Interferencja Fal

Maciej Piwek

Szymon Pietraszek

Kacper Żółkiewski

1. Wstęp

Celem projektu było zaprezentowanie interferencji dwóch fal na powierzchni wody, membrany (w przestrzeni 3D).

2. Krótki opis

Program jako dane wejściowe przyjmuje współrzędne źródeł obu fal oraz częstotliwości i amplitudy ich drgań. W kolejnym tworzy w pamięci komputera animację (w rzucie 3D) prezentującą rozchodzenie się obu fal oraz ich interferencję. Następnie animacja jest prezentowana na ekranie komputera. Dodatkowo program posiada możliwość wyświetlania pojedynczej klatki animacji. Zastosowane przez nas technologie to:

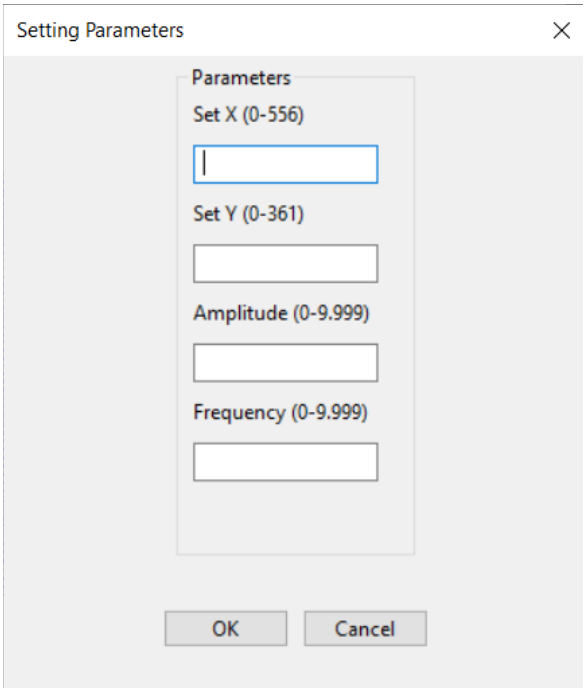
- Standardowa Biblioteka języka C++
- wieloplatformowa biblioteka klas C++ WxWidgets 3.0.5
- wxFormBuilder (version Jun 17 2015)

3. Opis szczegółowy

3.1 Dane wejściowe

Użytkownik programu posiada możliwość ustawienia parametrów dla dwóch źródeł fal:

- x, y - odpowiada za położenie generowane źródła fal
- amplitude - odpowiada za amplitudę generowanej fali
- frequency - odpowiada za częstotliwość generowanej fali

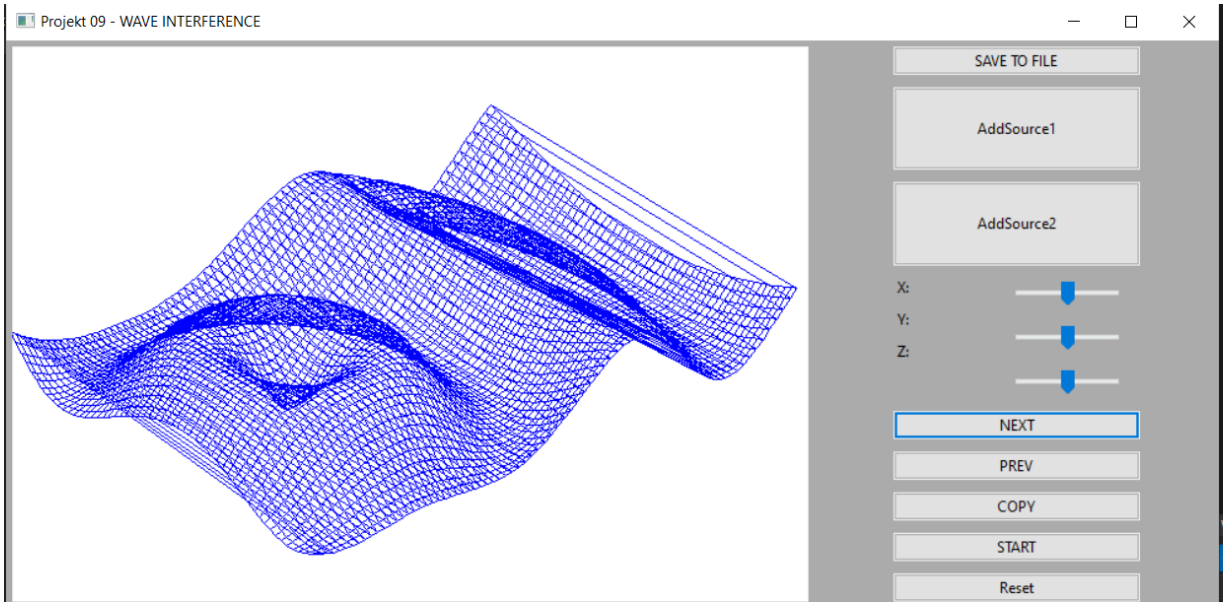


Istnieje również możliwość obrotu płaszczyzny, wokół osi: x, y, z , za pomocą regulowanych suwaków.

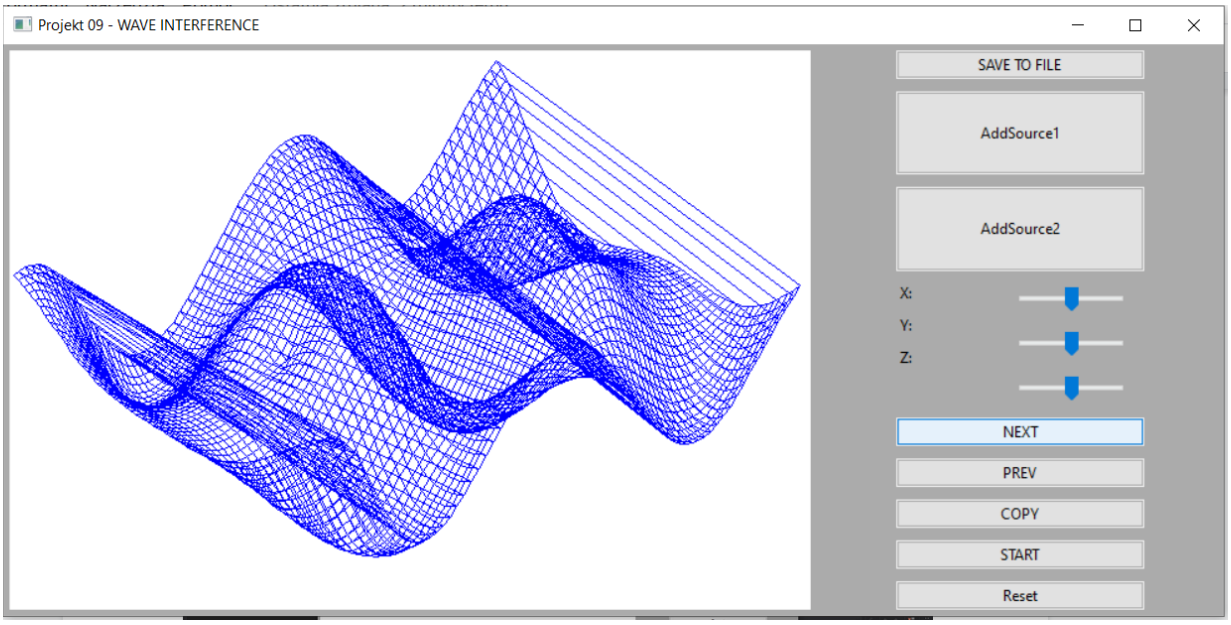
3.2 Dane wyjściowe

Na wyjściu zostaje zaprezentowana animacja 3D interferencji fal

Dla 1 źródła:



Dla 2 źródeł



3.3 Struktura

Specyfikacja klas:

MyFrame2 – zapewnia obsługę głównego okna. Jej metody są przeciążone w klasie GUIMyFrame. Klasa ta została automatycznie wygenerowana przy użyciu WxFormBuildera.

GUIMyFrame – zapewnia implementację dziedziczonej klasy MyFrame2.

Publiczne pola:

- wxImage interferenceImage - obiekt klasy obsługującej tworzenie zdjęcia z bitmapy
- wxBitmap myBuffer - obiekt klasy, na której rysujemy
- wxClientDC myClient - obiekt klasy dzięki, której umożliwione jest rysowanie na panelu klienta
- wxTimer timer - obiekt klasy umożliwiający wykonywanie kodu w określonych interwałach
- myMatrix transformMatrix - macierz 4D, w której przechowywane są dane o transformacjach
- unsigned seconds = 0 - zmienna pomocnicza wykorzystywana do przesuwania fali w czasie, również jako przesuwanie pojedynczych klatek
- int duration = 6 - czas trwania symulacji
- double time = 0.0 - zmienna pomocnicza do obsługi czasu
- double scaleX, scaleY - wartość, dzięki której punkty są rozmieszczone równomiernie
- unsigned pointsX, pointsY - rozmiar siatki(powierzchni wody) na której prezentowana jest symulacja
- std::vector<std::vector<wxPoint>> drawPoints - punkty po transformacjach tworzące animacje
- std::vector<std::vector<wxPoint>> points - punkty tworzące płaszczyznę

Dla pierwszego źródła:

- std::vector<std::vector<double>> points1Distance - odległość punktu od ustawionego przez usera pierwszego źródła
- double points1Amplitude - przechowuje amplitude
- double points1Frequency - przechowuje częstotliwość
- bool flag1 = true - pomocnicza zmienna do aktualizacji widoku

Dla drugiego źródła:

- std::vector<std::vector<double>> points2Distance - odległość punktu od ustawionego przez usera drugiego źródła
- double points2Amplitude - przechowuje amplitude
- double points2Frequency - przechowuje częstotliwość
- bool flag2 = true - pomocnicza zmienna do aktualizacji widoku

Publiczne metody:

- void onUpdateUI(wxUpdateUIEvent& event) - metoda aktualizująca panel, kiedy program jest uruchomiony
- void saveToFileClick(wxCommandEvent& event) - metoda obsługująca zapis obrazka

- void on_addSource1Click(wxCommandEvent& event) - metoda obsługująca dodanie pierwszego źródła
- void on_addSource2Click(wxCommandEvent& event) - metoda obsługująca dodanie drugiego źródła
- void onScrollX(wxScrollEvent& event) - metoda pośrednicząca umożliwiająca obsługę suwaka kontrolującego obrót wokół osi OX
- void onScrollY(wxScrollEvent& event) - metoda pośrednicząca umożliwiająca obsługę suwaka kontrolującego obrót wokół osi OY
- void onScrollZ(wxScrollEvent& event) - metoda pośrednicząca umożliwiająca obsługę suwaka kontrolującego obrót wokół osi OZ
- void nextClick(wxCommandEvent& event) - metoda umożliwiająca przejście do następnej klatki symulacji
- void prevClick(wxCommandEvent& event) - metoda umożliwiająca cofnięcie o jedną klatkę symulację
- void copyClick(wxCommandEvent& event) - metoda obsługująca kopiowanie widoku z panelu do schowka
- void startClick(wxCommandEvent& event) - metoda obsługująca start symulacji po wciśnięciu przycisku
- void resetClick(wxCommandEvent& event) - metoda resetująca podane parametry, tafla wraca do stanu pierwotnego
- void Paint() - metoda umożliwiająca rysowanie na panelu
- void scroll() - metoda umożliwiająca obrót wokół osi
- double measureDistance(const double x1, const double y1, const double x2, const double y2) - obliczanie dystansu, dla zadanych koordynat
- void addSource(std::vector<std::vector<wxPoint>> &points, … , bool &flag, unsigned &counter) - metoda pośrednicząca przy dodawaniu źródeł
- void initializeVec(unsigned pointsX, unsigned pointsY, double scaleX, double scaleY)
- void clearDistance(std::vector<std::vector<wxPoint>> &points, std::vector<std::vector<double>> &pointsDistance, double &pointsAmplitude, double &pointsFrequency) - metoda obsługująca czyszczenie danych z wektorów

MyDialog – zapewnia obsługę wyskakujące okna, które pobiera parametry od użytkownika. Klasa ta została automatycznie wygenerowana przy użyciu WxFormBuildera.

Prywatne pola:

double x - zmienna przechowująca wartość dla x wprowadzoną przez użytkownika

double y - zmienna przechowująca wartość dla y wprowadzoną przez użytkownika

double amplitude - zmienna przechowująca wartość dla amplitudy wprowadzoną przez użytkownika

double frequency - zmienna przechowująca wartość dla częstotliwości wprowadzoną przez użytkownika

bool added - zmienia swoją wartość na false jeżeli zostanie wprowadzona wartość nie spełniająca wymogów

Pola chronione:

Obsługa wprowadzanej wartości x:

- wxStaticText* m_staticText_x
- wxTextCtrl* m_textCtrl_x

Obsługa wprowadzanej wartości y:

- wxStaticText* m_staticText_y
- wxTextCtrl* m_textCtrl_y

Obsługa wprowadzanej wartości amplitude:

- wxStaticText* m_staticText_amp
- wxTextCtrl* m_textCtrl_amp

Obsługa wprowadzanej wartości frequency:

- wxStaticText* m_staticText_freq
- wxTextCtrl* m_textCtrl_freq

Obsługa zatwierdzania lub cofania wprowadzonych zmian:

- wxStdDialogButtonSizer* m_sdbSizer_OK_CANCEL
- wxButton* m_sdbSizer_OK
- wxButton* m_sdbSizer_Cancel

Publiczne metody:

- MyDialog(…) - konstruuje obiekt klasy Mydialog i odpowiada za kontrolę poprawności parametrów
- ~MyDialog( ) – destruktor
- bool running() const - metoda kontrolująca flag added
- const auto getX() const - metoda zwracająca wartość pola x
- const auto getY() const - metoda zwracająca wartość pola y
- const auto getAmplitude() const - metoda zwracająca wartość pola amplitude(amplituda) –
- const auto getFrequency() const - metoda zwracająca wartość pola frequency(częstotliwość)

myVector – klasa reprezentująca wektor 4 wymiarowy

Prywatne pola:

- std::vector<double> data - nasz wektor

Konstruktor:

myVector(double x = 0.0, double y = 0.0, double z = 0.0) - konstruktor z wartościami domyślnymi

Publiczne metody:

- void set(double x, double y, double z) - ustawia parametry wektora
- double& operator[](unsigned index) - przeciążony operator [] obsługujący zmiany
- const double& operator[](unsigned index) const - przeciążony operator [] obsługujący odczytanie wartości
- myVector operator+(const myVector& obiect) const - przeciążony operator dodawania wektorów
- myVector operator-(const myVector& obiect) const - przeciążony operator odejmowania wektorów

myMatrix – klasa reprezentująca macierz 4 wymiarową

myMatrix() - konstruktor

- std::vector<double>& operator[](unsigned index) - przeciążony operator [] obsługujący zmiany
- std::vector<double> operator[](unsigned index) const - przeciążony operator [] obsługujący odczytanie wartości
- myMatrix operator*(const myMatrix& obiect) const - operator obsługujący mnożenia dwóch macierzy
- myVector operator*(const myVector& obiect) const - operator obsługujący mnożenia macierzy i wektora

Użyte funkcje:

- myMatrix setRotation(double alpha_x, double alpha_y, double alpha_z) - zwraca macierz po obrocie według wszystkich osi
- myMatrix setRotationX(double alpha) - zwraca macierz obrotu względem osi OX
- myMatrix setRotationY(double alpha) - zwraca macierz obrotu względem osi OY
- myMatrix setRotationZ(double alpha) - zwraca macierz obrotu względem osi OZ
- myMatrix setTranslation(double width, double height) - zwraca macierz translacji
- myMatrix setScale(double scale) - zwraca macierz skali
- myMatrix setPerspective() - zwraca macierz perspektywy