JOYSTICK USB

Labolatorium nr 4

Maciej Bronikowski 248838
Michał Droń 248832

1 WSTĘP

Joystick USB jest urządzeniem wejścia komputera, służącym do manipulacji obiektami wyświetlanymi na ekranie. Standardowa wersja joysticka składa się z umieszczonego na podstawce drążka, którym można sterować ruchem w osiach X, Y (czasem spotykane są kontrolery umożliwiające sterowanie również w osi Z, poprzez przekręcanie drążka) oraz umieszczonych na drążku i podstawce przycisków, które uruchamiają przypisane im akcje. Sygnały wejściowe kontrolera można podzielić na dwa typy. Przyciski mogą znajdować się jedynie w dwóch stanach: aktywny lub nieaktywny. Jednak stan samego drążka (oraz często dodatkowych elementów jak mały drążek umieszczony u góry manipulatora, lub suwak) podawany jest jako wartość ze zdefiniowanego przedziału. Możliwe jest to dzięki zastosowaniu potencjometru, którego wartość rezystancji zmieniana jest podczas manipulacji drążkiem i przeliczana na kąta nachylenia.

2 ZADANIA DO WYKONANIA

1. Napisać program, odczytujący nazwę zainstalowanego joysticka, wykorzystując w tym celu API DirectInput
2. Napisać program ilustrujący działanie joysticka: stwierdzający naciśnięcie myszy, zmianę położenia drążka (w przestrzeni 2D) oraz suwaka.
3. Napisać program zastępujący działanie myszy. Program ma umożliwiać sterowanie kursorem za pomocą joysticka oraz obsługę przycisków fire jako kliknięć myszy
4. Napisać program realizujący prosty edytor graficzny - rysowanie przy pomocy Joysticka

3 REALIZACJA ZADANIA

Do realizacji zadania z powodu braku joysticka wykorzystany został kontroler Xbox. Mimo zupełnie innego wyglądu, realizacja programu przebiega w podobny sposób, jak z wykorzystaniem joysticka. Kontroler Xbox posiada cztery analogowe elementy: dwa drążki (L, R), oraz dwa spusty (LT, RT) wysyłające wartość ich nachylenia. Dodatkowo, oprócz normalnych przycisków (A, B, X, Y, LSB, RSB, LB, RB, start, back), których odczytanie wartości odbywa się jedynie poprzez sprawdzenie wartości ich stanów, na kontrolerze znajdują się również strzałki, których odczytywany stan jest wartością kątu obrotu. Oznacza to, że jeżeli wciśnięta jest jedynie górna strzałka, kąt obrotu będzie równy 0, jeżeli wciśnięta jest dolna, kąt obrotu będzie równy 180, a jeżeli wciśnięta jest górna oraz prawa, kąt obrotu będzie równy 45.

Wartości maksymalne elementów analogowych jak drążki definiowane są przez sprzęt, jak również można je (nie zawsze) ustawić podczas połączenia z kontrolerem. Wartości kątów zgłaszanych podczas używania strzałek jak wynika z dokumentacji Microsoftu są wartościami kątów pomnożonych przez 1000. Oznacza to że wciśnięcie prawej strzałki zgłaszane będzie przez wartość 90000.

Program zajmujący się obsługą zdarzeń kontrolera został napisany w języku C#, z użyciem biblioteki SharpDX oraz jej składowej biblioteki SharpDX.DirectInput. Interfejs użytkownika napisany został z użyciem interfejsu Windows Forms dostarczanego w ramach Microsoft .NET Framework.

3.1 IMPLEMENTACJA

Program został podzielony na cztery klasy odpowiadające za funkcje związane z aplikacją:

- JoystickForm
- JoystickControler
- MouseControler
- DrawControler

3.1.1 JoystickForm

Klasa odpowiedzialna jest za tworzenie graficznego interfejsu użytkownika, oraz obsługę zdarzeń z nim związanych. Oprócz instancji pozostałych z przedstawionych klas oraz możliwości wywoływania poszczególnych ich funkcjonalności w zależności od akcji użytkownika, posiada również instancje klasy BackgroundWorker, która poprzez uruchomienie oddzielnego wątku umożliwia aplikacji pobieranie informacji z kontrolera, bez potrzeby blokowania wątku graficznego interfejsu.

3.1.2 JoystickControler

Klasa ta odpowiedzialna jest za wszystkie funkcjonalności związane z obsługą kontrolera. Pozwala na:

- Pobranie listy dostępnych urządzeń wraz z ich nazwami (metoda loadDevicesList)
- Wybranie oraz przechowywanie id wybranego urządzenia (metoda ChooseJoystick)
- Pobranie nazwy urządzenia (metoda GetName)
- Pobranie informacji na temat analogowych elementów kontrolera (metody GetRotationStick oraz GetPositionStick)
- Pobranie informacji na temat stanu wciśniętych przycisków (metoda GetButtonsPressed)
- Pobranie informacji o tym, czy stan kontrolera uległ zmianie (metoda AreNewEvents)

Podczas inicjalizacji połączenia z kontrolerem, zakresy elementów analogowych ustawiane są jako wartość double z zakresu [-1, 1].

W punkcie 3 przedstawione były cztery analogowe elementy kontrolera Xbox, jednak tylko dwie metody zajmują się ich sczytywaniem. Spowodowane jest to tym, że stan spustów LT oraz RT zgłaszany jest jako składowa Z pozycji lewego drążka. Jeżeli lewy tylko spust jest wciśnięty do końca, składowa Z będzie równa -1, jeżeli tylko prawy spust jest wciśnięty do końca, składowa Z będzie równa 1. Jeżeli oba spusty są wciśnięte do końca, składowa Z będzie równa 0. Oznacza to, że jeżeli oba spusty są wciśnięte jednocześnie, wartością podaną w składowej Z będzie suma wartości LR oraz RT.

3.1.3 MouseControler

Klasa odpowiada za możliwości związane z poruszaniem kursorem za pomocą kontrolera, oraz symulowaniem wciśnięcia lewego oraz prawego przycisku myszy. Sterowanie myszą odbywa się poprzez ruch lewego drążka kontrolera, kliknięcie lewego przycisku myszy realizowane jest przez wciśnięcie przynajmniej do połowy spustu RT, a kliknięcie prawego przycisków poprzez wciśnięcie przynajmniej do połowy spustu LT.

Obsługa myszy została zrealizowana z pomocą biblioteki instrukcji Windows user32.dll. Pozwala ona nie tylko na poruszanie kursorem (za pomocą funkcji SetCursorPos), ale również na wywołanie zdarzeń myszy poprzez podanie odpowiednich kodów przycisków (za pomocą funkcji mouse_event). Kolejne kody zdarzeń obsługiwanych przez funkcje są:

- LeftDown = 0x00000002,
- LeftUp = 0x00000004,
- MiddleDown = 0x00000020,
- MiddleUp = 0x00000040,
- Move = 0x00000001,
- Absolute = 0x00008000,
- RightDown = 0x00000008,
- RightUp = 0x00000010

Trzecią zaimportowaną funkcją z biblioteki jest GetCursorPos, która pozwala na pobranie aktualnej pozycji myszy.

Metodami dostarczanymi przez klasę MouseControler są:

- enable – włączenie obsługi sterowania sterowania za pomocą kontrolera
- disable – wyłączenie obsługi sterowania myszą za pomocą kontrolera
- IsEnabled – zwraca informację o tym, czy sterowanie myszą jest aktualnie włączone
- MouseInput – Konwertowanie zdarzeń kontrolera na aktualną pozycję myszy.

3.1.4 DrawControler

Klasa odpowiada za możliwość rysowania za pomocą kontrolera, oraz zapis rysunku do pliku w formacie .bnp. Poruszanie się kursorem po polu wyznaczonym do rysowania odbywa się na podstawie podobnych założeń co w klasie MouseControler, jednak w tym wypadku nie potrzebna jest biblioteka user32.dll, ponieważ kursor jest jedynie kwadratem nakładanym dodatkowo na aktualny rysunek, który pokazuje aktualną pozycję rysowania oraz rozmiar pędzla.

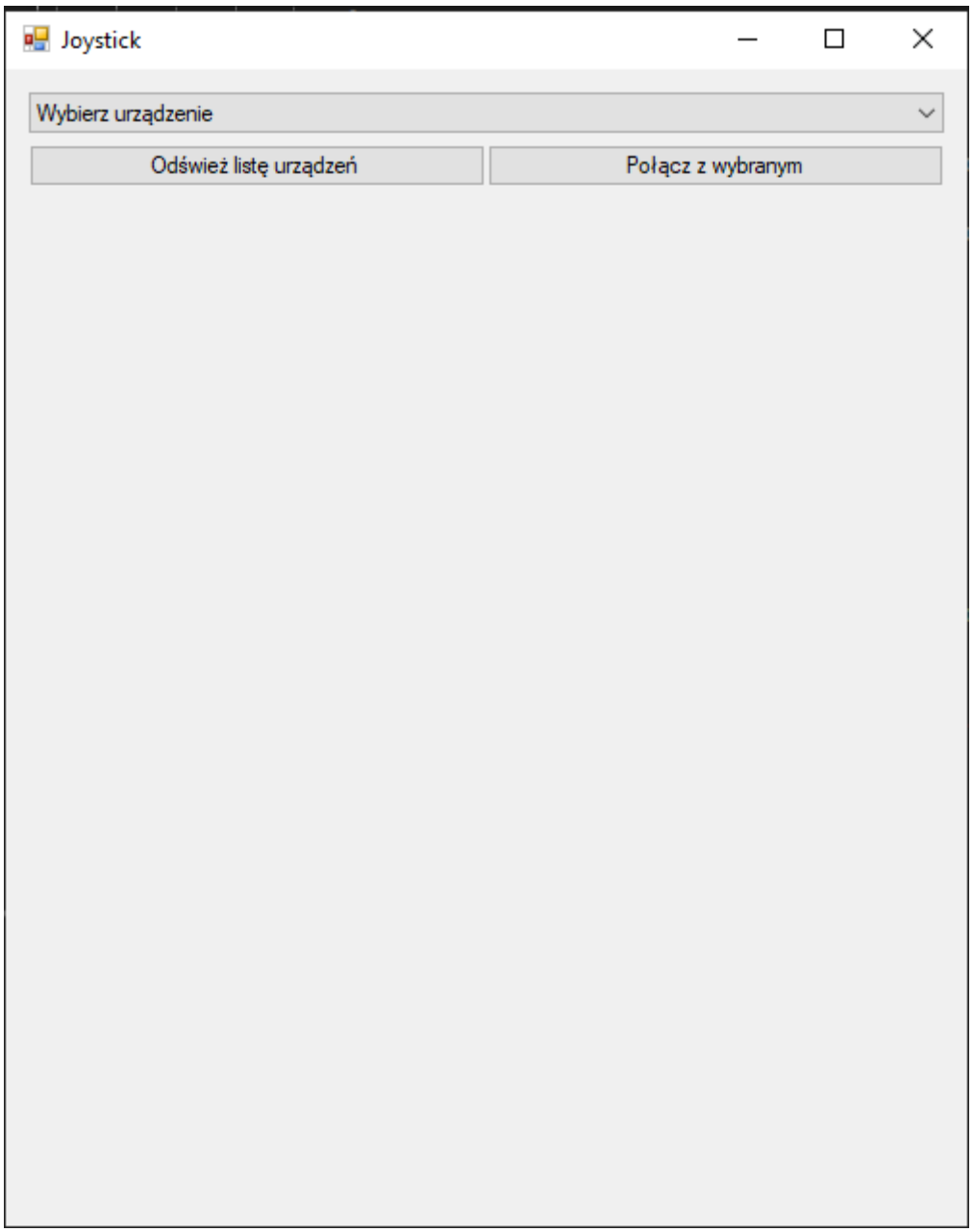
Rysowanie odbywa się z pomocą dwóch instancji klasy Bitmap, z których pierwsza przechowuje rysunek, a druga przedstawia pozycję kursora, poprzez nałożenie na rysunek kwadratu z pozycją kursora. Dodatkowo użyta została instancja klasy System.Windows.Forms.Timer, która dostarcza funkcjonalność wywoływania metody po zadanej liczbie milisekund. W przypadku obecnie opisywanej klasy pozwala to na zmianę raz na sekundę koloru kursora. Jeżeli kursor posiadałby tylko jeden kolor, mógłby być niewidoczny na pomalowanym w odpowiednim kolorze obszarze.

Klasa dostarcza takie metody jak:

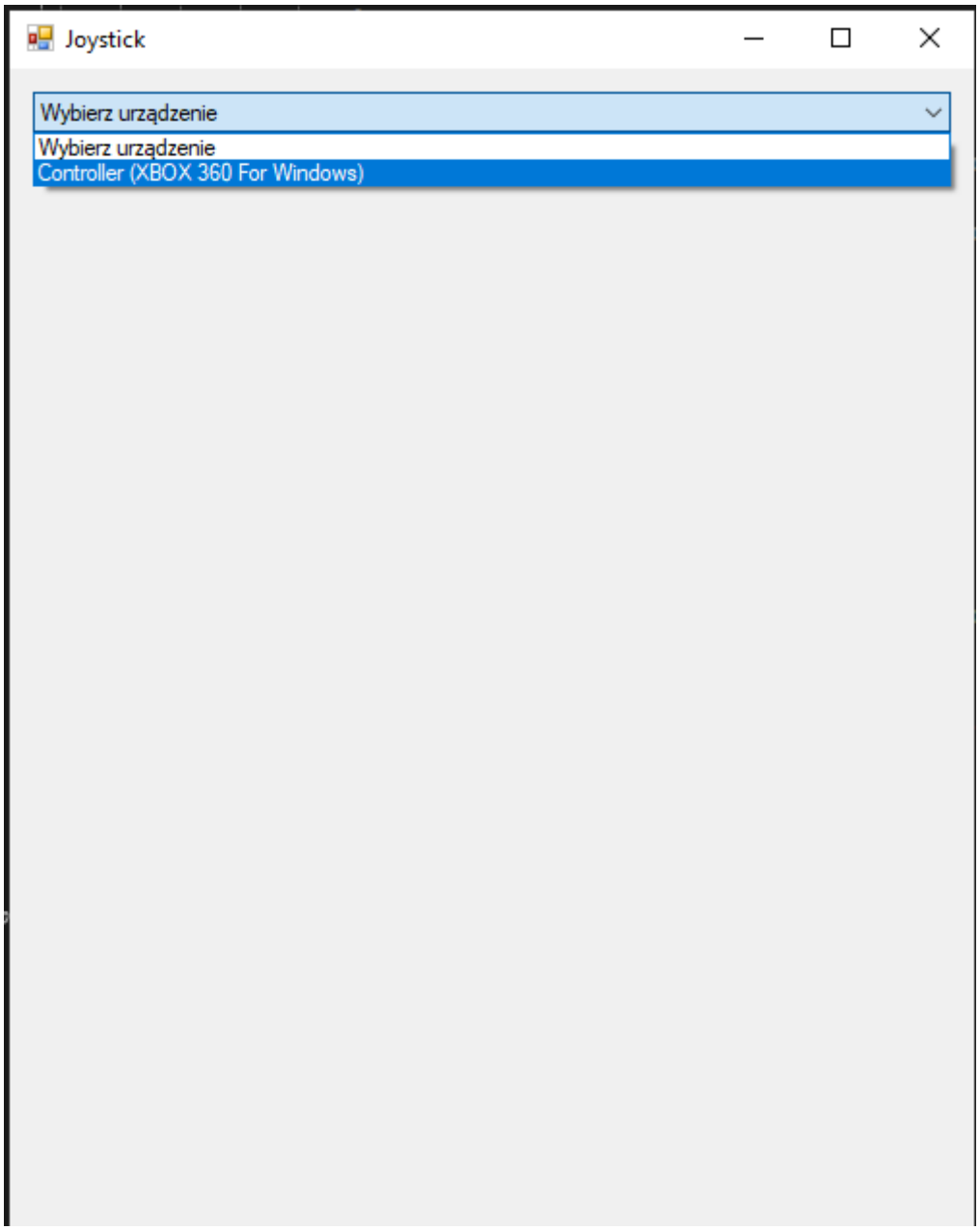
- BrushColor – ustawienie koloru pędzla
- DrawInput – Konwertowanie zdarzeń kontrolera na aktualną pozycję pędzla
- tick_function – Zmiana informacji o tym w jakim stanie znajduje się kursor. Zapewnia to funkcjonalność mrugania kursora.
- SetCursor - Zapewnia funkcjonalność kopiowania aktualnie narysowanego obrazu, oraz wstawianie w odpowiednie miejsce kursora (pędzla), aby pokazać w którym miejscu odbywać się będzie rysowanie.
- Clear – Pozwala na wyczyszczenie obrazu kolorem białym
- Save – Pozwala na zapis namalowanego obrazu do pliku

3.2 ZRZUTY EKRANU APLIKACJI

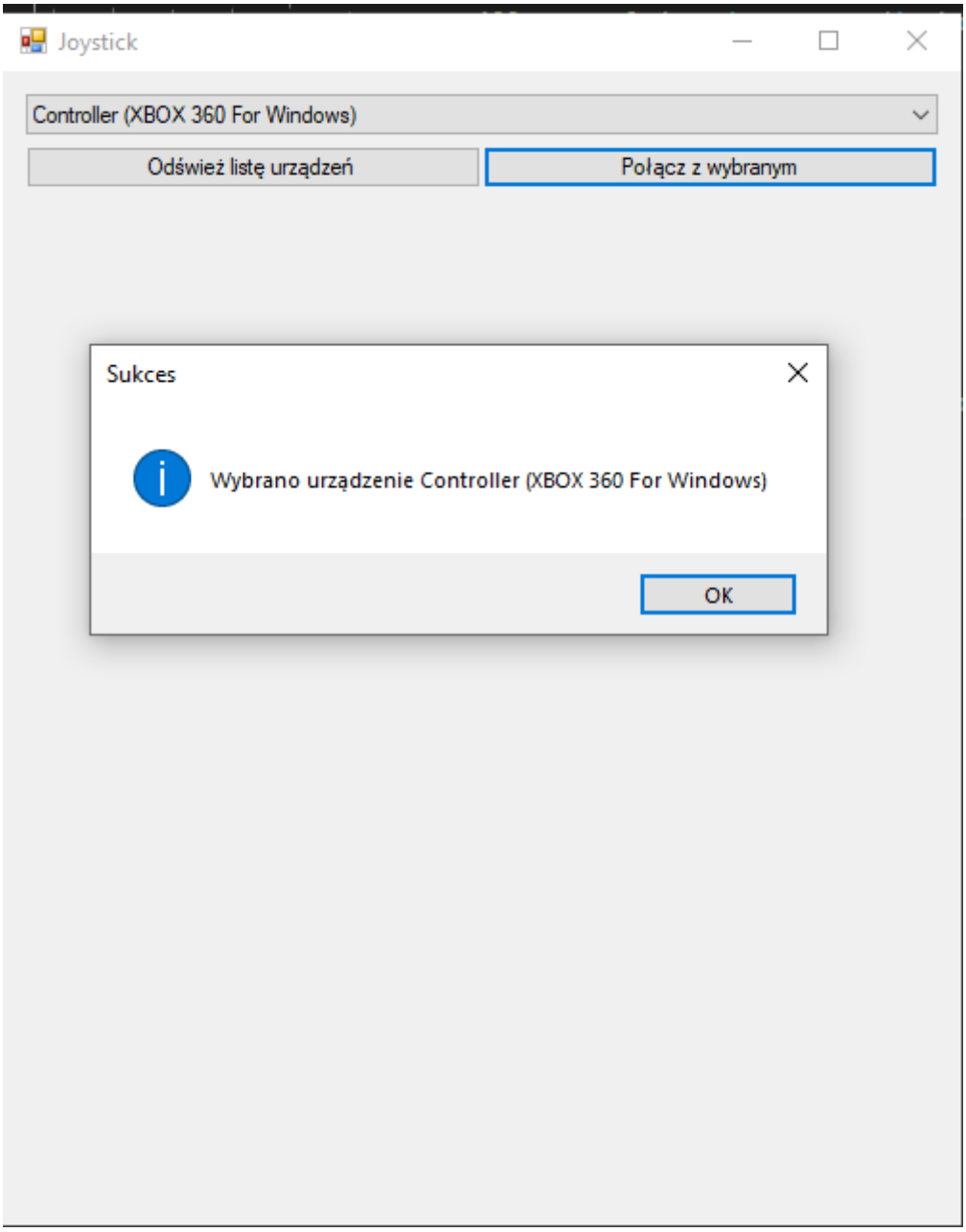
3.2.1 Pierwsze uruchomienie aplikacji



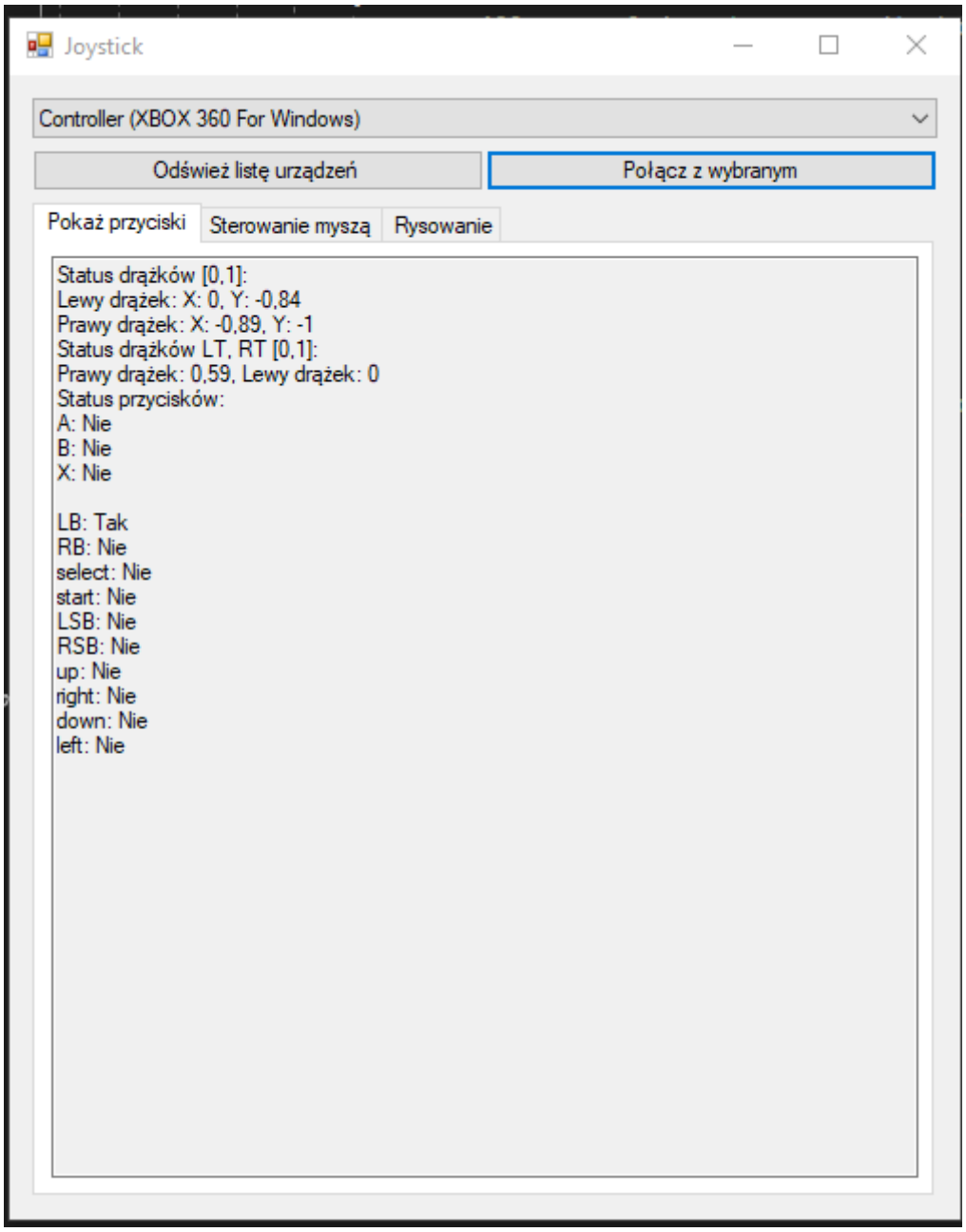
3.2.2 Wybór urządzenia



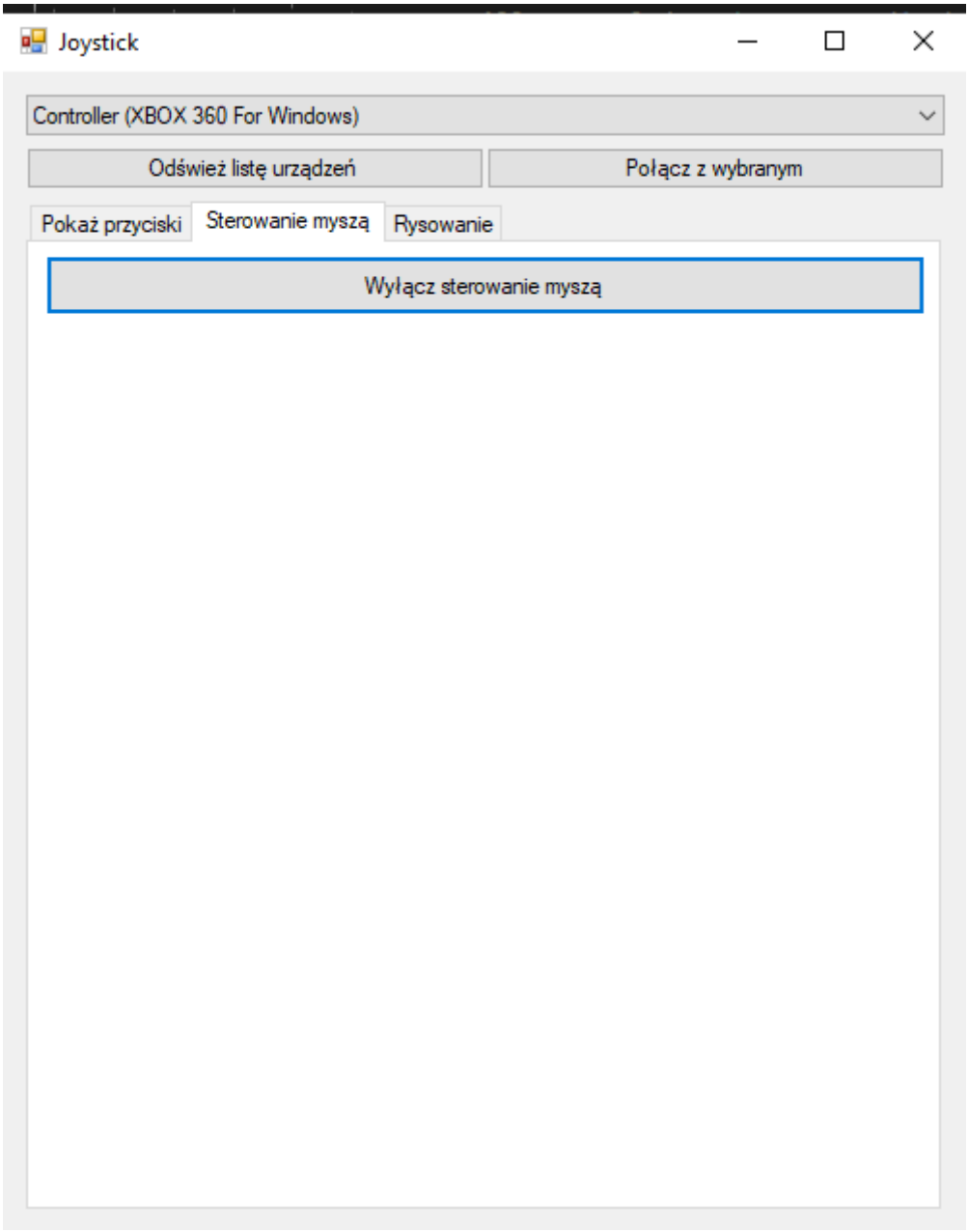
3.2.3 Informacja o połączeniu z urządzeniem oraz jego nazwie



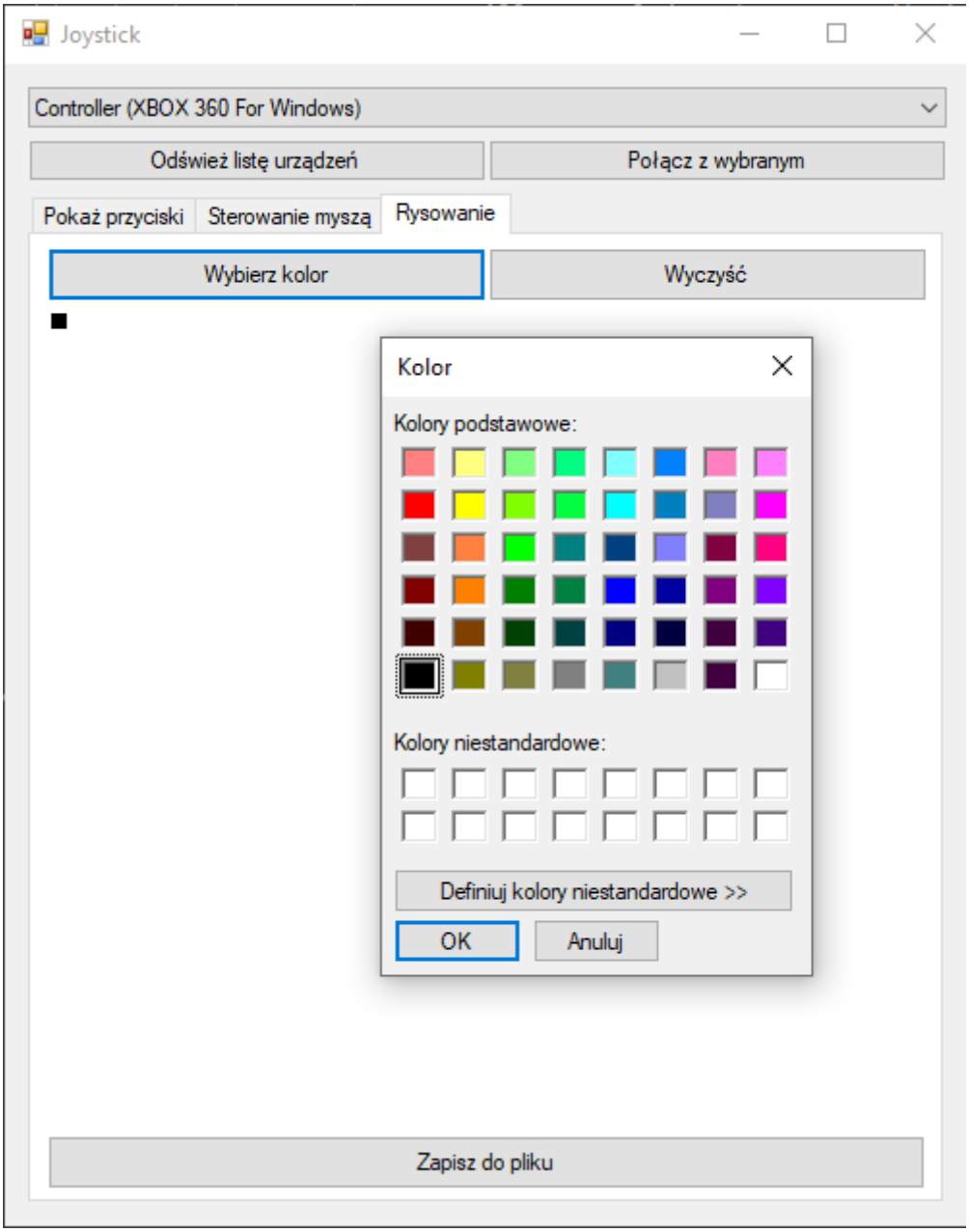
3.2.4 Lista przycisków i drążków, oraz ich aktualne stany



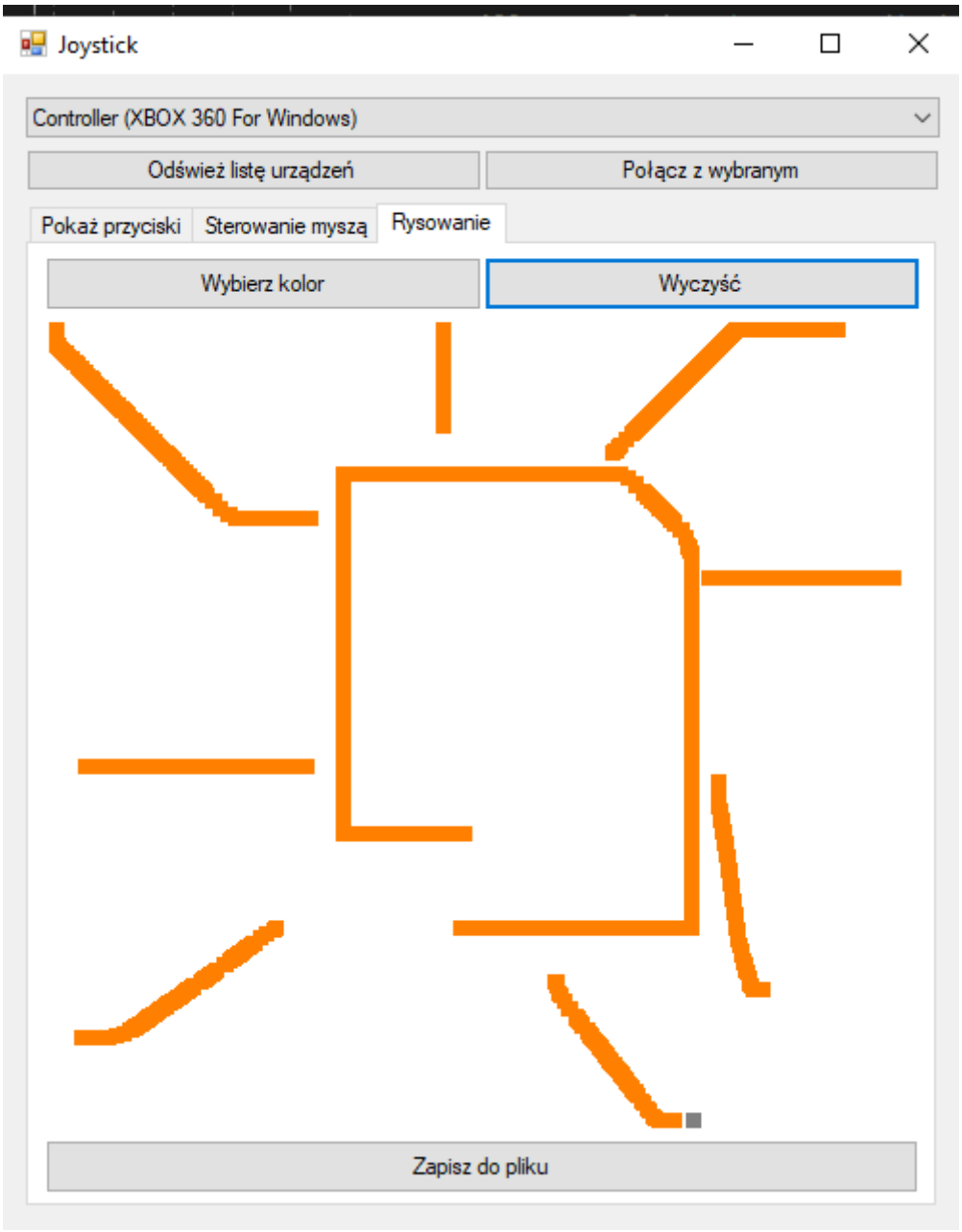
3.2.5 Obsługa myszy za pomocą kontrolera



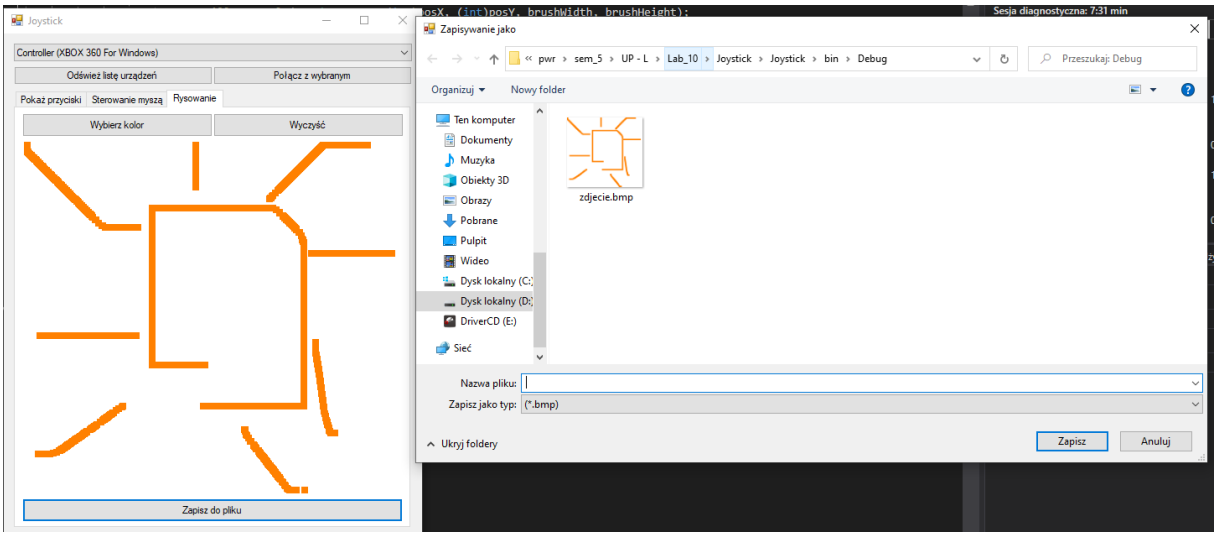
3.2.6 Wybór koloru pędzla



3.2.7 Rysowanie za pomocą kontrolera



3.2.8 Zapis namalowanego obrazu do pliku



4 PODSUMOWANIE

Wszystkie z wymienionych zadań udało się zrealizować bez żadnego problemu. Podczas realizacji zadania poznaliśmy zasady pobierania i obsługiwania zdarzeń związanych z kontrolerem i joystickiem. Dodatkowo w ramach punktu obsługi myszy za pomocą joysticka poznaliśmy funkcje biblioteki user32.dll związane z obsługiwaniem zdarzeń kursora.

Pełny kod aplikacji znajduje się na zdalnym repozytorium Github pod adresem:
https://github.com/JeLLek1/UrzadzeniaPeryferyjne/tree/main/Lab_10/Joystick