Podstawy Grafiki Komputerowej

Dokumentacja projektu 43

Krzywe Lissajous w 3D

Karolina Klimek, Mateusz Lewandowski, Natalia Przetocka

20 czerwca 2022

Spis treści

1 Opis projektu

Celem projektu jest stworzenie programu generującego odpowiednik krzywych Lissajous w 3D. Okno programu zawiera cztery główne części:

1. Panel na rysowaną krzywą.
2. Pola do wpisywania parametrów.
3. Panel z suwakami do obrotu narysowanej krzywej.
4. Pola do wyboru stylu rysowania krzywej.

2 Założenia wstępne przyjęte w realizacji projektu

Program daje podgląd na rysowaną krzywą Lissajous w 3D z możliwością zmiany parametrów dla współrzędnych biegunowych i kartezjańskich, rysowania krzywej za pomocą krzywych lub punktów oraz rotacji krzywej względem osi X, Y, Z. Program daje możliwość zmiany stylu rysowania krzywej na statyczną lub animowaną (jest to zrealizowanie założeń rozszerzonych).

3 Analiza projektu

3.1 Specyfikacja danych wejściowych

3.1.1 Pola do wpisywania parametrów

Modyfikacji krzywych można dokonywać wpisując wartości liczbowe parametrów A, ϕ, θ dla każdej z krzywych. Zmiany implementowane są wraz z edycją danych w oknie. Przy podawaniu danych liczbowych zmiennoprzecinkowych należy używać kropki jako separatora.

3.1.2 Panel z suwakami

Wygenerowane krzywe można obracać używając suwaków względem osi X, Y oraz Z.

3.1.3 Pola do wyboru stylu rysowania krzywej

Istnieje opcja wyboru rysowania krzywych przestrzennych jako zbioru punktów lub zbioru odcinków, krzywej statycznej lub animowanej oraz opcja rysowania za pomocą współrzędnych kartezjańskich albo biegunowych.

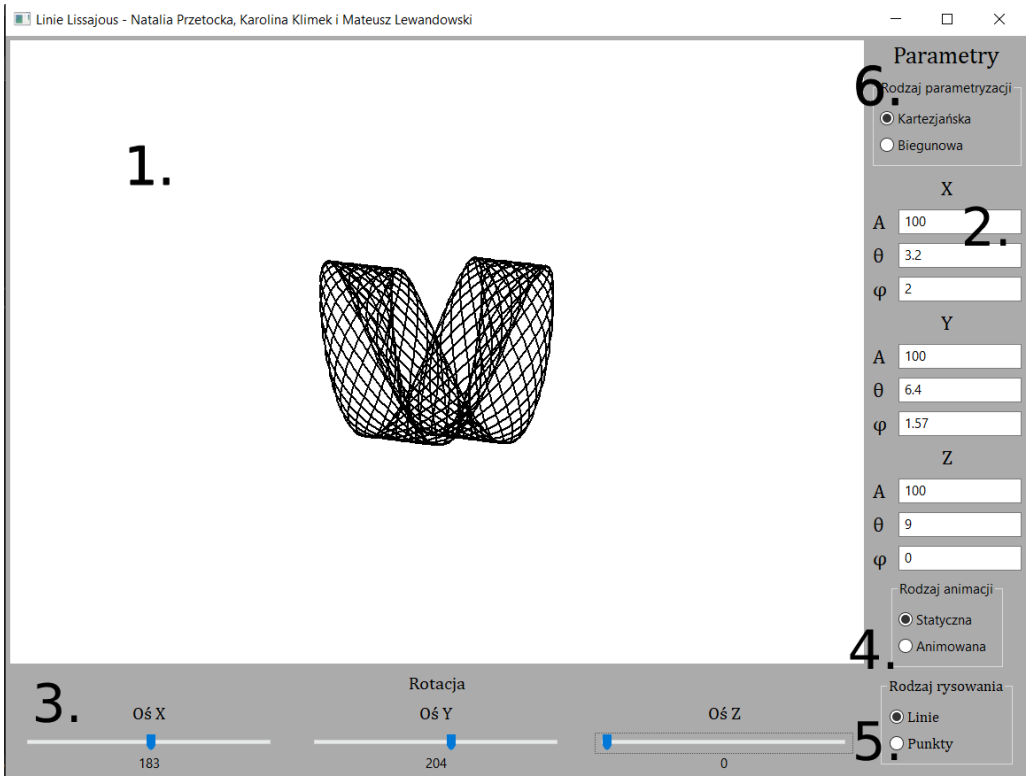
3.2 Zdefiniowanie struktur danych

Za pomocą programu *wxFormBuilder* wygenerowano klasę $MyFrame1$ z interfejsem użytkownika i wirtualnymi metodami związanymi z edycją krzywej. Następnie zaimplementowano klasę $GUIMyFrame$, pochodną po $MyFrame1$, która operuje zmianami wprowadzonymi przez użytkownika do aplikacji.

Następnie stworzono klasę $ConfigClass$ przechowującą parametry krzywej i metody je zmieniające oraz klasę $ChartClass$ z metodą, która rysuje krzywą.

Wykorzystaliśmy również klasy $Vector4$ i $Matrix4$ wraz z zaimplementowanymi metodami, stworzone przez prowadzącego przedmiot *Podstawy Grafiki Komputerowej*.

3.3 Specyfikacja interfejsu użytkownika



Rysunek 1: Interfejs użytkownika.

1. Panel na rysowaną krzywą.
2. Pola do wpisywania parametrów.
3. Panel z suwakami do obrotu narysowanej krzywej.
4. Pola do wyboru rodzaju animacji.
5. Pola do wyboru rodzaju rysowania.
6. Pola do wyboru rodzaju parametryzacji.

3.4 Wyodrębnienie i zdefiniowanie zadań

3.4.1 Interfejs

Wygenerowanie interfejsu przez program $wxFormBuilder$.

3.4.2 Krzywa

Utworzenie klasy przechowującej parametry krzywej oraz metody wprowadzające zmiany parametrów krzywej.

3.4.3 Rysowanie krzywej

Utworzenie klasy z metodą rysującą krzywą.

3.5 Decyzja o wyborze narzędzi programistycznych

Projekt został napisany w języku $C++$ w standardzie $C++17$ wraz z biblioteką $wxWidgets$.

Do wykonania projektu wykorzystano środowisko programistyczne $Microsoft\ Visual\ Studio$ z wbudowanym kompilatorem. Środowisko zostało wybrane ze względu na wsparcie dla repozytorium Git i hostingu $GitHub$ co ułatwiało współpracę, oraz na wygodę użycia z biblioteką $wxWidgets$.

4 Podział pracy i analiza czasowa

Pracę rozpoczęto poprzez wygenerowanie interfejsu w programie $wxFormBuilder$. Następnie rozszerzono program o potrzebne pliki i zaimplementowano wstępnie metodę rysującą "na sztywno" tzn. z zadanymi w kodzie parametrami.

Następnie zaimplementowano rotację widoku na obraz i napisanie metod pozwalających na pobieranie danych od użytkownika. Ostatecznie zaimplementowano również pola zaznaczania, pozwalające na wybór metody rysowania (linie lub punkty, animowane lub statyczne oraz za pomocą współrzędnych kartezjańskich lub biegunowych).

Napisanie projektu zajęło około 2.5 tygodnia. Wspólnie zaprojektowano aplikację w $wxFormBuilder$ i podstawową strukturę programu, Mateusz zajął się rysowaniem krzywej oraz jej animacją, Karolina zaimplementowała metody rotacji i rysowania za pomocą współrzędnych kartezjańskich i biegunowych wraz z częścią dokumentacji, a Natalia zaimplementowała metody zmiany parametrów, rysowania kropek lub punktów i dokumentację.

5 Opracowanie i opis niezbędnych algorytmów

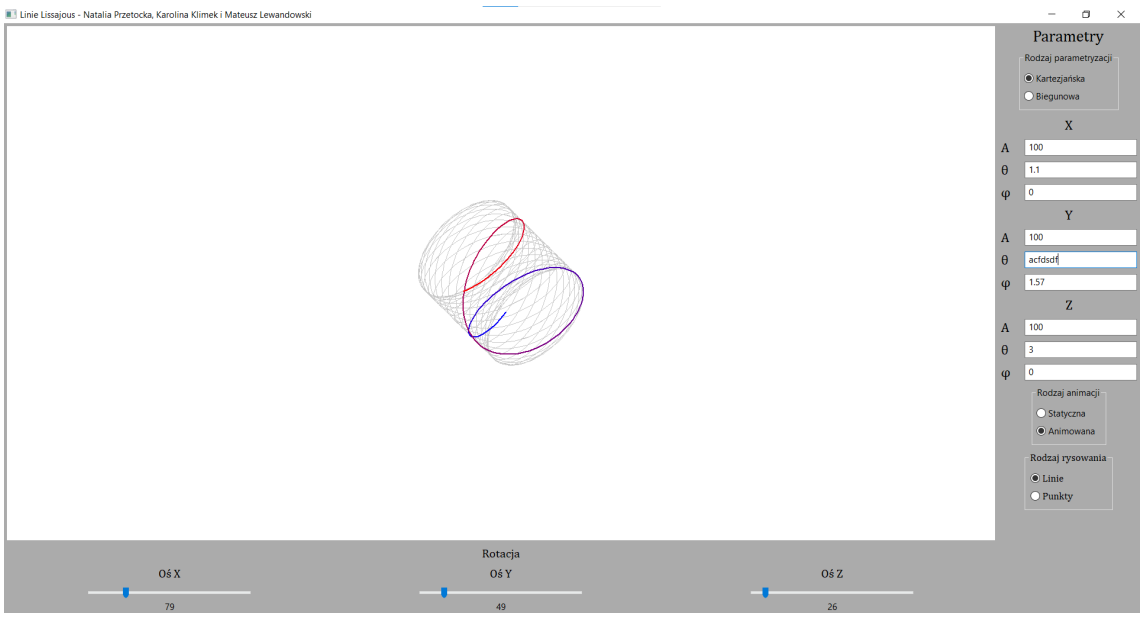
Konieczne do realizacji programu było zaimplementowanie algorytmu, który pozwalałby na reinterpretację punktów znajdujących się w przestrzeni na płaski obraz z zachowaniem ich trójwymiarowego wyglądu. Wykorzystano do tego informacje zawarte na ten temat w wykładzie dotyczącym $Grafiki\ 3D$ oraz implementację macierzy i wektora zapewnioną przez prowadzącego przedmiot. Odpowiednie mnożenie macierzy i wektorów pozwoliło na uzyskanie pożądanego wyniku.

6 Kodowanie

Dokumentacja została wygenerowana za pomocą $Doxygen$.

7 Testowanie

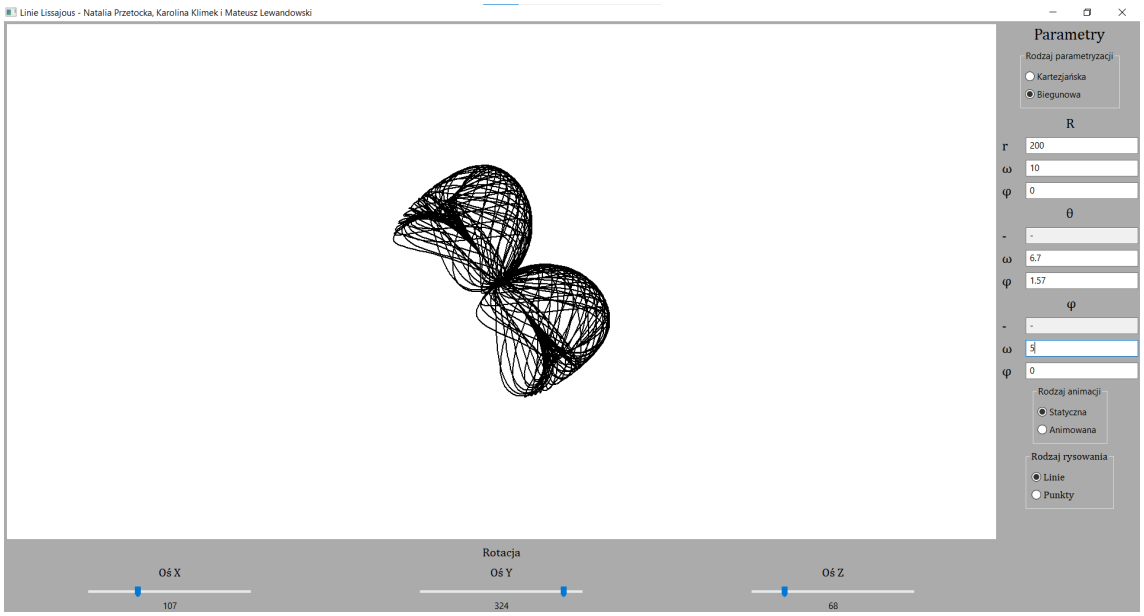
Testowanie programu rozpoczęto od wpisywania jako parametrów liter lub wartości nieliczbowych.



Rysunek 2: Program z parametrami ustawionymi jako wartości nieliczbowe.

W przypadku gdy użytkownik wprowadzi wartości nieliczbowe jako parametry krzywej, program zostawia krzywą jak dla ostatnich poprawnie wpisanych parametrów.

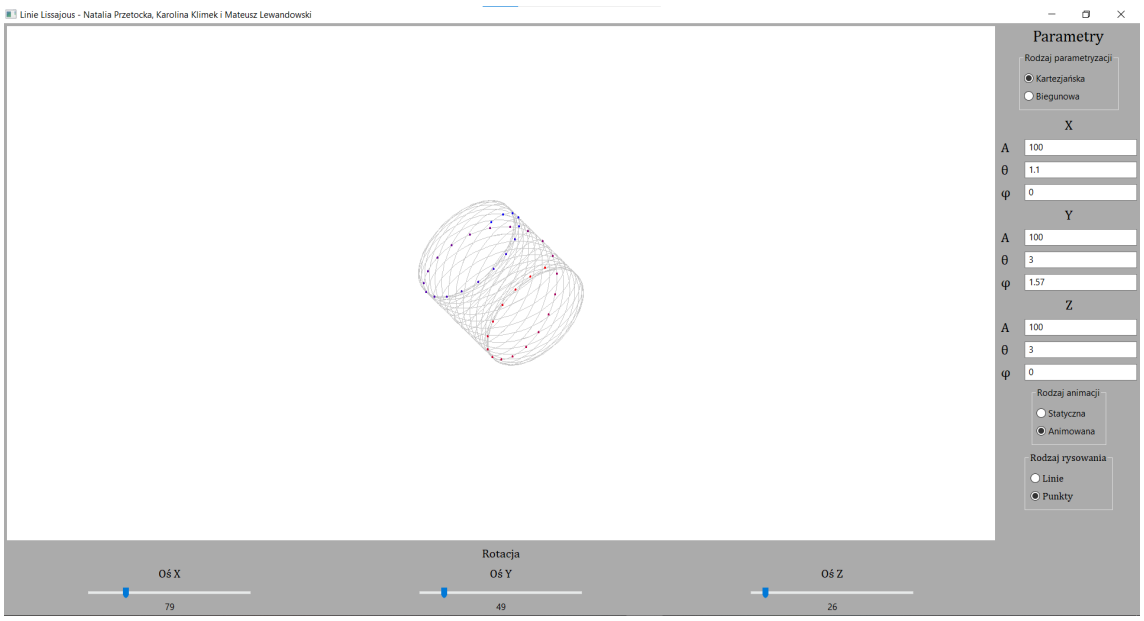
Następnie sprawdzono generowanie krzywych za pomocą współrzędnych biegunowych.



Rysunek 3: Krzywa wygenerowana za pomocą współrzędnych biegunowych.

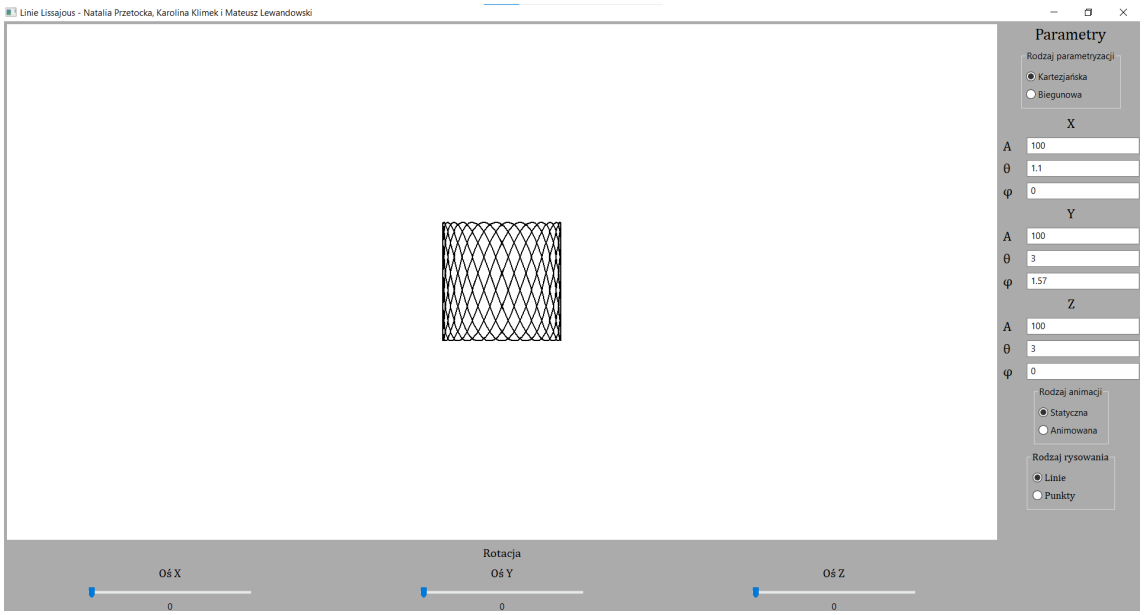
Krzywa generuje się poprawnie.

Następnie testowano rysowanie krzywej za pomocą punktów. Krzywa generuje się poprawnie.

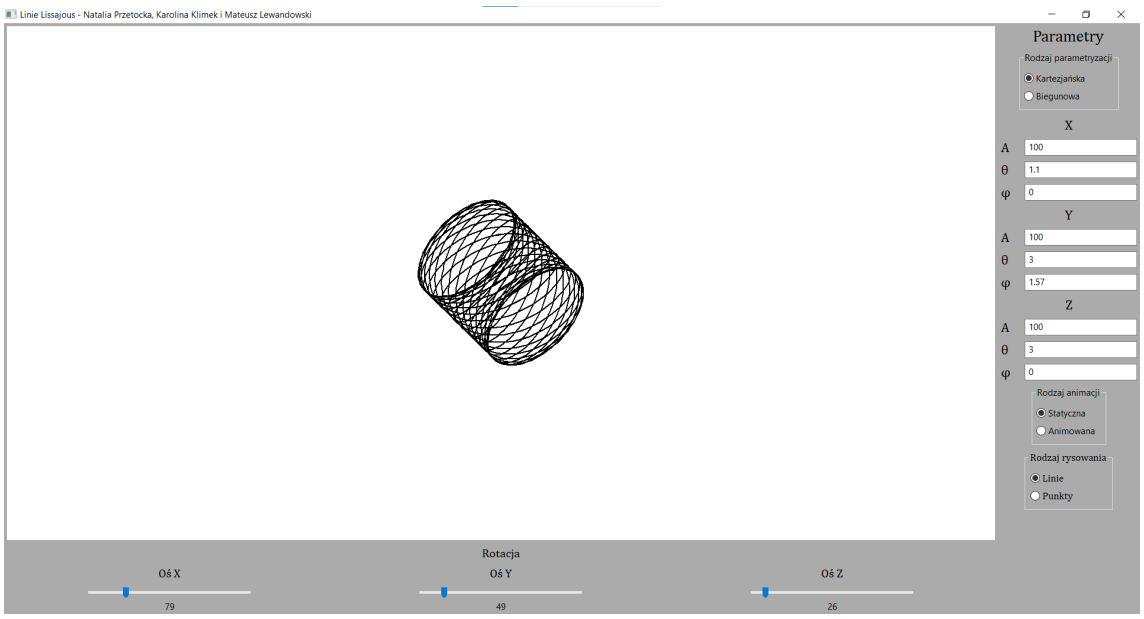


Rysunek 4: Krzywa narysowana jako zbiór punktów.

Obracanie krzywej względem osi X, Y i Z.



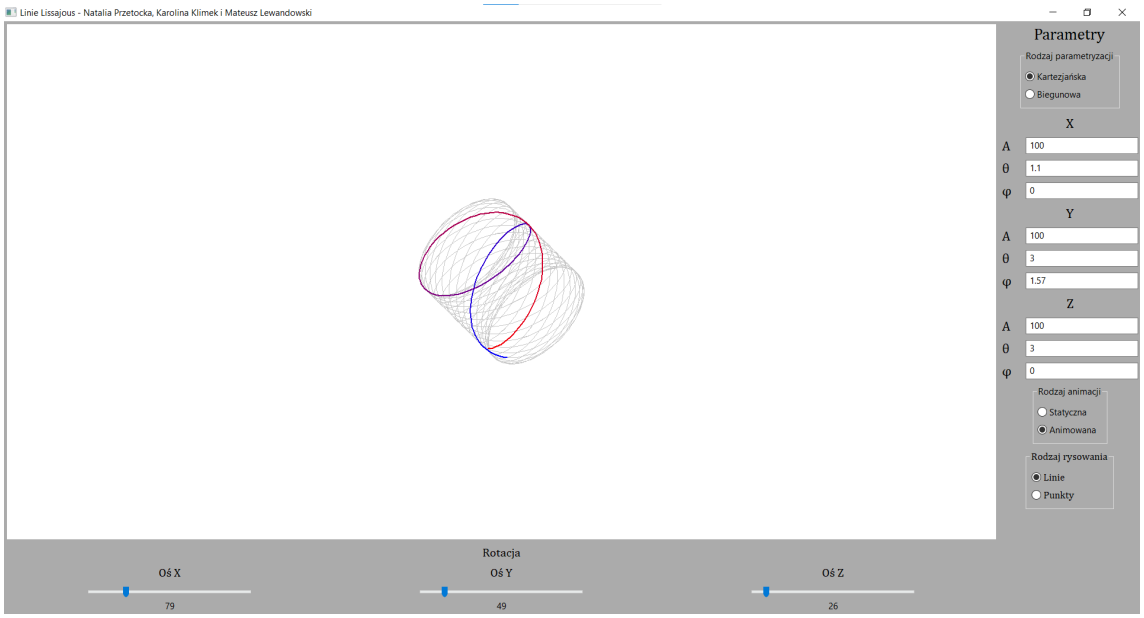
Rysunek 5: Krzywa przed obrotem.



Rysunek 6: Krzywa po obrocie.

Obracanie krzywej przebiegło poprawnie.

Animacja krzywej.



Rysunek 7: Animowana krzywa.

Krzywa animuje się.

8 Wdrożenie, raport i wnioski

Projekt zrealizował wszystkie wymagania podstawowe, pozwala na generowanie krzywych Lissajous w 3D i edycję parametrów rysowanej krzywej. Poprawnie działa rysowanie krzywej za pomocą współrzędnych biegunowych oraz kartezjańskich i rysowanie za pomocą kropek lub odcinków.

Wymagania rozszerzone zostały spełnione. Istnieje opcja animowania krzywej. Cała krzywa jest zaznaczona szarym kolorem. Jeden odcinek o zadanej długości porusza się po szarym zaznaczeniu, na początku pojawiają się nowe odcinki (lub punkty), a na końcu znikają odcinki (lub punkty).

Dodatkowo początek ruszającego się odcinka ma kolor czerwony i przechodzi gradientem w kolor niebieski. Jednak po pewnym czasie działania programu z animowaną krzywą pojawia się błąd i program przestaje działać.

Dla lepszego działania można byłoby pomyśleć nad nad dodaniem większej ilości rodzajów animacji lub suwaka szybkości animacji. Można byłoby również dać możliwość użytkownikowi na wpisanie ilości kropek, z której rysowano by krzywą.

Praca nad programem pozwoliła nam na zapoznanie się z biblioteką *wxWidgets* oraz innymi narzędziami takimi jak *GitHub*, *Overleaf* i *Visual Studio*.