Rafał CHODZIDŁO, Krzysztof JASKOT
Politechnika Śląska

WIZUALIZACJA PRACY ROBOTA ADEPT W ŚRODOWISKU UNITY 3D

Streszczenie. W artykule opisano sposób wizualizacji pracy robota AdeptSix 300. Przedstawiono wykorzystane narzędzia użyte do stworzenia modelu robota - Blender oraz jego wizualizacji i interakcji z otoczeniem w środowisku Unity 3D.

VISUALIZATION OF THE ROBOT IN THE UNITY 3D ENVIRONMENT

Summary. The article describes how to visualize the work of the AdeptSix 300 robot. It presents the tools used to create the robot model - Blender and its visualization and interaction with the environment in the Unity 3D environment.

1. Wstęp

W dzisiejszych czasach wizualizacja wpływa na sposób prowadzenia badań naukowych, jest rutynowo wykorzystywana w dyscyplinach technicznych i medycynie, służy celom dydaktycznym, a także bywa pojmowana, jako środek wyrazu artystycznego [3, 9].

Umiejętność wizualizacji, czyli tworzenia obrazów w głowie, to umiejętność niezwykle ważna dla każdego, kto interesuje się rozwojem osobistym. Tworząc wizualizację nastawiamy swój mózg na działanie w jakimś określonym przez nas kierunku. Jest to najważniejsze narzędzie pracy z naszym umysłem. Jednak, aby narzędzie to było skuteczne, trzeba z niego prawidłowo korzystać. Szybki dostęp do potrzebnych informacji oraz ich trafne przedstawianie wymaga interakcji użytkownika z narzędziem - przede wszystkim komputerem. Informatycy starają się uczynić ten proces jak najbardziej efektywnym i nieabsorbującym. Aby mówić o interaktywnej wizualizacji (inaczej: wizualizacji, kiedy użytkownik bezpośrednio wpływa na przeprowadzaną symulację) musi istnieć możliwość wprowadzania danych za pomocą zewnętrznego kontrolera, a wyniki powinny być aktualizowane w czasie rzeczywistym.

Przykładem takiej interakcji są symulatory rzeczywistości wirtualnej (ang. virtual reality, VR). Użytkownik zostaje zanurzony w cyfrowym świecie, na który może wpływać, korzystać z określonych urządzeń. Inną formą realizacji powyższych postulatów są narzędzia do wizualizacji grupowej (ang. collaborative visualization). Na przedstawiany obraz ma wpływ cały zespół, który na bieżąco wymienia się uwagami.

Wizualizacja, tak jak techniki komputerowe w ogólności, upowszechniła się w wielu dziedzinach nauki. Każdy sposób interaktywnej, zmysłowej reprezentacji surowych danych wejściowych, która ma pomóc w zrozumieniu istoty problemu, tworzeniu hipotez oraz trafnym wyciąganiu wniosków, podlega pod jej definicję. Wizualizacja

produktów jest zbiorem technik, które umożliwiają analizę dokumentacji, przeglądanie planów technicznych oraz podgląd i manipulacje trójwymiarowym modelem produktu przed wyprodukowaniem jego prototypu. Obecnie postrzega się je, jako zasadniczy element cyklu zarządzania życiem produktu (ang. Product Lifecycle Management, PLM). Realistyczne odwzorowanie właściwości obiektu ułatwia kontrolę nad projektem; czyni go też bardziej dostępnym dla wszystkich członków zespołu: od projektantów do specjalistów sprzedaży i marketingu.

Narzędzia, dzięki którym możliwa jest wizualizacja obiektów, pozwalają na wygodne i szybkie tworzenie i poprawianie układu jeszcze przed jego wytworzeniem. Rozwiązanie takie pozwala na dużą oszczędność czasową oraz finansową, co jest istotną kwestią w obecnym przemyśle.

Wizualizacja jest w dzisiejszych czasach mocno rozwijanym zagadnieniem opisującym technologie wspomagające projektowanie oraz wytwarzanie fizycznego modelu przedmiotu. Wykorzystując narzędzia znane pod wspólnym pojęciem Projektowanie Wspomagane Komputerowo (ang. Computer Aided Design) można zaprojektować, a następnie wyprodukować dowolny przedmiot. Technologie te często są kojarzone z narzędziami przeznaczonymi do tworzenia trójwymiarowych modeli przedmiotów. Przy ich produkcji niejednokrotnie wykorzystywane są drukarki 3D oraz wszelkie inne narzędzia, które w bezpośredni sposób pozwalają na przeniesienie cyfrowego projektu do rzeczywistości.

2. Opis problemu

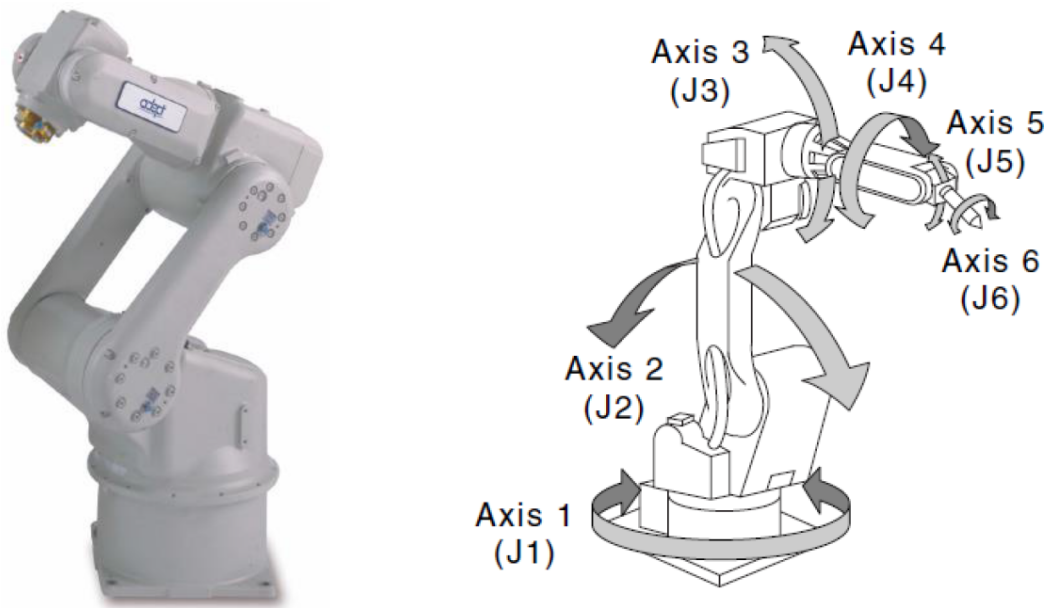
Ideą projektu było stworzenie wizualizacji robota AdeptSix 300 [1]. Ma on pozwalać na szybkie i wygodne przedstawienie pracy robota bez konieczności posiadania rzeczywistego urządzenia. Rozwiązanie to posiada możliwość bezpośredniej zmiany wartości parametrów członów robota w trybie online w czasie rzeczywistym.

Celem projektu było stworzenie modelu 3D robota Adept, a następnie oprogramowanie go w środowisku Unity 3D. Zrealizowane oprogramowanie działa na komputerach klasy PC. Środowisko Unity daje także możliwość publikacji aplikacji na inne platformy np. iOS, Android czy też w wersji, którą można umieścić na stronie internetowej.

Do realizacji wykorzystano dwa środowiska projektowe: Blender 3.6 LTS [7, 10], w którym został stworzony model robota oraz Unity 3D [6, 13], w którym robotowi nadano cechy fizyczne, możliwość sterowania oraz stworzono otoczenie robota. Do sterowania robotem używa się klawiatury oraz specjalnej nawigacji sterowanej myszką. Efektem pracy jest aplikacja pozwalająca na łatwe zmiany położenia ramion manipulatora. Odbywa się to w oparciu o podstawowy zestaw wejść komputera, czyli klawiatura i myszka lub ewentualnie joystick. Na panelu informacyjnym są wyświetlane dane o prędkości obrotu ramienia manipulatora lub jego aktualny kąt. Aplikacja stworzona w ramach niniejszego projektu, stanowi tanią i prostą w obsłudze alternatywę dla badania manipulatorów.

Wygląd modelowanego robota AdeptSix 300 został przedstawiony na rysunku 1. Jest to podstawowy robot o sześciu członach obrotowych i maksymalnym udźwigu 3kg. Jest on bardzo dobrym przykładem robota, który można wykorzystać w ramach laboratorium z podstaw robotyki. Opracowane narzędzie do wizualizacji pozwala na za-

poznanie się z działaniem robota bez fizycznego dostępu do niego co pozwala na lepsze przygotowanie się do jego obsługi i programowania.



Rys. 1. Robot AdeptSix 300 oraz opis jego osi [1]

3. Wykorzystane narzędzia

W celu realizacji zadania wykorzystano dwa środowiska do modelowania i symulacji Blender oraz Unity 3D. Oba dostępne są całkowicie za darmo w celach niekomercyjnych.

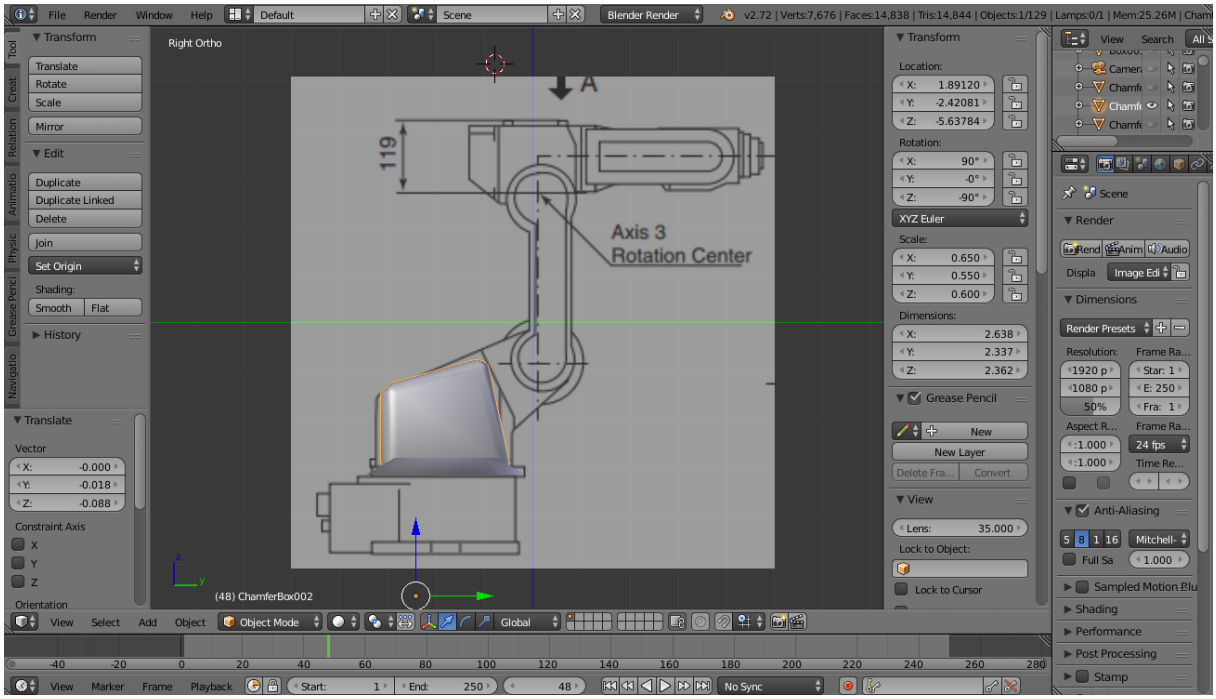
3.1. Blender

Blender – oprogramowanie do modelowania i renderowania obrazów oraz animacji trójwymiarowych o niekonwencjonalnym interfejsie użytkownika. Umożliwia także tworzenie prezentacji interaktywnych (np. gier) na własnym silniku graficznym (istnieje osobny program pozwalający uruchamiać takie prezentacje). Na rysunku 2 przedstawiono okno edycji programu Blender wraz z modelowanym robotem Adept [2, 5]. Natomiast na rysunku 3 możemy zobaczyć gotowy model 3D robota Adept stworzony wykorzystując możliwości edycyjne środowiska Blender.

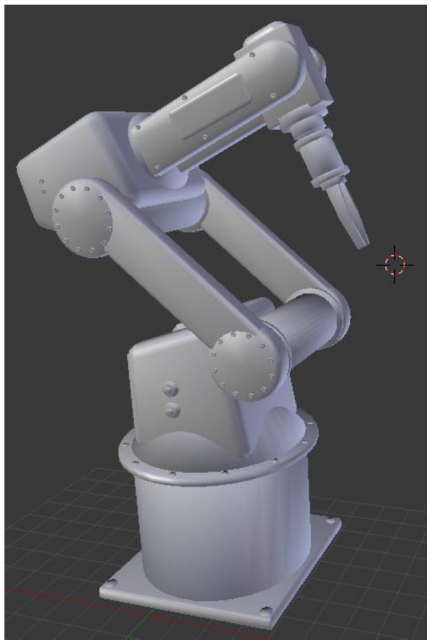
3.2. Unity 3D

Unity – zintegrowane środowisko do tworzenia gier trójwymiarowych lub innych materiałów interaktywnych takich jak wizualizacje czy animacje trójwymiarowe. Środowisko działa natywnie na platformie Microsoft Windows, Linux oraz MacOS, a gry wyprodukowane na tym silniku mogą działać na takich platformach jak Windows, Linux, MacOS, Xbox, PlayStation, Nintendo, iPad, iPhone, Android.

Silniki gier takich jak Unity są znanymi, docenianymi, a przede wszystkim potężnymi narzędziami wspierającymi tworzenie gier. Środowisko Unity jest jednym z najczęściej używanych oraz najbardziej cenionych pakietów pozwalających na projektowanie gier komputerowych. Może być ono wykorzystywane przez szerokie spektrum użyt-



Rys. 2. Ekran edycji Blender'a razem z modelowanym robotem



Rys. 3. Model 3D robota AdeptSix 300 utworzony przy użyciu narzędzia Blender

kowników, poczynając od hobbystów, a kończąc na dużych firmach. Pozwala tworzyć gry oraz interaktywne aplikacje dla przeglądarek internetowych, komputerów stacjonarnych, urządzeń przenośnych oraz konsol. Dzięki intuicyjnemu i prostemu w obsłudze zestawowi narzędzi środowiska unity oraz licznym publikacjom łatwo, jak nigdy dotąd, można stać się twórcą gier komputerowych [2, 5].

Skrypty należą do komponentów środowiska Unity 3D. Są uważane za kluczowy składnik wykorzystywany w procesie produkcji gier. Najpopularniejsze języki to C# oraz JavaScript. Unity oferuje możliwość użycia języka Boo pochodzącego od ję-

zyka Python.

Podczas używania systemu Unity nie jest wymagane zrozumienie, w jaki sposób został zakodowany jego własny silnik czy też jak można go zmodyfikować. Prostota użycia skryptów w środowisku Unity polega na tym, iż każdy z nich będzie zrozumiany dla kogoś, kto posiada podstawy z danego języka. Każda stworzona klasa dziedziczy po klasie MonoBehaviour [14].

Obecnie szczególnie cenione są języki programowania, które pozwalają na błyskawiczne osiągnięcie oczekiwanych efektów. Dodatkowo absolutnie niezbędne jest zachowanie możliwości uruchamiania raz napisanej aplikacji na różnych platformach bez konieczności jej przepisywania. C# to nowoczesny język, który zdobył uznanie programistów na całym świecie [8]. Obsługa zdarzeń czyli sterowanie poszczególnymi członami manipulatora została napisana właśnie z wykorzystaniem języka C#. Korzystając z możliwości jakie oferuje środowisko Unity można do samego modelu robota dodać jego otoczenie takie jak przeszkody oraz obiekty manipulacji co zostało przedstawione na rysunku 4. Dzięki wykorzystaniu języka C# można dodać elementy fizyki, która w znacznym stopniu podniesie jakość całej symulacji [4].

Rys. 4. Model robota AdeptSix 300 umieszczony w środowisku Unity 3D przygotowany do interakcji z otoczeniem

Implementowanie fizyki podczas tworzenia symulacji w Unity 3D obejmuje wykorzystanie wbudowanego silnika fizyki i komponentów do symulacji realistycznych interakcji między obiektami. Oto przegląd kroków związanych z implementacją fizyki na potrzeby wizualizacji pracy robota. Podstawowym elementem implementacji fizyki jest komponent *Rigidbody* dla obiektów, które wymagają interakcji fizycznych. Komponent ten umożliwia oddziaływanie na obiekty sił, grawitacji i kolizji. Następnym krokiem jest dodanie komponentu *Collider* do obiektów, aby zdefiniować ich kształt na potrzeby wykrywania kolizji. Wykorzystując ten komponent możemy wykrywać kolizje i reagować na nie. Środowisko Unity udostępnia komponenty połączeń, które umożliwiają tworzenie połączeń między obiektami w naszym przypadku między poszczególnymi członami manipulatora. Połączenia mogą tworzyć realistyczne interakcje między obiektami (człony manipulatora). Na listingu 1 znajduje się przykładowy kod do wykrywania kolizji członu 6 (J6, rysunek 1) manipulatora z otoczeniem zaimplementowany w C#.

Listing 1: Przykład obsługi kolizji w środowisku Unity zaimplemntowany przy użyciu C# dla członu 6 (J6)

```
using UnityEngine;
        public class CollisionHandler : MonoBehaviour
        {
        private void OnCollisionEnter(Collision collision)
                {
                // Ta metoda jest wywolywana, gdy nastapi kolizja.

                // Sprawdz, czy kolizja dotyczy konkretnego znacznika.
                if (collision.gameObject.CompareTag("J6"))
                {
                        // Wykonaj akcje w przypadku kolizji z okreslonym znacznikiem.
                        Debug.Log("Kolizja z obiektem oznaczonym J6");
                }

                // Mozna rowniez uzyskac dostep do informacji o zderzeniu,
                // takich jak punkty styku itp.
                ContactPoint contact = collision.contacts[0];
                Debug.Log("Kolizja w punkcie: " + contact.point);
                }
                // Mozna uzyc dodatkowych metod kolizji,
                // takich jak OnCollisionStay, OnCollisionExit itp.
        }
```

4. Podsumowanie

Celem projektu było stworzenie oprogramowania pozwalającego na wizualizację robota AdeptSix 300. Pozwala to na szybkie i wygodne przedstawienie pracy robota bez konieczności posiadania rzeczywistego manipulatora. Rozwiązanie to posiada możliwość bezpośredniej zmiany wartości parametrów członów robota w trybie online w czasie rzeczywistym. Do sterowania robotem używa się klawiatury oraz specjalnej nawigacji sterowanej myszką. Aplikacja stworzona w ramach niniejszego projektu, stanowi tanią i prostą w obsłudze alternatywę dla badania manipulatora AdeptSix 300. Projekt można rozwinąć o dodatkowe elementy. Poprzez protokół TCP/IP można się połączyć z rzeczywistym manipulatorem i pobierać informacje o położeniu osi, które w łatwy sposób można przedstawić w aplikacji.

Dzięki wykorzystaniu pakietów Blender i Unity 3D możliwe było stworzenie modelu całego stanowiska zrobotyzowanego co częściowo przedstawiono na rysunku 4, które z powodzeniem może być wykorzystane w procesie edukacyjnym związanym z programowaniem robotów a nawet można spróbować zbudować całą linię montażową, którą będzie można programować w jednym z wymienionych języków np. C# lub Python. Prezentowane rozwiązanie stanowi bardzo dobrą alternatywę dla komercyjnych rozwiązań związanych z budową i programowaniem stanowisk zrobotyzowanych takich jak Robot Studio [12]. Aktualnie nawet twórcy oprogramowania Unity 3D zauważyli potencjał w systemach wizualizacji robotów przemysłowych i mobilnych tworząc odpowiednie narzędzie (Unity Robotics Visualizations Package), które dodatkowo można połączyć z oprogramowaniem ROS (Robot Operating System). Systemy wizualizacji i symulacji zostały również dostrzeżone przez największe firmy technologiczne takie jak NVIDIA i ich środowisko Omniverse [11]. Jednak próg wejścia do ekosystemu NVIDIA jest dużo wyższy niż zaprezentowane rozwiązanie oparte o oprogramowanie Blender i Unity 3D.

LITERATURA

1. Adept Technology, Inc „AdeptSix 300 Robot Instruction Handbook, Rev. C”, 2005.
2. Alan Thorn A.: Unity i Blender. Praktyczne tworzenie gier, Helion, 2015.
3. Catarci T., Santucci G., Silva SF.: An interactive visual exploration of medical data for evaluating health centres, Journal of Research and Practice in Information Technology, 2003, vol. 35(2), p.99-119.
4. David M. Bourg D. M.: Fizyka dla programistów gier, Helion, 2003.
5. Geig M.:Unity. Przewodnik projektanta gier. Wydanie III, Helion, 2019.
6. Goldstone W.: Projektowanie gier w środowisku Unity 3.x, Helion, 2012.
7. Mullen T.: Blender. Mistrzowskie animacje 3D, Helion, 2010.
8. Skeet J.: C# od podszewki. Wydanie III, Helion, 2012.
9. Polychronidou E., Kalamaras I., K. Votis K., Tzovaras D.: Health Vision: An interactive web based platform for healthcare data analysis and visualisation, 2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Siena, Italy, 2019, p. 1-8.
10. www.blender.org
11. https://www.nvidia.com/en-us/omniverse/
12. https://new.abb.com/products/robotics/robotstudio
13. www.unity.com
14. https://docs.unity3d.com/ScriptReference/MonoBehaviour.html