Zadanie 6 - projekt własny

Dominik Kuczkowski 180235
Izabela Załęska 180322

06.06.2022

1 Założenia

- Aplikacja będzie służyć do prostej wizualicaji 2D robota przemysłowego
- Aplikacja posiadać będzie interfejs graficzny
- Możliwe będzie zapisywanie ruchów robota
- Robot wyposażony będzie w magnes, który umożliwi mu przenoszenie klocków z miejsca na miejsce
- Możliwe będzie sterowanie robotem za pomocą klawiatury
- Program napisany zostanie w języku C++ z wykorzystaniem biblioteki SFML

2 Opis programu

2.1 Interfejs graficzny

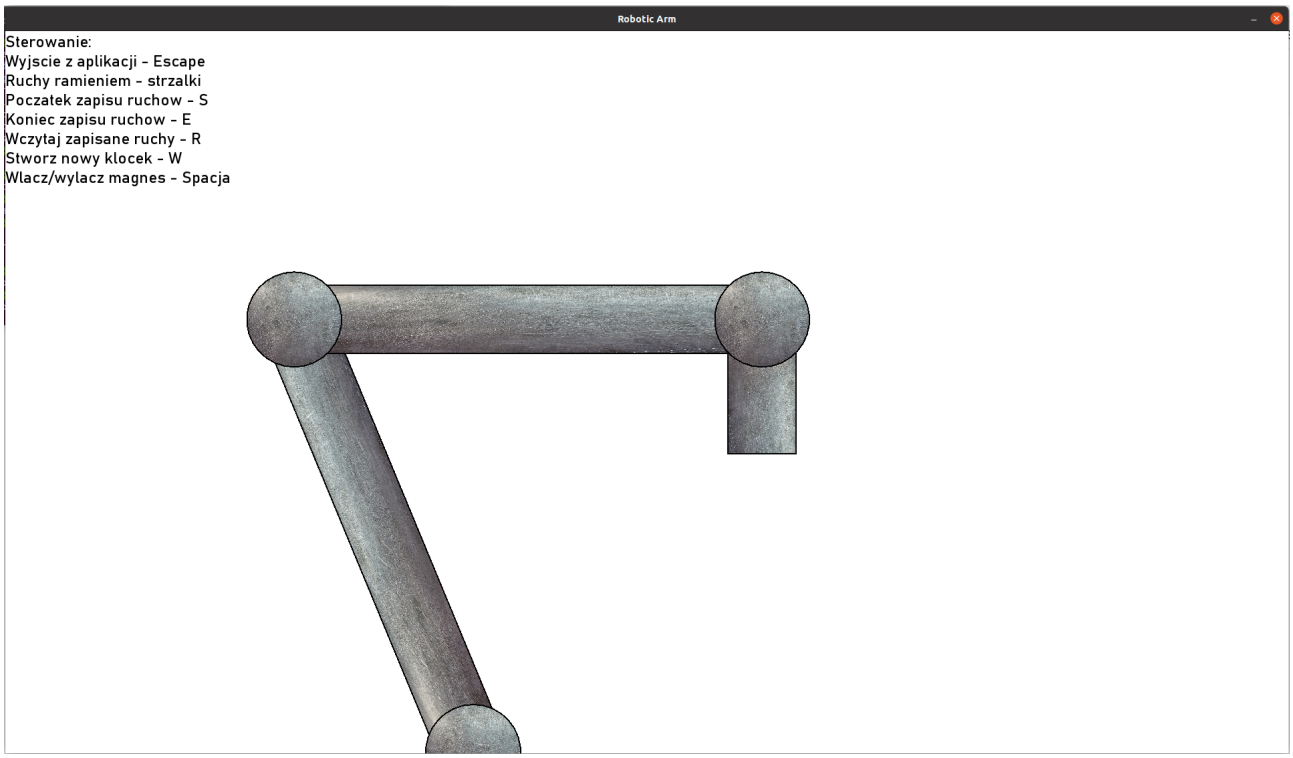

Figure 1: Interfejs graficzny

Na rys. 1 przedstawiony został interfejs graficzny programu. Przedstawia on prostą wizualizację ramienia robotycznego o trzech jointach. W oknie programu wyświetlany jest sposób sterowania robotem. Wykorzystuje się do tego strzałki. Klawisz W umożliwia stworzenie klocka, który pojawia się u dołu ekranu. Wciśnięcie spacji włącza lub wyłącza magnes na końcu ostatniego z fragmentów robota. Umożliwia to przenoszenie klocków. Klawisze S, E i R służą do zapisu i odtwarzania ruchów robota.

2.2 Implementacja

Punktem startowym programu jest plik main.cpp, w którym odbywa się stworzenie instancji klasy robot oraz uruchamiana jest pętla programu. W pętli wywoływane są metody update() i render() z klasy Robot. Metoda update odpowiada za wywoływanie wewnętrznych metod związanych z obsługą klawiatury i zmianą stanu programu. Metoda render wywołuje funkcje związane z wyświetlaniem obrazu. W programie do odczytywania klawiszy z klawiatury wykorzystywane są Eventy udostępniane przez bibliotekę SFML. Na rys. 2 przedstawiona jest funkcja sprawdzająca wciśnięcie klawiszy odpowiedzialnych za zamykanie okna, tworzenie klocków oraz włączanie magnesu.

```cpp
void Robot::pollEvents()
{
    /*
    Handled keyboard inputs or events:
    -close event,
    -close on escape,
    -create new box,
    -magnet on and off
    */
    while (this->window->pollEvent(this->event))
    {
        switch (this->event.type)
        {
        case sf::Event::Closed:
            this->window->close();
            break;
        case sf::Event::KeyPressed:
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))
                this->window->close();
            break;
        case sf::Event::KeyReleased:
            if (this->event.key.code == sf::Keyboard::W)
            {
                this->createnewbox();
            }
            else if (this->event.key.code == sf::Keyboard::Space)
            {
                this->magnetonoff = !this->magnetonoff;
            }
            break;

        }
    }
}
```

Figure 2: Funkcja czytająca klawisze

Jak widać biblioteka udostępnia przyjemne API do sprawdzania zarówno czy klawisz został wciśnięty czy puszczony. Po sprawdzeniu klawiszy w programie wywoływana jest funkcja poruszająca ramieniem, która reaguje na eventy od strzałek. Wyliczanie nowych pozycji poszczególnych elementów robota odbywa się za pomocą funkcji trygonometrycznych, co zostało przedstawione na rys. 3.

```
void Robot::moveArm()
{
    //Updating position variables
    this->arm2position.x = this->arm1position.x + this->armsize.x * std::sinf(this->arm1angle * (this->pi / 180.f));
    this->arm2position.y = this->arm1position.y - this->armsize.x * std::cosf(this->arm1angle * (this->pi / 180.f));

    this->gripperposition.x = this->arm2position.x + this->armsize.x * std::sinf((this->arm2angle) * (this->pi / 180.f));
    this->gripperposition.y = this->arm2position.y - this->armsize.x * std::cosf((this->arm2angle) * (this->pi / 180.f));

    this->gripperendposition.x = this->gripperposition.x;
    this->gripperendposition.y = this->gripperposition.y + this->grippersize.x;

    //Updating elements positions
    this->arm2.setPosition(arm2position);
    this->gripper.setPosition(gripperposition);
    this->circle2.setPosition(this->arm2position);
    this->circle3.setPosition(this->gripperposition);
}
```

Figure 3: Funkcja obliczająca nowe położenia przegubów robota

W programie zaimplementowano opcję zapisu ruchów robota. Odbywa się ona z wykorzystaniem strutktury danych z biblioteki standardowej - Queue. Na rys. 4 pokazany jest moment zapisu do kolejki jednego z ruchów

```
//Saving moves
if (this->isbeingsaved)
{
    this->queue.push(sf::Vector2f(2.f, -this->movementAngle));
}
```

Figure 4: Dodawanie do kolejki jednego z ruchów robota

robota. Zapisywany jest numer przegubu, który wykonuje ruch oraz kąt o jaki należy ten przegub przesunąć. Taki sposób zapisu pozwala w łatwy sposób wczytywać ruchy.

3 Omówienie wyników

Program spełnia swoje założenia co do symulowania prostych ruchów robota na płaszczyźnie. Wadą projektu jest bardzo prosta symulacja, która pozwala na łapanie tylko kwadratowych przedmiotów. Zaletą projektu jest dobrze napisany kod w C++, który działa wydajnie i mógłby zostać z powodzeniem rozszerzony o nowe funkcjonalności.