

Uživatelská příručka

Vítejte ve specificačním dokumentu k programu SimpleKnots, který vznikl jako závěrečný projekt předmětu PV097 v podzimním semestru roku 2012. Zadáni projektu pojednává o trojdimenzionální vizualizaci uzlů pomocí jejich matematické reprezentace. Uzly jsou reprezentovány parametrickými rovnicemi, které jsou poté pomocí samotného programu vykresleny ve ViewPortu podle uživatelem zadaných parametrů. Parametry pro danou křivku jsou poměrně abstraktní proměnné, nicméně uživatelské rozhraní je snad natolik intuitivní, aby bylo snadné mu porozumět a snadno vygenerovat obstojně vypadající uzly.

Tutorial

Program SimpleKnots je rozdělen do tří hlavních částí. V horní části okna se nachází aplikační menu, které sdružuje všechny důležité ovládací prvky. V levé prostřední části se nachází tzv. Viewport, který zobrazuje všechny vizuální elementy při běhu programu. V pravé prostřední části se nachází ovládací panel, kde se pod sebe rovnají ovladače k vlastním uzlům. Zde je možné uzly následně upravovat i pokud vznikly v jiné podobě, než jsme očekávali podle zadaných parametrů.

Jakým stylem tedy v programu SimpleKnots vygenerujeme nějaké vizuálně zajímavé uzly? Je to jednoduché. Pokud se nám nelíbí iniciální uzel, který uvidíme spolu se spuštěním programu, máme dvě možnosti. Buď jej můžeme smazat pomocí tlačítka „Delete knot“ v ovládacím panelu daného uzlu, případně v aplikačním menu, anebo můžeme zvolit položku New Project v aplikačním menu, která nám Vyčistí ViewPort a zbaví nás všech uzlů. Toto tlačítko také poté můžeme používat pokud se nám nelíbí námi dosažené výsledky a chceme smazat například pět uzlů zároveň.

Nový uzel tedy přidáme pomocí Drop-down menu z aplikačního jména označeného jako Scene, v němž následně vybereme volbu Add knot. Po stisku této volby se nám zobrazí dialogové okno, kde máme na výběr z typů uzlů, které program SimpleKnots podporuje, jejich parametrů a barvy. Typy, jenž program SimpleKnots podporují jsou Toroidní, Trochoidní, EpiTrochoidní a Torus uzly. Ty jsou uvnitř zdrojového kódu reprezentovány pomocí parametrických rovnic a následně vykreslovány ve ViewPortu. Počty parametrů se u jednotlivých typů uzlů liší. To je následně zohledněno hned po změně volby typu uzlu, kdy se nám překreslí tabulka s parametry. Názvy parametrů jsou odlišné, což zřejmě může působit zmatečně. Ovšem přistoupil jsem k takovému řešení, neboť jsem u různých tipů našel v literatuře právě takto pojmenované parametry a snažil jsem se zachovat ustálené pojmenování parametrů. Každý to ovšem může vnímat různě. Pokud chceme následně scénu i ukládat, je vhodné v názvu uzlu nepoužít mezeru. Občas se při exportu scény do xml stane chyba, která vyplývá právě z mezery. Je proto vhodné ji v názvu nepoužívat.

Poté, co zadáme vybrané parametry do dialogového okna, stiskneme OK a uzel se nám vykreslí do ViewPortu a jeho „ovladač“ se přidá do ovládacího panelu. Zpětně už jej není možné přejmenovat. V ovladači je možné následně změnit barvu, počet bodů, kterými je křivka proložena (implicitní hodnota je nastavena na optimálních 180, při menší hodnotě uzly nevypadají moc vzhledně, ale zase při vysokém počtu bodů program ztrácí na svižnost), X, Y a Z souřadnice – je tedy možno uzel posouvat po Viewportu jak se nám líbí. Je možné jej taky rotovat v úhlech -180 až 180 stupňů ve směru všech tří os. Následně je samozřejmě také možné upravit parametry pomocí nichž je uzel zadán. Co se týče hodnot vstupních parametrů, tak jsou shora omezeny hodnotou 20. Při zadávání parametrů celočíselně uzly negenerují

Uživatelská příručka

nějaké nežádoucí artefakty. To se může stát při zadání neceločíselných hodnot, ovšem někdy i neceločíselné hodnoty mohou vytvořit pěkné vizuální efekty. Hodnoty je možné zadávat jak pomocí posuvníku, který má standardní krok nastavený na hodnotu 1 (tj. při kliknutí do prostoru, po kterém se jezdec pohybuje se hodnota o 1 zvýší/sníží), ovšem při táhnutí jezdcu je možné takto zadat hodnoty i neceločíselné. Hodnoty je možné zadávat i do textových polí, které jsou s hodnotami jezdců svázány. Dále je pak v ovladači ke každému uzlu ještě tlačítko pro jeho smazání.

K ovládání ViewPortu je používána myš, případně tlačítka na klávesnici. Tlačítka na klávesnici umí objekt ve středu ViewPortu rotovat podle směru šipky. Pomocí pravého tlačítka myši je možné ViewPortem pohybovat. Kombinací Shift + pravé tlačítko myši můžeme objekt ve ViewPortu pohybovat do stran. Pomocí Ctrl + pravé tlačítko myši a pohyb vpřed či vzat můžeme objekt přibližovat či oddalovat. Kliknutím na libovolnou ze stran krychle, které slouží jako orientační body uvnitř ViewPortu se nám model otočí do pohledu nárýsu/půdorysu/bokorysu soudě dle hrany na kterou jsme klikli. Zpět do "3D" se lze dostat opět pomocí pravého tlačítka myši, případně šipkami.

Použité technologie

Projekt je postaven na .Net Frameworku 4.5, přestože věřím, že je kompatibilní i s Frameworkem 3. Použil jsem také podpůrný Helix Toolkit, který je lehkou nadstavbou nad knihovnou WPF 3D. Z Helix Toolkitu je použit upravená verze ViewPortu a také objekt TubeVisual3D, který je použit k vizuální reprezentaci samotných uzlů. Také jsem použil jednu externí GUI komponentu, konkrétně se jedná o ovládací prvek, kterým lze vybrat barvu uzlů z balíku Extended WPF Toolkit. Vše ostatní je už mnou vypracovaný kód.

Uzly jsou reprezentovány pomocí základní třídy Knot, z níž jsou následně odvozeny další konkrétní třídy reprezentující jednotlivé typy uzlů. V základní třídě je zapouzdřena největší část atributů (pro všechny typy uzlů společné jako barva, souřadnice atd.). V třídách reprezentujících konkrétní typy uzlů jsou uchovány prakticky jen jejich vlastní parametry a překrytí abstraktních metod z třídy Knot.

Hodnocení projektu

Předně jsem rád za to, že jsem se konečně ponořil do tajů knihovny WPF 3D, protože si myslím, že je ne úplně známá, přesto se s ní pracuje poměrně snadno. Co jsem ovšem podcenil je její vlastní smysl existence pokud se to tak dá nazvat. Sám jsem trochu doufal, že půjde poměrně jednoduše modelovat zajímavé scény včetně stínů a textur, jako například s použitím DirectX, to ovšem u této knihovny možné není. Je vhodná spíše na vizualizaci jednoduchých 3D objektů bez použití komplikovaných materiálů a osvětlení. Potýkal jsem se taky s, pravděpodobně, bugem co se ViewPortu týče. Je schopné z něj exportovat pouze do obrázku s přesně 2x většími rozměry v každém směru než je ViewPort sám. Zkoušel jsem exportovat do poměrově ekvivalentních velikostí souborů a po několika hodinách se mi nepodařilo získat odpovídající výstupy. Je to velká škoda pro WPF 3D. Takovéto malichernosti vývojáře ne vždy potěší.

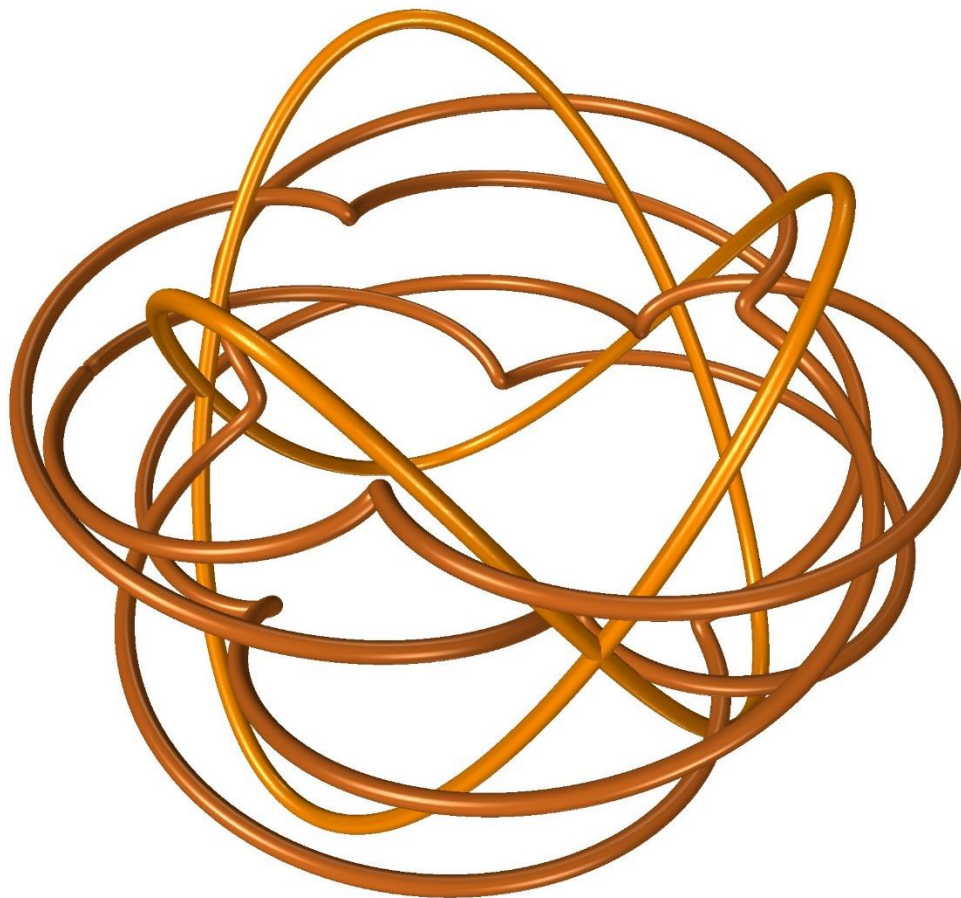
Za co bych se zkritizoval a to otevřeně je absence načítání uložené scény z XML, do kterého byla předtím exportována. Na nic se vymlouvat nebudu. Prostě jsem to NEzvládl implementovat. I přestože jde o poměrně triviální parsování XML souboru, nebyl jsem schopný dodat rozumně fungující verzi, která by scénu věrně rekonstruovala. Určitě chci na program ještě dál pracovat kvůli prohloubení znalostí s WPF 3D, toto bude jedna z prvních věcí, do kterých se chci pustit. Tuším, že jsem tím nesplnil zadání. Jsem připraven nést odpovídající důsledky této nedostatečnosti.

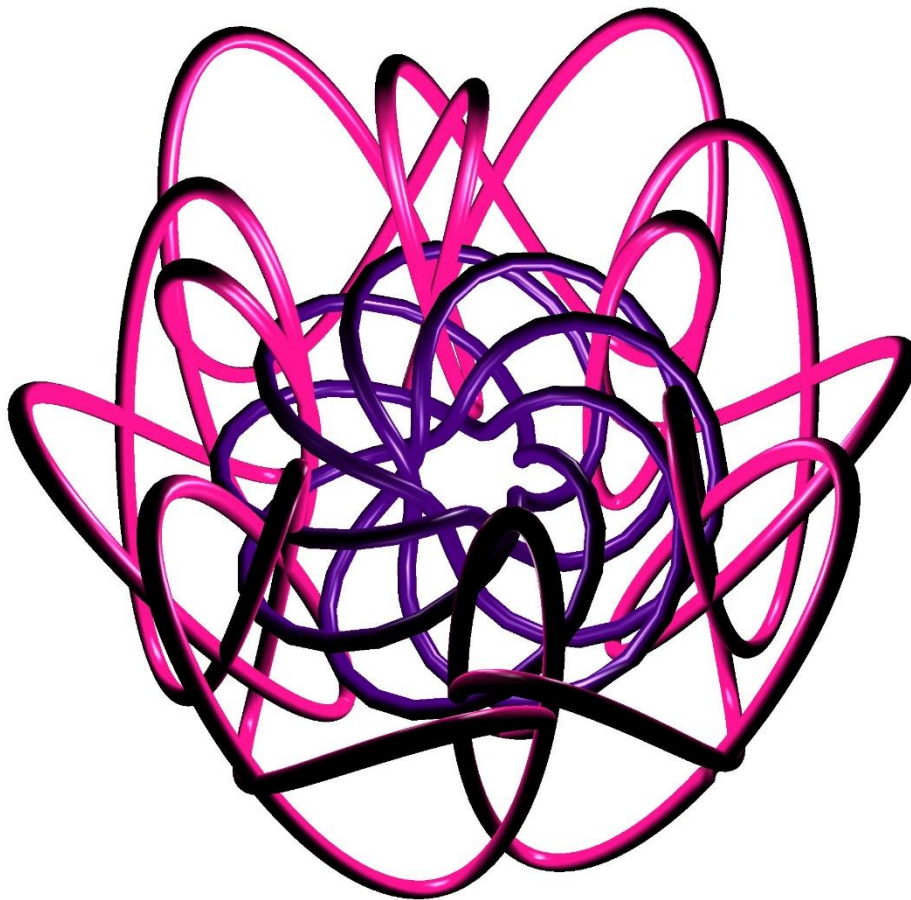
Erik Vaněk, #359416

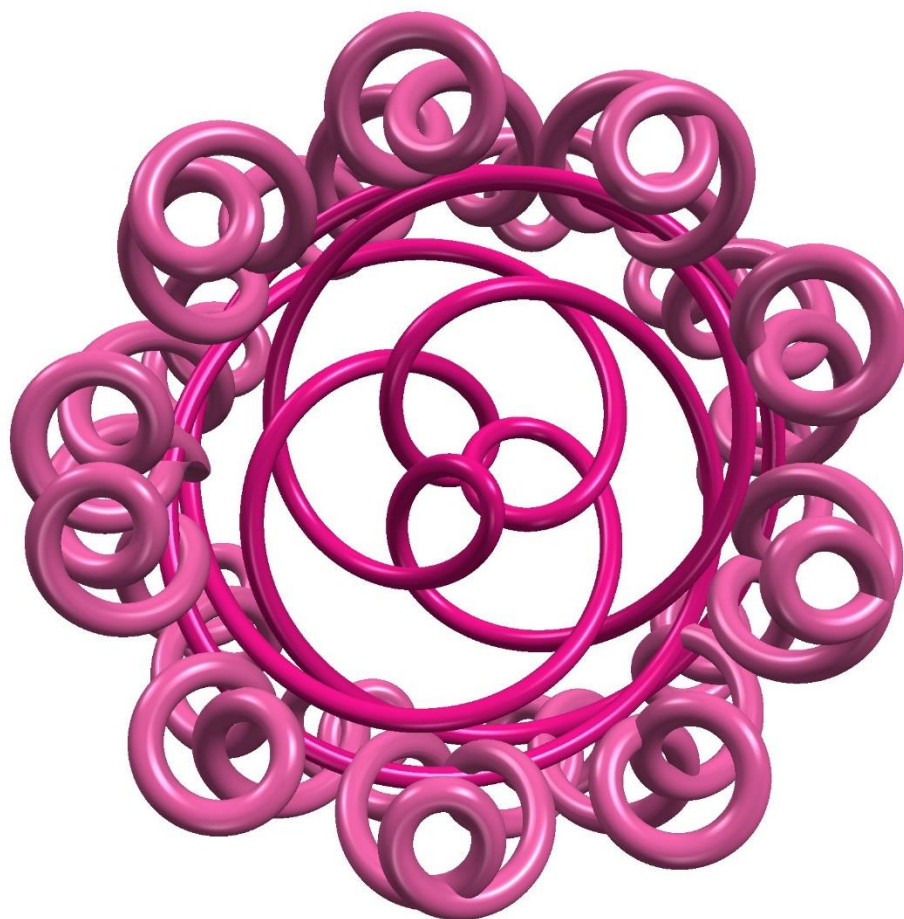
PV097

Uživatelská příručka

[Ukázky](#)







Erik Vaněk, #359416

PV097

Uživatelská příručka

