Když by pozměnil nějaký z parametrů křivky, jako je například stupeň křivky nebo pozice bodů, změnil by se i celý průběh křivky. V případě změny pozice řídících bodů by pak uživatel nezjistil například to, že křivka oscilovala. Proto si aplikace klade za cíl, aby se jednotlivé křivky vizualizovaly a uživatel získal přehled nad průběhem a chováním křivky.

Způsobů, jak lze na webové stránce vykreslovat je více. Jedna z variant je za pomocí pluginů, mezi které patří například Java nebo Flash Player.

[1] `https://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html`

[2] `https://symfony.com`

[3] `https://nette.org/en/`

[4] `https://framework.zend.com`

Tato varianta nebyla zvolena z důvodů dostupnosti, protože vyžaduje, aby měl uživatel tento plugin ve svém webovém prohlížeči nainstalován. Mezi další možnosti, které jsou ale tentokrát již v prohlížečích zabudované, patří vykreslování za pomocí SVG, Canvas 2D context a WebGL. SVG umožňuje vykreslovat vektorovou grafiku a podporuje dokonce i vykreslování Bézierovy křivky. Problém je však s výkonem, protože se při vykreslování mění DOM (document object model). Varianta Canvas 2D context tento problém nemá a zvládá vykreslovat velké množství objektů. Porovnání těchto dvou možností uvádí zdroj [22]. V případě WebGL je výhodou, že umožňuje vykreslovat 3D. Protože se ale jedná o webovou variantu OpenGL, je zde vyžadováno více volání než v případě Canvas 2D contextu. Je tím ale umožněno získat větší výkon, protože máme větší kontrolu nad tím co provádí grafická karta. Canvas 2D context ale umí vykreslovat i text, který se hodí například při popisu bodů štítky.

Canvas 2D context a WebGL je použit v několika enginech. Některé enginy kombinují vykreslování za pomocí obou metod. Mezi enginy používající tyto metody patří například Unity [5], PixiJS [6], Babylon.js [7], three.js [8]. Výhodou těchto enginu je v tom, že mají implementováno řadu funkcionalit, jako je například vykreslování modelů, osvětlení, animace, zvuky a další. Nevýhodou je, že určují strukturu, jak má být aplikace psána a často provádí další operace na pozadí. V případě 3D enginu je problémem také text, který se vykresluje na billboard. Pro jeho funkčnost je zapotřebí větší množství kódu. Z důvodu jednoduchosti, dobrého výkonu, přehlednosti a udržitelnosti bylo zvoleno naprogramování vlastního enginu s použitím **Canvas 2D context**. Třída aplikace bude potřebovat pro svůj běh pouze podědit od abstraktní třídy a vytvořit svojí instanci.

V případě, že by bylo požadováno vykreslování nejen křivek ale i ploch, bylo by možné implementovat vlastní 3D vykreslování. To by transformovalo tři body 3D trojuhelníku do 2D prostoru canvas elementu. Další možnost, jak aplikaci rozšířit o 3D by bylo za pomocí překrytí dvou canvas elementů, kde by se jeden použil na vykreslování 2D a druhý na 3D používající engine nebo samotné WebGL.

6.2.3 Knihovny třetích stran

Pro aplikaci bylo uvažováno použití několika různých knihoven, které mohou pomoci usnadnit, nebo zpřehlednit vývoj. Protože se jedná o aplikaci zabývající se křivkami, pracuje se zde s body, vektory a maticemi. Body a vektory mají nejčastěji dvě dimenze, protože postačují pro kreslení 2D křivky. V případě NURBS mají body ale i třetí souřadnici (váhu) a lze je tedy z programového hlediska považovat za 3D. Cílem ale je, aby byly křivky schopny pracovat v libovolných dimenzí, jestliže to jejich vlastnosti dovolují. V takovém případě se musí pracovat s body, u kterých předpokládáme li-

[5] `https://unity.com`
[6] `https://www.pixijs.com`
[7] `https://www.babylonjs.com`
[8] `https://threejs.org`

bovolnou dimenzi. Z důvodu 2D vykreslování je tato dimenze omezena ze zdola dvěma. Aby nedocházelo k neustálému opakování stejného kódu na práci s vektorovými a bodovými daty, je třeba zajistit zdrojový kód v podobě knihovny, která tyto operace s daty provádí. Takovou matematickou knihovnu by bylo možné vytvořit vlastní. Vzhledem k tomu, že ale existují již vytvořené přehledné knihovny, které sebou nesou řadu dalších potencionálně užitečných funkcionalit, tak byla vybrána právě jedna z nich. Mezi zkoumané knihovny patřila mathjs[9], glMatrix[10], Sylvester[11], AvoMatrix[12]. Z těchto knihoven byla pro práci vybrána knihovna **mathjs**. Její velkou výhodou je, že pracuje s daty libovolné dimenze. Dále je také rozšířena o velké množství užitečných funkcionalit. Jedna z nich je například to, že je schopna řešit lineární rovnice, ale umí například i QR rozklad. Nevýhodou by ale mohlo být, že vektory jsou zde vytvářeny stejně jako matice, což by mohlo uživatele zmást. Vhodná je ale také i vnitřní datová reprezentace matic za pomocí JS pole polí místo reprezentace objekty. K takovým datům lze jednoduše přistoupit a zpracovávat je i v kódu, u kterého požadujeme z důvodu obecnosti, aby nebyl na matematické knihovně závislý. Křivky a kód jednotlivých ukázek dem a tutoriálu budou smět používat tuto matematickou knihovnu. Právě ale z důvodu obecnosti se knihovna nebude používat v ostatních částech aplikace, jako je například samotný renderer diskutovaný v kapitole 6.2.2.

Požadavkem aplikace je, aby bylo umožněno měnit parametry křivky, jako je například stupeň křivky, čas aktuálně zkoumaného bodu na křivce, změna pozic bodů a podobně. Z tohoto důvodu se uvažovaly další typy knihoven, které jsou zaměřené na tvorbu GUI (graphical user interface), které slouží na generování HTML elementů a i zpracování dat. Mezi uvažované knihovny patřila jQuery UI[13], React[14] a Vue[15]. V případě této aplikace, jsou na některé vygenerované vstupní elementy kladeny požadavky. Například po uzlovém vektoru je požadováno, aby měl neklesající hodnoty prvků. Z tohoto důvodu bylo rozhodnuto, že tyto knihovny nebudou využity. Generování vstupních elementů bude zajištěno za pomocí nativních funkcí pracující s DOM. Výhodou je, že je možné mít plnou kontrolu nad chováním takto vygenerovaných elementů.

6.3 Doménový model

Doménový model je znázorněn na obrázku 6.1 a byl vytvořen za pomocí PlantUML[16]. Na obrázku je vidět rozvržení tříd s jejich nejdůležitějšími metodami a vztahy mezi nimi. Podrobný seznam tříd a jejich metod se nachází v dokumentaci na CD. Třídy reprezentující aplikaci jsou celkem

[9] `https://mathjs.org`
[10] `http://glmatrix.net`
[11] `http://sylvester.jcoglan.com/`
[12] `https://github.com/Avolord/AvoMatrix`
[13] `https://jqueryui.com`
[14] `https://reactjs.org`
[15] `https://vuejs.org`
[16] `http://plantuml.com`

čtyři a požadují, aby jim bylo při konstrukci předáno Id canvas elementu, kde budou moci vykreslovat. Jednotlivé instance aplikací můžou mezi sebou vzájemně komunikovat pomocí sdílených proměnných, nebo jedna aplikace může vytvářet a spravovat metodami druhou aplikaci. Na webové stránce se tedy bude moct nacházet i více aplikací a to jak navzájem nezávislých, tak i provázaných.

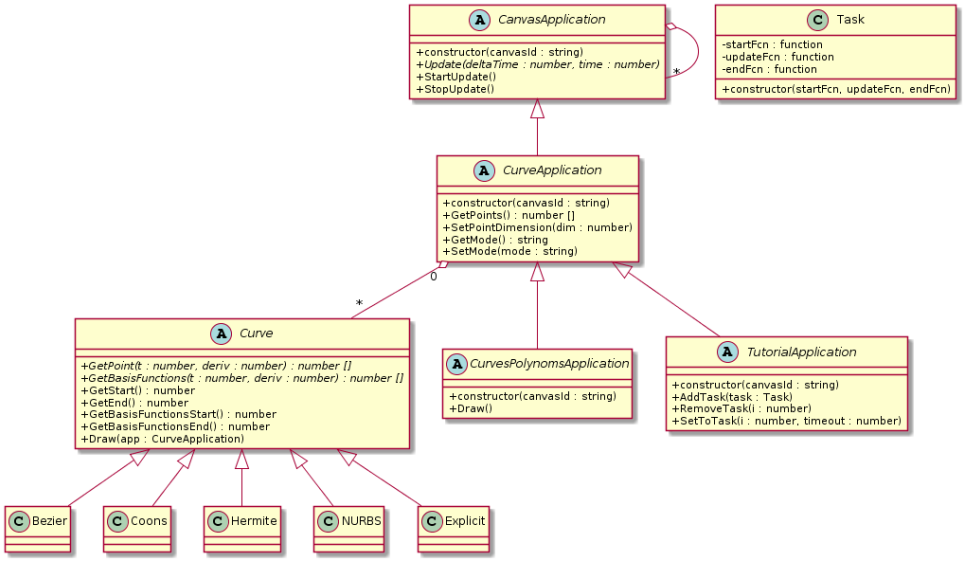
Nejzákladnější třídou reprezentující aplikaci je abstraktní třída **CanvasApplication**. Pro aplikaci takového typu nejprve potřebujeme vlastní třídu, která od této abstraktní dědí. Při konstrukci předá předkovi *canvasId* (Id canvas elementu) voláním metody *super(canvasId)* a následně již jen vytvoří vlastní *Update()* metodu, kde bude popsáno veškeré chování aplikace (vykreslování, zpracování dat, ovládání křivky atp.). Metodě *Update()* a jejímu spouštění se věnuje samostatná kapitola 6.4. Hlavním cílem této třídy je reprezentovat rozhraní mezi jednotlivými ukázkami aplikací a vykreslováním diskutovaném v kapitole 6.2.2.

Abstraktní třída **CurveApplication** dědí veškeré vlastností třídy CanvasApplication. Navíc implementuje funkcionality, které zajišťují práci s body, které se používají v jednotlivých křivkách. Pro práci s body byly zvoleny 3 módy: edit (měnící bodům pozici), delete (mazající body) a add (přidávající body). Těmto bodům umožňuje aplikace měnit i jejich dimenzi. Aplikace je také rozšířena o metody, které vytváří a editují vstupní elementy. Způsob jakým je vytváření řešeno, je navrhnut v kapitole 6.2.3.

Další abstraktní třída reprezentující aplikaci je **TutorialApplication**. Ta dědí vlastnosti od třídy CurveApplication. Třída je rozšířena o práci s úkoly (viz třída **Task**). Během její konstrukce je možné vytvořit jednotlivé úkoly, které ovládají běh aplikace. Každý úkol má možnost přecházet libovolně na jiný, podle toho jak je implementována. Úkol se skládá ze startovní funkce, která se spouští při jeho počátku. Z aktualizační funkce, která je volána *Update()* funkcí aplikace. A také z ukončovací funkce, která je volána na konci úkolu. Update funkce aplikace se zde však nesmí přepsat (přetížit). Aktivní úkol, který ovládá aplikaci, je vždy jen jeden.

Poslední abstraktní třídou aplikace je **CurvesPolynomsApplication**, která dědí od CurveApplication. Rozšíření této třídy je v možnosti vykreslování jednotlivých bázových polynomů křivek. Zde bude aktualizační smyčka *Update()* vypnuta, z důvodu ušetření výkonu. Ze zadané křivky, jejichž polynomy se mají pomocí této aplikace vykreslit se použije několik metod. Mezi tyto metody patří: *GetBasisFunctionsStart()*, *GetBasisFunctionsEnd()* a *GetBasisFunctions()*. Aplikace pomocí nich zjistí počátek a konec bázových funkcí a následně postupně iteruje v tomto časovém intervalu a kreslí získané hodnoty polynomů. V případě, že křivka umožňuje získat i derivace bázových funkcí, pak můžou být také pomocí této aplikace vykresleny.

V případě křivek, je nejdůležitější abstraktní třída **Curve**. Od této třídy všechny křivky dědí. Děděním se získá například možnost vykreslení křivky metodou *Draw()*. Ta se pak nemusí implementovat, protože je již implementována tímto předkem. Křivka však může tuto metodu předka přepsat. To se může provést například v případě, když známe efektivnější algoritmus. Každá



Obrázek 6.1: Doménový model

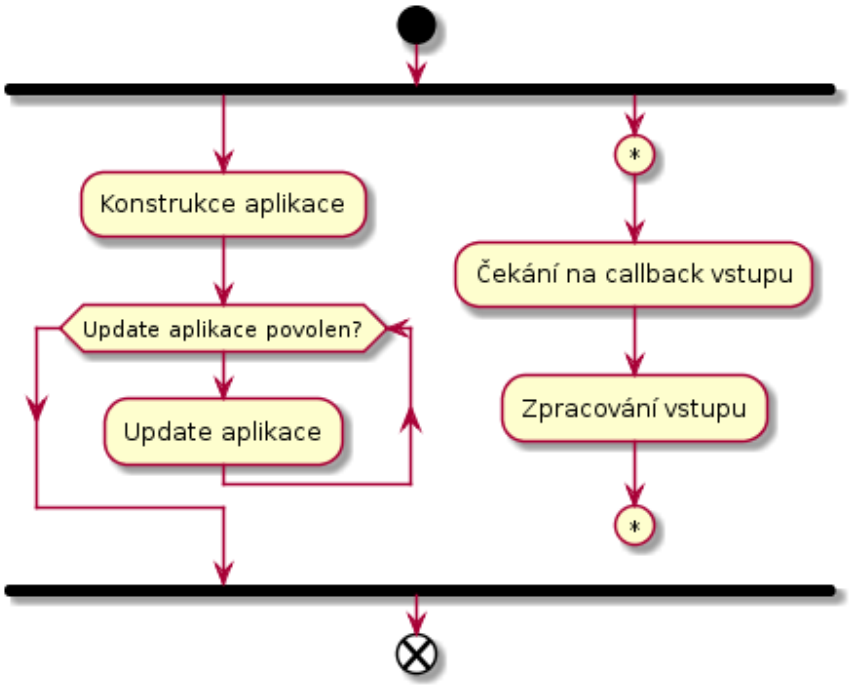
z křivek má povinnost vyplnit alespoň metodu *GetPoint()*, protože se používá pro získání bodu v čase, když se křivka kreslí. Doporučené je vyplnit i metodu *GetBasisFunctions()*, aby bylo možné kreslit polynomy křivky pomocí aplikace CurvesPolynomsApplication, jak je zmíněno v předchozím odstavci. Další metody jsou již dobrovolné. V základu mají poděděné metody na zjištění počátku a konce v případě bázové funkce a křivky počátek v 0 a konec v 1.

6.4 Aplikační smyčka

Na začátku se budou po načtení stránky spouštět jednotlivé scripty. První se provede konstrukce samotné aplikace. Ta zajistí například nastavení parametrů pro vykreslování, vygeneruje vstupní HTML elementy, vytvoří callback funkce pro zpracování vstupů a další. Po této konstrukci aplikace se začne spouštět aktualizační *Update()* smyčka aplikace. Tato smyčka se bude neustále opakovat několikrát (nejčastěji šedesátkrát) za sekundu. Volání bude řízeno webovým prohlížečem pomocí funkce *requestAnimationFrame()*. Smyčka slouží hlavně k vyčištění a následnému vykreslování do canvas elementu, ale aktualizují se v ní i parametry pro křivku na základě vstupních elementů.

Důležité je, že už při načítání stránky JavaScript zachytává a vykonává jednotlivé callback funkce. Aplikace pro svůj běh během konstrukce zaregistruje také své potřebné callback funkce. Jedna z nich je například na zachytávání událostí myši. Nejdůležitější callback funkcí je ale samotná funkce *Update()*.

Automatické volání funkce *Update()* bude možné z aplikace při jejím běhu i zakázat. Díky ale registrovaným callback funkcím, ji bude možné i tak opět spustit, když by bylo třeba. Křivky by bylo možné vykreslovat pouze při jejich inicializaci a při změně dat. Vzhledem k tomu, aby bylo možné provádět i animace (například pohyb jezdce po křivce), tak se tato možnost nezvolila.



Obrázek 6.2: Diagram aplikační smyčky

V případě ale instance objektu na vykreslování bázových funkcí křivek se tato smyčka zakáže. Vykreslení bází se provede vynucením voláním funkce *Draw()* a v případě, že bude třeba polynomy překreslit, tak se například z aktualizační smyčky hlavní aplikace toto volání znovu provede. Diagram aktualizační smyčky je znázorněn na obrázku 6.2 a byl vytvořen za pomocí PlantUML[16].

Kapitola 7
Implementace

7.1 Implementované křivky

Celkem bylo implementováno 5 různých křivek. Každá z křivek dědí od základní třídy *Curve*, tak jak je vysvětleno v návrhu v kapitole 6.1. Křivky a i jejich polynomy jsou vykreslovány iterativně. Počátek a konec iterativního vykreslování je určen funkcemi: *GetBasisFunctionsStart()*, *GetBasisFunctionsEnd()*, *GetStart()*, *GetEnd()*. Tyto základní zděděné metody jsou v intervalu od 0 do 1. Často jsou tyto funkce křivkami přepsány, protože začínají a končí v jiných časových úsecích. Tento úsek lze však přímo specifikovat i při vykreslování. Vykreslení křivky se provede za pomocí zděděné metody *Draw(app, from, to, numLines)*, kde *app* je instance třídy *CurveApplication*, pomocí které se provede vykreslování. Proměnné *from* a *to* reprezentují čas počátku a konce vykreslování křivky. V případě, že se tyto proměnné nenastaví (tedy bude jejich hodnota undefined), tak se použijí již zmíněné funkce pro získání počátku a konce. Proměnná *numLines*, která reprezentuje počet čar na vykreslení křivky, se také nemusí specifikovat a pak bude automaticky nastavena. Toto metoda, která se používá na vykreslování volá metodu *GetPoint(t, deriv=0)*, která vrací bod na křivce nebo její derivaci, podle proměnné *deriv* v zadaném čase *t*. Tuto metodu musí implementovat každá křivka. V případě, že chceme, aby bylo možné vykreslovat i bázové funkce křivky, je nutné aby se implementovala i metoda *GetBasisFunctions (t, deriv=0)*. V hlavní třídě *Curve*, od které křivky dědí se implementovala také funkce *GenerateTimeVector (t, deriv=0, degree=3)*, která je užitečná pro vytvoření vektoru času v t s derivací *deriv* a stupněm křivky *degree*. Tento vektor se používá hlavně při maticové notaci, jak je znázorněno v rovnici 5.9.

Jednou z implementovaných křivek byla křivka explicitní. Ta je popsána v rovnici 5.1. Tato křivka nepracuje s bázovými funkcemi. Její průběh je ovlivněn jejími body, kterými prochází. Křivka je užitečná hlavně pro znázornění problematiky oscilace. Jeden bod na konci křivky dokáže velmi prudce ovlivnit její druhý konec. Problémem je také to, že křivka musí být funkcí. Z toho tedy plyne, že žádný z dvou bodů nesmí mít stejné souřadnice X.

Další implementovanou křivkou je Hermitova kubika, která je popsána v kapitole 5.2. Této křivce bylo implementováno několik funkcionalit. Jedna z nich je například ta, že je křivce možné nastavit tečny tak, aby odpovídaly

křivce Catmull-Rom, Cardinal spline nebo křivce Kochanek–Bartels. Implementována byla také metoda, která vrací řídicí body, pomocí nichž je možné vytvořit Bézierovu křivku stejného tvaru. Problémem této křivky byl v její geometrické podmínce. Ta je složena celkem z dvou bodů a dvou vektorů. Aplikace však pracuje pouze s body a proto bylo potřeba při předávání řídicích bodů křivce na vstupu upravit data tak, aby odpovídaly požadavkům křivky. Tuto kubiku je možné definovat více než čtyřmi body. V takovém případě se bude skládat z více segmentů. Každé dva nové body definuje další nový segment.

Implementovaná byla i Bézierova křivka z kapitoly 5.3. Křivce je možné nastavit libovolný stupeň n. Tento stupeň určí i minimální počet bodů, který je $n + 1$. V případě, že se zadá více bodů, nebudou tyto body požity. Když by jsme chtěli křivku skládat z více segmentů, byla pro to vytvořena metoda, které se předá stupeň spojitosti a křivka, na kterou se bude aktuální napojovat. Metoda vrátí novou Bézierovu křivku dle zadaných parametrů. Další metoda, která byla implementována reprezentuje algoritmus De Casteljau, který je vysvětlen ve vzorci 5.21. Díky tomuto algoritmu byla implementována i metoda, která rozdělí současnou křivku na dvě, bez změny tvaru.

Další křivkou, která byla implementována je Coonsova kubika. Křivka je vysvětlena v kapitole 5.4. Protože je křivka kubikou, tak se jedná o křivku třetího stupně. Proto je požadováno, aby měla křivka alespoň 4 řídící body. Každý další přidaný bod definuje nový segment. Čtyři body tedy definují jeden segment, pět bodů dva, šest bodů tři a tak dále.

Poslední implementovaná křivka byla NURBS. Tato křivka je popsána v kapitole 5.5. Podobně jako u Bézierovy křivky je možné i u této křivce nastavit libovolný stupeň n. Křivka požaduje, aby jí bylo předáno alespoň $n + 1$ řídicích bodů. U každého bodu se poslední souřadnice chápe jako jeho váha. Ve 2D má tedy bod 3 souřadnice (x,y a váhu). Dále je křivce předán uzlový vektor k. Po vektoru je požadováno, aby měl neklesající prvky a jeho velikost byla $n + 1 + p$, kde p je počet řídicích bodů. Křivka používá pro vygenerování bázových polynomů algoritmus Cox De Boor. Časový interval křivky je určen z uzlového vektoru, který se indexuje od nuly. Křivka tedy začíná v čase $k[n]$ a končí v čase $k[p]$.

Každá z křivek byla implementována ve vlastním JavaScript souboru. Tyto soubory jsou: Explicit.js, Hermite.js, Bezier.js, Coons.js, NURBS.js. Podle konkrétní křivky je tedy pojmenován název souboru, ale také i třída implementující danou křivku. Všechny tyto třídy použily matematickou knihovnu *mathjs*, která je zmíněna 6.2.3. Její použití spočívalo hlavně v zpřehlednění zdrojového kód. Například pro násobení bodů hodnotami bázové funkce a poté jejich sečtení pro získání bodu v čase, bylo použito maticové násobení, místo iterace přes všechny body a jejich souřadnice. Knihovna také pomohla snadno implementovat explicitní křivku, protože umožňuje řešit systém lineárních rovnic. Hlavní třídu *Curve*, od které všechny křivky dědí, lze nalézt v souboru common.js.

7.2 Implementované tutoriály

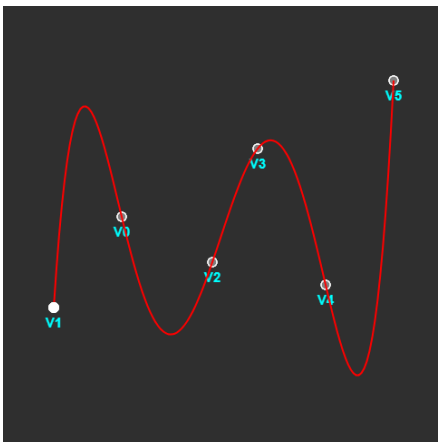
Celkem bylo implementováno 17 tutoriálů. Tyto tutoriály používají křivky, které jsou zmíněné v kapitole 7.1. Pro většinu křivek je vytvořeno více tutoriálů, protože znázorňují jejich různé vlastnosti. Explicitní křivka má však pouze jeden tutoriál, který je na obrázku 7.1a, znázorňující samotné chování křivky. Stejně je tomu i v případě Coonsovy křivky, jejíž tutoriál je na obrázku 7.1k. Samostatný tutoriál byl vytvořen také pro znázornění derivací křivek. Ten je na obrázku 7.1q. V případě tutoriálů byl kladen velký důraz hlavně na interaktivitu, tedy aby bylo možné měnit parametry křivky a řídící body. Díky tomu je zřejmé, jak jednotlivé parametry mění vlastnosti křivky. V jednotlivých tutoriálech, kde se zobrazují i bázové funkce křivek, jsou řídicí body barevně zvýrazněny. Jejich barva je stejná jako odpovídající barva bázové funkce pro řídící bod. Řídicí body, které souvisí s danými bázovými funkcemi se často mění v jednotlivých segmentech. To se děje například v případě Coonsovy křivky. Když bude aktuální bod v čase v prvním segmentu, tak budou barevně označeny první čtyři body. V druhém segmentu bude označen druhý, třetí, čtvrtý a pátý bod. V případě, že geometrickou podmínkou křivky není bod, ale vektor, tak je barevně označen vektor.

Pro Hermitovu křivku bylo vytvořeno 5 tutoriálů. První tutoriál znázorňuje chování křivky a je na obrázku 7.1b. Umožňuje přidávat libovolný počet bodů. Každé další dva nové body definují nový segment. Tečné vektory lze zde měnit bez omezení. Vykreslují se zde také i bázové funkce křivky. Geometrické podmínky, které jsou tvořeny řídícími body a tečnými vektory jsou barevně odlišeny podle barev odpovídajících bázových funkcí. Další tutoriál pro tuto křivku znázorňuje Catmull-Rom (viz obrázek 7.1c). Zde lze z tečných vektorů křivky měnit pouze první příchozí a poslední odchozí. Ostatní tečné vektory jsou nastaveny automaticky, tak jak je vysvětleno rovnicí 5.12. Další tutoriálem je Cardinal spline, který je na obrázku 7.1d. Opět se zde mění pouze příchozí a odchozí tečný vektor a zbylé jsou nastavovány automaticky podle vzorce 5.13. Podobně je tomu i v případě tutoriálu na Kochanek-Bartels z obrázku 7.1e. Jeho tečné vektory se nastavují podle rovnice 5.14. Poslední tutoriál je znázorněn na obrázku 7.1f. Jeho cílem je interaktivně znázornit převod Hermitovské kubiky na Bézierovu křivku. Tento převod je vysvětlen již v kapitole 5.2.

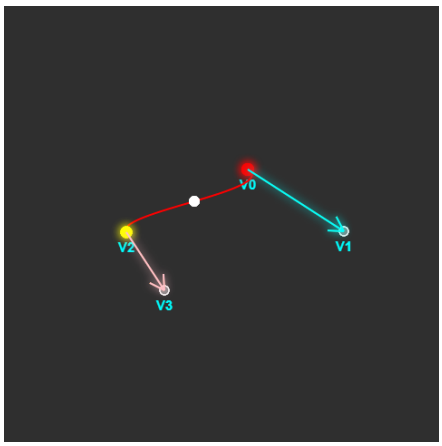
Další tutoriály se týkaly Bézierovy křivky. Celkem byli implementovány čtyři. První znázorňuje chování křivky jako takové. Je ho možné vidět na obrázku 7.1g. Body jsou barevně odlišeny podle bázových funkcí. Křivce lze nastavit libovolný stupeň za pomocí vstupního HTML elementu. Body lze libovolně přidávat a mazat, ale využije se vždy tolik bodů, jaký je řád křivky (stupeň křivky + 1). Další tutoriál, který je na obrázku 7.1h interaktivně znázorňuje algoritmus De Casteljau. Křivce lze stále měnit stupeň. Zde se však pro znázornění algoritmu využívá i vstupního slider elementu času, který v algoritmu ovlivňuje interpolace. Jednotlivé vrstvy rekurzivního algoritmu jsou znázorněny cestou úseček se stejnou barvou. Barvy bodů zde nejsou spojeny s barvami bázových funkcí. Díky tomuto algoritmu byl vytvořen i tutoriál

na dělení křivky na dvě Bézierovy křivky (bez změny tvaru). Tento tutoriál se nachází na obrázku 7.1i. Opět je možné měnit stupeň křivky. Tentokrát je zde ale i vstupní element, pomocí kterého se určí, kolikrát se mají křivky rozdělit. Bude-li zadána nula, tak se křivka nebude dělit. Zadá-li se jednička, křivka se jednou rozdělí. Pro dvojku se tyto dvě křivky znovu rozdělí a budou celkem čtyři. Místo dělení je určeno vstupním HTML slider elementem, který znázorňuje čas. Posledním tutoriálem spojeným s Bézierovou křivkou je možné vidět na obrázku 7.1j. Jeho cílem je interaktivně znázornit spojitost Bézierovy křivky. Zde se nenastavuje stupeň křivky. Ten je u obou napojovaných křivek 3. Jedné křivce je možné libovolně měnit body. Druhá křivka je určena na napojování. Ta je znázorněna hned dvakrát. Před napojením je vykreslena méně výrazně (průhledně). Po napojení je křivka již zobrazena sytou barvou. Díky tomu je možné pozorovat provedené změny při napojování. Stupeň parametrické spojitosti lze měnit za pomocí vstupního HTML elementu na C^0, C^1 i C^2 spojitost. V rámci názornosti jsou zobrazeny souřadnice a i derivace v bodě spojení pro hlavní i napojovanou křivku.

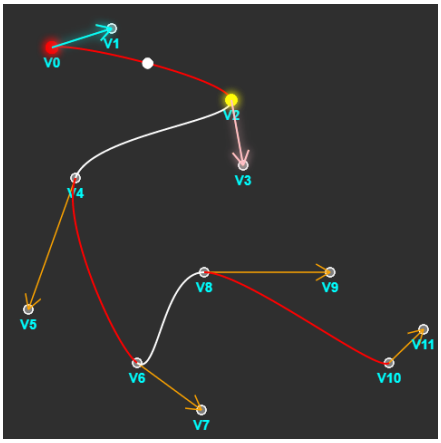
Tutoriálů, které mají souvislost s křivkou NURBS je celkem 5. První tutoriál, který je na obrázku 7.1l znázorňuje chování křivky. Ze vstupních HTML elementů je možné křivce měnit její stupeň a i uzlový vektor. Změny uzlového vektoru jsou odchyceny a používají se pro překreslení bázových funkcí křivky. Každý bod je barevně označen podle odpovídající bázové funkce. Křivce je možné přidávat libovolný počet bodů. V případě změny počtu bodů se automaticky změní i počet prvků uzlového vektoru. Další související tutoriál znázorňuje možnost definování Coonsovy kubiky za pomocí křivky NURBS. Tutoriál je na obrázku 7.1m. Zde se již uzlový vektor nemění, protože je nastavován aplikací automaticky. Uživatel pouze mění počty bodů, případně jejich pozice. Stejně je tomu i v tutoriálu na definování Bézierovy křivky pomocí NURBS, který je na obrázku 7.1n. Protože křivka NURBS umožňuje vytvoření i kružnice, byl pro to také vytvořen tutoriál, který je na obrázku 7.1o. Ze stejného důvodu byl vytvořen i čtverec viz obrázek 7.1p.



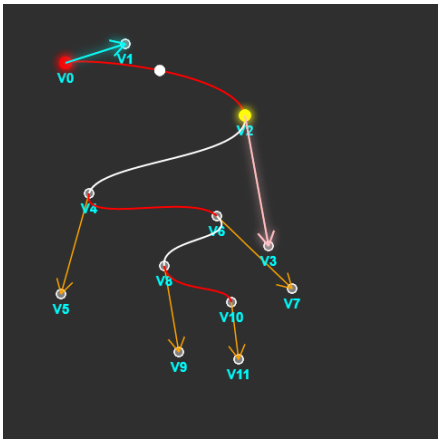
(a) : Explicitní křivka.



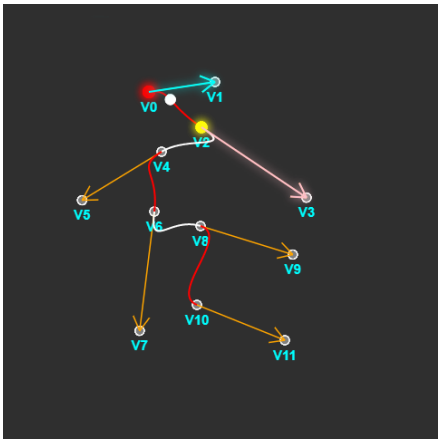
(b) : Hermitova kubika.



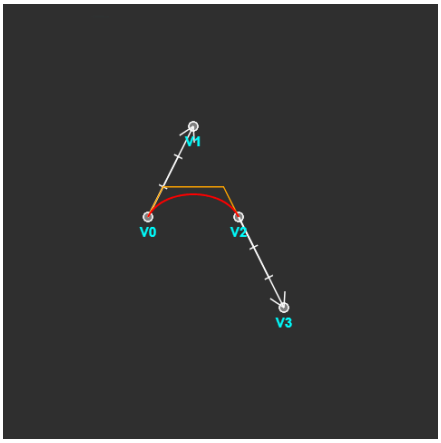
(c) : Křivka Catmull-Rom.



(d) : Křivka Cardinal spline.



(e) : Křivka Kochanek–Bartels.

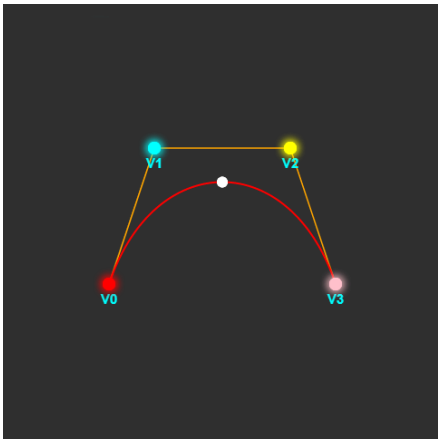


(f) : Převod Hermitovy křivky na Béziera.

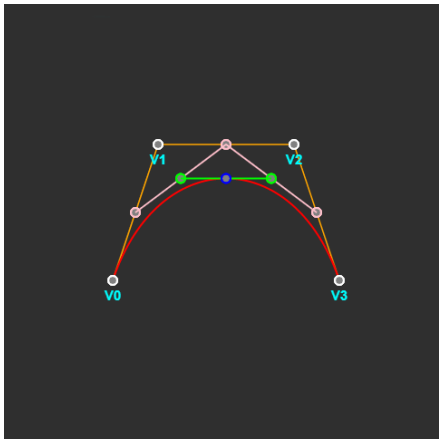
7.3 Úvodní tutoriál

Cílem úvodního tutoriálu je seznámit uživatele s aplikací, s jejím ovládáním a významem jednotlivých funkcí. Tutoriál se skládá celkem z 10 částí. V každé části musí uživatel provést v aplikaci požadované operace, aby byl automaticky přesunut na další část tutoriálu. Uživatel je o požadovaných operacích informován textovou podobou, která se nachází nad samotnou aplikací.

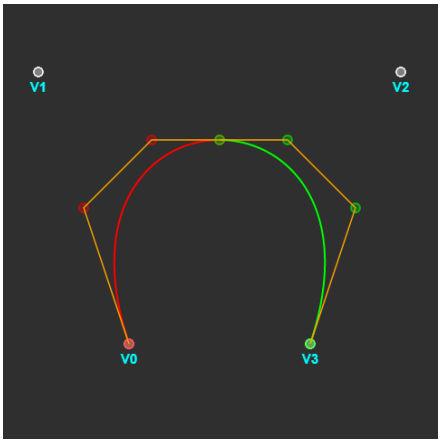
První část oznámí uživateli, že aplikace pracuje celkem se třemi módy a že má aktuálně nastavený mód (*add* mód), který přidává nové body. Cílem uživatele je, aby přidal alespoň 5 bodů. V další části je nastaven *edit* mód a uživatel má přesunout čtyři body do červeného kruhu pomocí kliknutí a táhnutí myši. V třetí části je nastaven *delete* mód, kde má uživatel smazat kliknutím na bod všechny zadané body. Čtvrtá část je zaměřena na alternativní přesouvání bodů za pomocí vstupních HTML elementů. Cílem je opět přesunout čtyři body do červeného kruhu. V páté části je uživatelův cíl, aby změnil mód



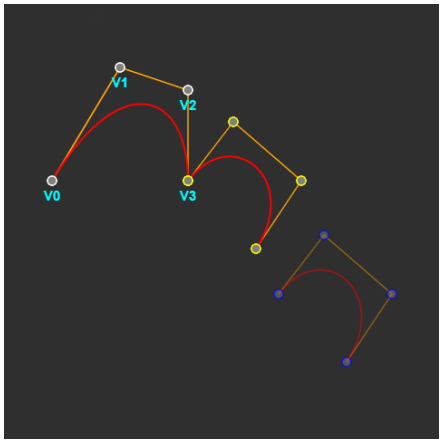
(g) : Bézierova křivka.



(h) : Algoritmus De Casteljau.

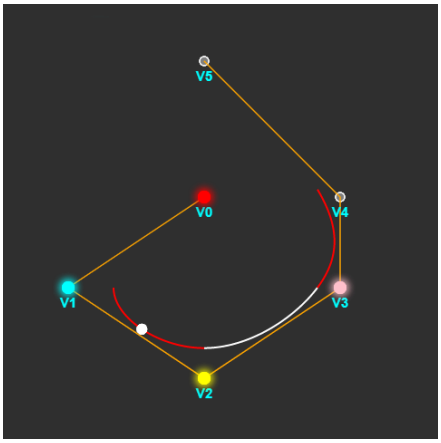


(i) : Rozdělení Bézierovy křivky.

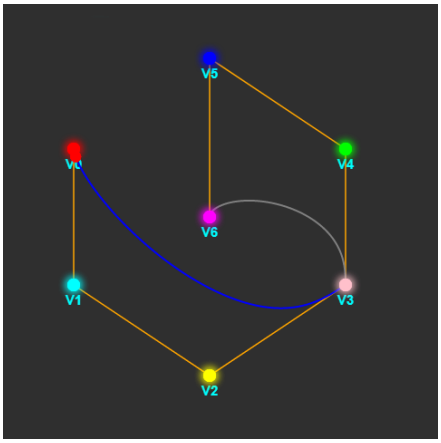


(j) : Spojitost Bézierovy křivky.

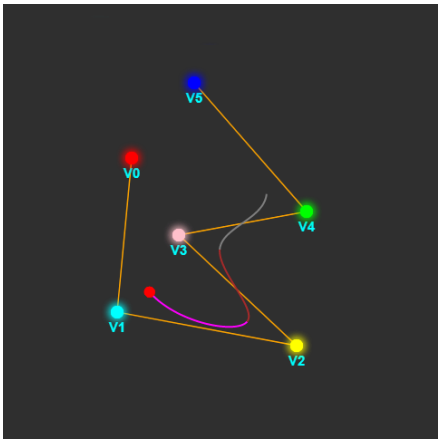
za pomocí HTML radio elementů. Nejprve má nastavit add mód a poté přidat alespoň dva body. Dále má změnit mód na delete a smazat všechny body co zadal. V šesté části má vypnout a poté zapnout vykreslování cesty zadaných bodů. To provede za pomocí HTML checkbox elementu. V sedmé části je uživatel informován o skutečnosti, že křivky mají nutné požadavky, které musí být splněny. Křivka je v této části ve stavu, kdy má nedostatečný počet zadaných bodů. Tato informace se uživateli zobrazuje v jeho menu, kde jsou i ostatní vstupní elementy. Uživatel má za cíl zmenit tento stav přidáním alespoň dvou bodů. Osmá část představí uživatelovi vstupní HTML slider element, kterým se mění čas aktuálního bodu na křivce. Pomocí toho elementu je možné zkoumat průběh křivky. To je užitečné zejména v případě NURBS, kde zdánlivě lineární cesta má své okraje úsečky jinak dlouhé než střed. Cílem uživatele je, aby přesunul aktuální bod v čase na konec křivky. V deváté části má uživatel změnit stupeň křivky na 5. Uživatel je zde také informován o faktu, že některé vstupní elementy jsou zašedivělé a nelze je měnit. Poslední část tutoriálu je zaměřena na bázové funkce křivky. Uživatel má vykresleny bázové funkce za pomocí aplikace. Je zde informován o faktu,



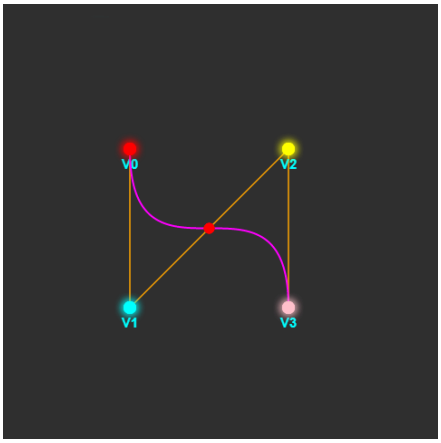
(k) : Coonsova kubika.



(l) : Křivka NURBS.



(m) : Vytvoření Coonsovy kubiky za pomocí NURBS.

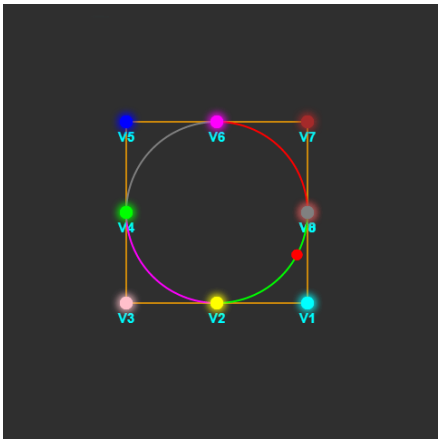


(n) : Vytvoření Bézierovy křivky za pomocí NURBS.

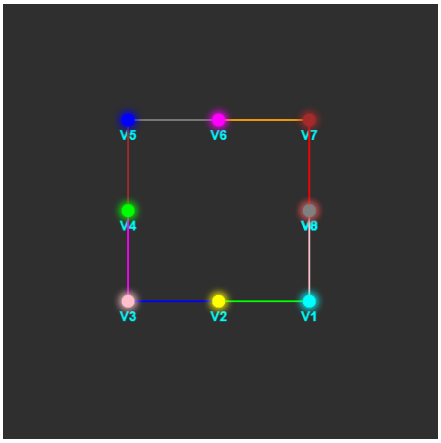
že body souvisejí s jednotlivými bázovými funkcemi a vzájemně si odpovídají barvou. Cílem je, aby uživatel zjistil, jaký je součet všech bázových funkcí. K tomu má použít aplikaci, která mu tyto bázové funkce vykresluje a umožňuje mu sledovat hodnoty podle aktuální pozice myši. Výsledek má zapsat do vstupního HTML elementu v menu.

7.4 Rozšiřování aplikace

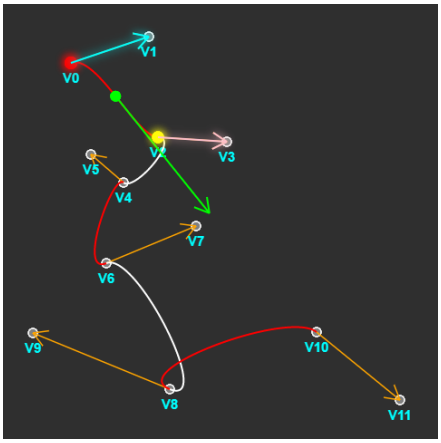
Webová aplikace byla podle návrhu z kapitoly 6 implementována tak, aby ji bylo možné snadno rozšiřovat. Rozšiřovat lze o nové křivky i tutoriály. Aplikace používá relativní cesty, aby ji bylo možné používat na serveru a i lokálně bez přístupu na internet. Nepoužívá se zde cesta kořenového adresáře, který by v případě lokálního přístupu odkazoval na kořenový adresář disku. Proto se využívá aktuální cesta HTML stránky a ostatní části webu (jako například CSS, obrázky a scripty) se získají za pomocí nadřazeného adresáře (tedy ../ prefix).



(o) : Kružnice za pomocí NURBS.



(p) : Čtverec za pomocí NURBS.



(q) : Derivace křivky.

Obrázek 7.1: Ukázky tutoriálů.

V případě, že chceme přidat novou křivku, tak její třídu umístíme do samostatného souboru ve složce *scripts*. Název souboru má odpovídat názvu křivky a má mít koncovku *.js*, aby bylo zřejmé, že se jedná o soubor v jazyce JavaScript. Třída této křivky má dědit od hlavní třídy *Curve*, která je v souboru *common.js*. Název třídy by měl být také stejný jako název křivky. Povinností programátora je, aby křivce implementoval konstruktor, kde bude volán i konstruktor předka. Dále je povinen implementovat alespoň metodu *GetPoint(t, deriv = 0)*, která se používá pro vykreslování a pro získání bodu nebo derivace na křivce v čase *t*.

Chceme-li aplikaci rozšířit o další tutoriál, tak ho budeme přidávat do nějaké ze složek ve složce *examples*. Složky mají název podle toho, s jakou křivkou tutoriál souvisí. Pro obecné tutoriály je vytvořena složka *other*. Každý tutoriál má nejméně dva soubory. První je HTML soubor, který můžeme libovolně pojmenovat a je zakončen koncovkou *.html*. Druhý soubor je v jazyku JavaScript a má stejný název jako HTML soubor, ale má navíc přidáno na konec názvu *-application* a má koncovku *.js*. V tomto souboru je

zdrojový kód pro tutoriál. Nachází se zde třída, která reprezentuje aplikaci a dědí od některé z abstraktních tříd pro aplikace. Zde se také od této třídy vytváří instance. Většina tutoriálů dědí od *CurveApplication* a implementují konstruktor a metodu *Update*. U úvodního tutoriálu se ale dědí od *TutorialApplication* a implementuje se pouze konstruktor, kde se vytváří jednotlivé části s úkoly pro uživatele. V případě HTML souboru je většina částí stejná jako v již implementovaných souborech. Změna bude hlavně v titulku stránky, počtu canvas elementů do kterých se vykresluje a v použitých JavaScript souborech. Obvyklé pořadí JavaScript souborů je, že první je *mathjs*, další je hlavní soubor s abstraktními třídami *common.js*, poté následují potřebné soubory s křivkami a dále již samotný JavaScript soubor, kde je implementovaný tutoriál. Posledním JavaScript souborem je *nav-menu.js*, který obsahuje funkci *AppendNavMenu(idNavMenu, pathToRoot)*. Ta se volá na každé HTML stránce, protože přidává navigaci. Proměnná *idNavMenu* představuje ID elementu, kam se vloží navigace. Proměnná *pathToRoot* je cesta ke kořenovému adresáři webu. Ta je již ze zmíněných důvodů relativní. Chceme-li, aby se nově přidaný tutoriál objevil i v navigaci, musí se tedy upravit funkce *AppendNavMenu(idNavMenu, pathToRoot)*. Změny se díky této šabloně projeví na každé z HTML stránek.

Kapitola 8
Testování

8.1 Uživatelské testování

Tato webová aplikace je určena nejen pro doplnění výuky přednášek, ale i pro samostudium studentů. Z tohoto důvodu bylo třeba aplikaci otestovat. Cílem bylo zjistit, zda jsou studenti schopni aplikaci ovládat a jestli rozumí datovým vstupům, výstupům a požadavkům jednotlivých ukázek aplikace.

Studenti byli během testování pozorováni. Díky tomu bylo možné zjistit, které části jsou nejvíce problémové. Test se skládal z několika úkolů, kterými byli studenti postupně provázeny. V případě, že by student měl vážný problém s některou z částí, bylo mu možné napovědět. Testovaná webová stránka byla dostupná z internetu, což studenti ocenili, protože nemuseli nic nového instalovat a nebylo třeba řešit problémy způsobené různými operačními systémy. Testování bylo rozděleno na tři hlavní části. První část měla za cíl dokončit úvodní tutoriál. Díky němu se měl student naučit jednotlivé ukázky chápat a ovládat. V druhé části se zjišťovalo, zda byl cíl tutoriálu splněn a student skutečně umí s ukázkami pracovat. Třetí část byla tvořena dotazníkem, který ukončuje a vyhodnocuje celý průběh testování.

Úkolů, které provázelo studenty celým testování bylo celkem 13. Všechny tyto úkoly jsou vypsány níže. Nejprve si každý student otevřel dostupnou webovou stránku, přečetl si na ní úvodní text a přešel na úvodní tutoriál. Tutoriál je zde počítán jako jeden samostatný úkol testování. Skládal se však z několika částí, které jsou popsány v kapitole 7.3. Když student tutoriál dokončil, přešel na Coonsovu křivku, kde bylo ověřeno že je schopen s touto křivkou pracovat. Zjišťovalo se, zda umí křivce měnit pozici řídicích bodů, odebírat body a přidávat nové body. Poté byl student vyzván, aby křivce nechal pouze dva body. Díky tomu se křivka dostala do stavu, že má nedostatečný počet bodů a není možné jí vykreslit. Studentům byla předložena otázka, zda poznají jestli křivka splňuje potřebné požadavky a zda vědí, kde tuto informaci mají hledat. Dále byli odkázáni na křivku NURBS, kde měli změnit váhu bodů. Poté bylo testování ukončeno dotazníkem, který se nachází níže pod úkoly k testu.

Testování se účastnilo celkem 10 studentů. Jeden student byl z ČZU a zbylých devět z ČVUT. Ze studentů z ČVUT byl jeden z Fakulty informačních technologii (FIT) a ostatní z Fakulty elektrotechniky (FEL). Všem se povedlo

celé testování dokončit. Otázky, které byly předloženy v úkolu 9. a 10., byly u všech studentů správně zodpovězeny. Závěrečný dotazník, který studenti vyplňovali na konci testování, byl vytvořen za pomocí Google formuláře. Grafy s odpovědmi byly přidány do přílohy C. Procentuální odpovědi jsou uvedeny i v závorkách v dotazníku níže.

Testování odhalilo některé nedostatky webové aplikace. Někteří studenti měli problém trefit bod, který chtěli přesunout. V případě křivky NURBS bylo předpokládáno, že studentovi dojde, že třetí souřadnice bodu je jeho váha. Na to však někteří studenti nepřišli. Některým studentům dělalo také problém čtení hodnot z bázových polynomů. Zmatení pramenilo z označení X a Y souřadnic v grafu. Jednomu studentovi se navíc povedlo odhalit chybu, která nastala při špatném zadání dat do vstupního elementu. Studenti navrhli i další možné vylepšení. Mezi ně patří například zvýraznění chybného statusu červenou barvou, přeskočení kroku v tutoriálu, zobrazit sumu bázových funkcí, možnost získat souřadnice z kliknutí na bázovou funkci, změnit osy grafu z X a Y na *Time* a *Value*. Testování se nejčastěji pohybovalo okolo *9* minut.

Úkoly k testu.

1. Otevřete webový prohlížeč a přejděte na hlavní stránku webové stránky.
2. Přečtěte si úvodní stránku.
3. Přejděte na stránku s tutoriálem.
4. Postupujte podle pokynů tutoriálu a pokuste se ho dokončit.
5. Přejděte na Coonsovu křivku.
6. Změňte pozici alespoň třech různých bodů.
7. Přidejte křivce alespoň dva body.
8. Odeberte tolik bodů, aby křivka měla pouze dva body.
9. Odpovězte na otázku: Splňuje křivka požadavky na to, aby ji bylo možné vykreslit?
10. Odpovězte na otázku: Kde zjistíte, jestli křivka splňuje požadavky?
11. Přejděte na křivku NURBS.
12. Pokuste se změnit váhu bodům.
13. Děkujeme za testování. Nyní přejděte na vyplnění dotazníku.

Dotazník po testu.

1. Jak dlouho vám testování trvalo?
   • do 5 minut (10%) • 5 až 10 minut (50%) • 10 až 20 minut (40%) • více než 20 minut (0%)

2. Dokončili jste testování? (zelený nadpis END)
   • Ano (100%) • Ne (0%)

3. Proč jste testování nedokončili?
   • Vaše odpověď (žádné odpovědi)

4. Všechny části tutoriálu byly snadno pochopitelné a lehce proveditelné.
   • Zcela souhlasím (10%) • Spíše souhlasím (80%) • Nevím (0%) • Spíše nesouhlasím (0%) • Zcela nesouhlasím (10%)

5. Přidat, mazat a měnit řídící body například u Coonsovy křivky bylo snadné.
   • Zcela souhlasím (100%) • Spíše souhlasím (0%) • Nevím (0%) • Spíše nesouhlasím (0%) • Zcela nesouhlasím (0%)

6. Je snadné najít a změnit váhu řídícím bodům u křivky NURBS.
   • Zcela souhlasím (20%) • Spíše souhlasím (60%) • Nevím (0%) • Spíše nesouhlasím (20%) • Zcela nesouhlasím (0%)

7. Je snadné zjistit, že má křivka nedostatečný počet zadaných řídících bodů.
   • Zcela souhlasím (90%) • Spíše souhlasím (0%) • Nevím (0%) • Spíše nesouhlasím (10%) • Zcela nesouhlasím (0%)

8. Je snadné zjistit hodnoty z grafu, který vykresluje bázové funkce.
   • Zcela souhlasím (30%) • Spíše souhlasím (20%) • Nevím (10%) • Spíše nesouhlasím (30%) • Zcela nesouhlasím (10%)

9. Proč jsou ve většině křivek body barevně odlišeny? Je jejich barva s něčím spojená?
   • Vaše odpověď (odpovědi v příloze)

10. Chybí Vám v aplikaci nějaká funkcionalita, křivka, nebo demo?
    • Ne (50%) • Vaše volná odpověď (odpovědi v příloze)

11. Chcete se k testování nebo k samotné aplikaci nějak vyjádřit?
    • Vaše odpověď (odpovědi v příloze)

Kapitola 9

Závěr

Práce se zabývala tvorbou aplikace na podporu výuky základních křivek používaných v počítačové grafice. Jako první byly prozkoumány existující materiály a dema zaměřená na výuku základních křivek. Tomu byla věnovaná kapitola 2. Dále byl proveden uživatelský výzkum, jehož cílem bylo zjistit, jak moc velký problém měli s pochopením křivek absolventi předmětu počítačové grafiky. Bylo zjištěno, že 50% z nich mělo s pochopením křivek problém. Cílem výzkumu bylo také zjistit, jaké výukové aplikace by si studenti byli ochotni spustit na svých počítačích. Z výzkumu vyplynulo, že studentům nevyhovují typy aktuálně dostupných aplikací (Java applet a spustitelný .exe). Nejvhodnější přišla studentům aplikace ve webovém prohlížeči psaná v jazyku JavaScript, kterou by bylo ochotno spustit až 88.9% studentů.

Na základě uživatelského výzkumu a rešerše byly zformulovány požadavky na výukovou aplikaci. Podle nich byla aplikace navrhnuta. Celkem bylo implementováno 5 křivek, pro které bylo vytvořeno 17 tutoriálů. Pro snadné pochopení ovládání aplikace byl také implementován úvodní tutoriál, který se skládal z 10 úkolů pro uživatele. Vytvořená webová aplikace byla dále také testována studenty. Všem studentům se povedlo testování úspěšně dokončit. Na základě výsledků testování byla aplikace upravena, aby byla studentům přívětivější.

Návrh na rozšíření aplikace

Aplikaci by bylo možné rozšířit o další křivky. Nabízí se například eliptické křivky, hyperboly, nebo fraktální křivky. Díky polymorfismu by se nové křivky a i jejich bázové funkce snadno vykreslily. Křivkám by se mohly vytvořit i nové tutoriály. Ty by se mohly věnovat například průsečíkům křivek, převodům mezi jednotlivými křivkami, nebo optimalizaci při vykreslování křivek (s využitím například tesselátorů). Další rozšíření by mohlo být o 3D plochy. Postup pro takové rozšíření aplikace, je vysvětlen v kapitole 6.2.2. Aplikaci by bylo možné použít i pro výuku mimo předmět počítačové grafiky. Příkladem je předmět výpočetní geometrie. V předmětu se vyučují například konvexní obálky. V takovém případě lze využít body aplikace pro konstrukci obálky.

Příloha A

Literatura

[1] Felkel, P.; Sloup, J.: Interpolation and approximation curves and surfaces Part I [online]. 2017, [cit. 2018-12-18]. URL https://cent.felk.cvut.cz/courses/PGR/lectures/10_Curves_I.pdf

[2] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika. 2. vyd.* Computer Press, 2005, ISBN 80-251-0454-0.

[3] Felkel, P.; Sloup, J.: Interpolation and approximation curves and surfaces Part II [online]. 2017, [cit. 2018-12-18]. URL https://cent.felk.cvut.cz/courses/PGR/lectures/11_Curves_II.pdf

[4] Matusik, W.: Bézier Curves and Splines [online]. 2012, [cit. 2019-1-29]. URL https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/MIT6_837F12_Lec01.pdf

[5] Matusik, W.: Curve Properties & Conversion, Surface Representations [online]. 2012, [cit. 2019-1-29]. URL https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/MIT6_837F12_Lec02.pdf

[6] Lentine, M.; Su's, J.; Chaudhuri's, P.: Animation Curves and Splines 1 [online]. [cit. 2019-1-29]. URL https://web.stanford.edu/class/cs248/pdf/class_04_animation_curves_and_splines_1.pdf

[7] Lentine, M.; Su's, J.; Chaudhuri's, P.: Animation Curves and Splines 2 [online]. [cit. 2019-1-29]. URL https://web.stanford.edu/class/cs248/pdf/class_05_animation_curves_and_splines_2.pdf

[8] Keith, P.: Coding Math: Episode 19 - Bezier Curves [online]. [cit. 2019-1-30]. URL https://www.youtube.com/watch?v=dXECQRlmIaE

[9] Kamermans, M.: A Primer on Bézier Curves [online]. [cit. 2019-1-29]. URL https://pomax.github.io/bezierinfo/

[10] Haugland, K.: Generate and understand NURBS curves [online]. [cit. 2019-1-30]. URL https://www.codeproject.com/Articles/1095142/Generate-and-understand-NURBS-curves

[11] Straub, R.; Prautzsch, H.; Kilian, M.; aj.: CAGD-Applets an interactive tutorial on geometric modelling [online]. [cit. 2019-1-30]. URL http://cagd-applets.webarchiv.kit.edu/mocca/html/noplugin/inhalt.html

[12] Linkeová, I.: *Základy počítačového modelování křivek a ploch*. ČVUT Praha., 2012.

[13] Salomon, D.: *Curves and Surfaces for Computer Graphics*. Springer-Verlag New York Inc., 2006, ISBN 0387241965.

[14] Boreskov, A.; Shikin, E.: *Computer Graphics: From Pixels to Programmable Graphics Hardware*. Chapman & Hall/CRC Computer Graphics, Geometric Modeling, and Animation Series, CRC Press, 2013, ISBN 9781482215571. URL https://books.google.cz/books?id=W4bSBQAAQBAJ

[15] Piegl, L.; Wayne, T.: *The NURBS book*. Springer Science & Business Media, 2012, ISBN 978-3-642-97387-1.

[16] Pipenbrinck, N.: Hermite Curve Interpolation [online]. [cit. 2019-4-28]. URL https://www.cubic.org/docs/hermite.htm

[17] Kenneth, J. I.: BERNSTEIN POLYNOMIALS [online]. [cit. 2019-5-5]. URL http://graphics.cs.ucdavis.edu/education/CAGDNotes/Bernstein-Polynomials.pdf

[18] Dodgson, N.: Bezier curves [online]. [cit. 2019-4-30]. URL https://www.cl.cam.ac.uk/teaching/2000/AGraphHCI/SMEG/node3.html

[19] McConnell, J. J.: *Computer Graphics: Theory Into Practice*. Jones and Bartlett Publishers., 2006.

[20] DeBoor, C.: *A Practical Guide to Splines*. Springer New York, 2000, ISBN 9780387989228.

[21] Procházková, J.; Procházka, D.: Implementation of NURBS Curve Derivatives in Engineering Practice [online]. [cit. 2019-5-5]. URL http://wscg.zcu.cz/wscg/Papers_2007/poster/B37-full.pdf

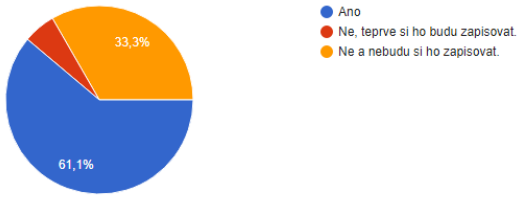
[22] Sinca, B.: SVG vs. Canvas [online]. [cit. 2019-5-7]. URL https://medium.com/@benisinca/svg-vs-canvas-92938aff799a

Příloha B

Dotazník uživatelského výzkumu

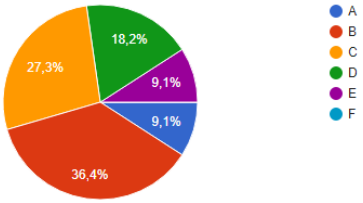
Měli jste zapsán předmět PGR (programování grafiky)?

18 odpovědí



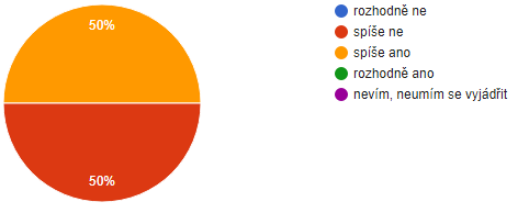
Jakou jste měli známku z PGR?

11 odpovědí

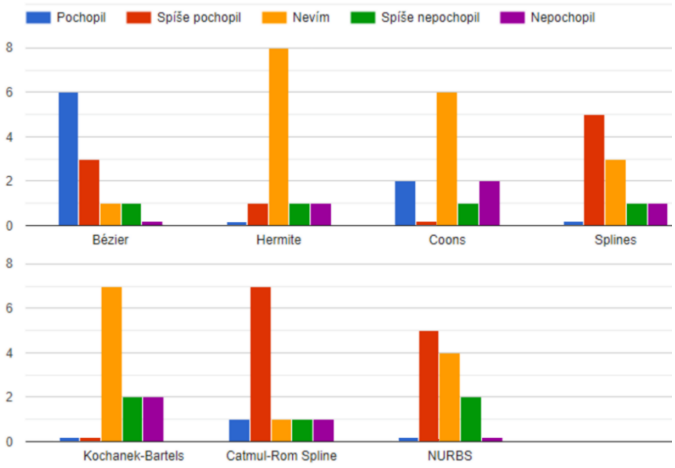


Měli jste problém s pochopením křivek?

10 odpovědí

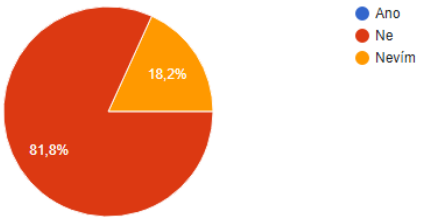


Jak dobře jste pochopili tyto křivky.



Zkusili jste si nějakou aplikaci na výuku křivek v předmětu PGR?

11 odpovědí



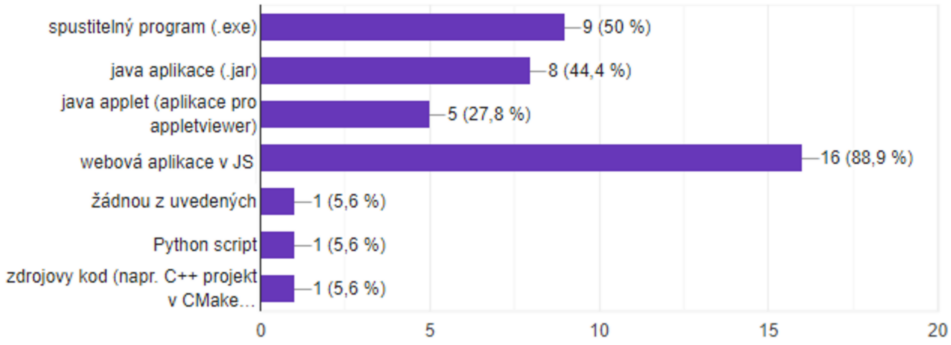
Co byste změnili na výuce křivek?

4 odpovědi

Znatelně chybí aplikace, která by dokázala tuto problematiku jednoduse vysvetlit.
Více času + nějaký domácí úkoly
Méně křivek
Nic mě nenapadá, výuka křivek mi prisla dobra.

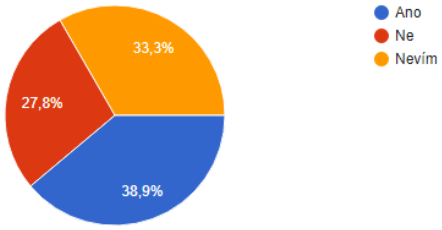
Spustili byste si následující typy výukových aplikací na svém počítači?

18 odpovědí



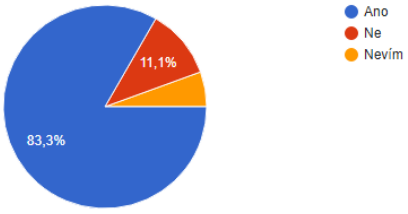
Spustili byste výukovou aplikaci, kdyby byla dostupná pro váš telefon?

18 odpovědí



Upřednostnili byste aplikaci pro PC před mobilní verzí?

18 odpovědí

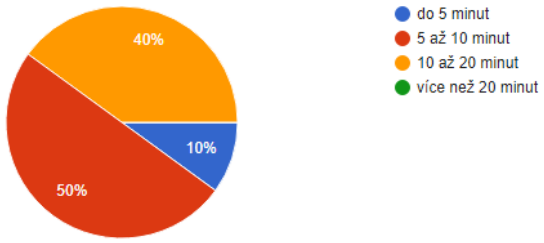


Příloha C

Dotazník k testování

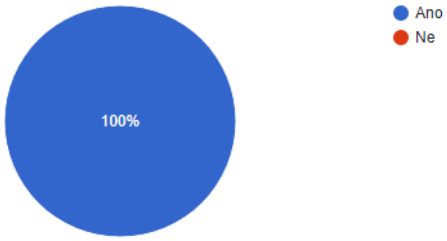
Jak dlouho vám testování trvalo?

10 odpovědí



Dokončili jste testování? (zelený nadpis END)

10 odpovědí



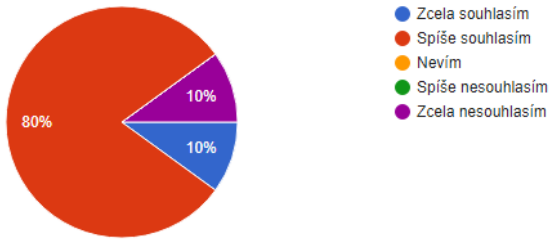
Proč jste testování nedokončili?

0 odpovědí

Na tuto otázku zatím nejsou žádné odpovědi.

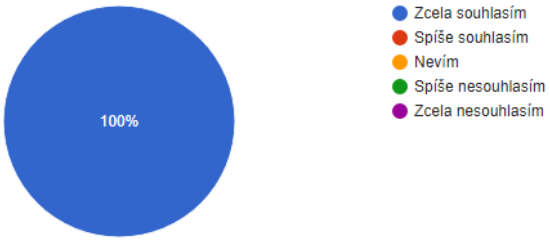
Všechny části tutoriálu byly snadno pochopitelné a lehce proveditelné.

10 odpovědí



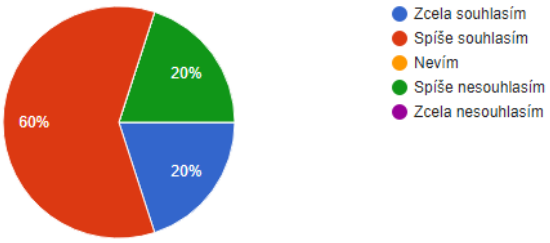
Přidat, mazat a měnit řídící body například u Coonsovy křivky bylo snadné.

10 odpovědí



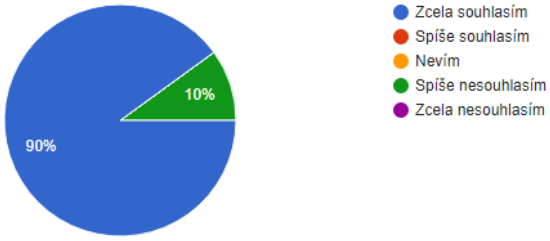
Je snadné najít a změnit váhu řídícím bodům u křivky NURBS.

10 odpovědí



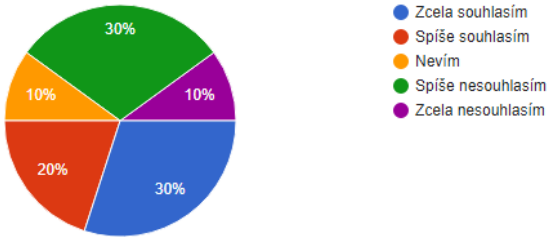
Je snadné zjistit , že má křivka nedostatečný počet zadaných řídících bodů.

10 odpovědí



Je snadné zjistit hodnoty z grafu, který vykresluje bázové funkce.

10 odpovědí



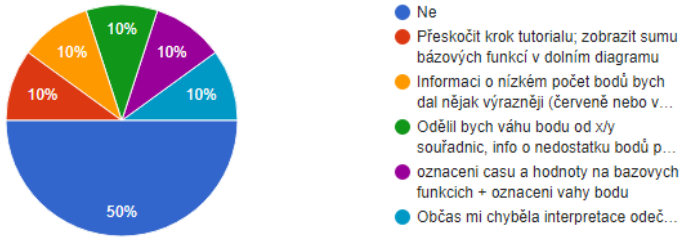
Proč jsou ve většině křivek body barevně odlišeny? Je jejich barva s něčím spojená?

10 odpovědí

Barva souvisí s korespondující bázovou křivkou v grafu, která má stejnou barvu
Urcuji, ktery bod na zacatku krivky je ridici bod.
Je spojena s bázovou funkcí.
s danou bazovou fci
Podle jejich bázových funkcí.
S poradim bodu?
To mě nenapadlo, netušim.
nezapamatoval jsem si
nejak ta barva souvisi s bazovymi funkcemi dole na strance
Nepamatuju si

Chybí Vám v aplikaci nějaká funkcionalita, křivka, nebo demo?

10 odpovědí



Chcete se k testování nebo k samotné aplikaci nějak vyjádřit?

3 odpovědi

:)
good luck u statnic :)
moc se mi libi, jak ti to krasne funguje :D

Příloha D

Uživatelská příručka

Webová stránka aplikace se nachází na CD v adresáři *DP-krivky/web*, kde jsou i všechny potřebné soubory pro její běh. Vzhledem k tomu, jak byla aplikace navrhnuta v kapitole 6, nepotřebuje žádný server. Aplikaci lze spustit pomocí libovolného webového prohlížeče, na kterém funguje JavaScript a podporuje ECMAScript 2015. Hlavní stránka webu je soubor index.html, který se nachází v adresáři *DP-krivky/web*. URL při lokálním spouštění bude začínat *file:///* a dále bude absolutní cesta k souboru index.html. Stránku je také možné spustit přetažením tohoto souboru na ikonu webového prohlížeče, nebo lze dvakrát kliknout na soubor.

V případě, že by byl zájem o nasazení aplikace na server, bylo by nutné adresářům a souborů v *DP-krivky/web* nastavit přístupová práva, aby byly všechny části aplikace na internetu přístupné. Uživatelé by pak přistupovali k aplikaci za pomocí URL severu.

Příloha E

Obsah přiloženého CD

- `DP-krivky` ..... Adresář aplikace
  - `.git` ..... Adresář verzovacího systému GIT
  - `doc` ..... Adresář s dokumentací aplikace
  - `web` ..... Adresář s webovou stránkou aplikace
  - `.gitignore` ..... Soubor verzovacího systému GIT
  - `README.md` ..... Zadání práce (holý text)
- `DP-latex` ..... Adresář se zdrojovými soubory pro LaTeX
- `DP-text` ..... Adresář s textem práce
  - `DP_Vomastek_Michal_2019.pdf` ..... Text práce ve formátu PDF