Jarosław SOCHA[1], Bartłomiej ŻYWCZAK[2]

Opiekun naukowy: Sławomir HERMA[3], Dawid KOTRYS[4]

SFEROGEBRA

Streszczenie: Artykuł opisuje działanie aplikacji, napisanej w języku HTML, służącej do wizualizacji zjawisk geometrycznych zachodzących na powierzchni sfery. Praca została zainspirowana licznymi implementacjami systemów 2D (m.in. Geogebra), służących do nauki geometrii.

Słowa kluczowe: oprogramowanie, program, inżynieria, sfera, geometria, informatyka, HTML, wizualizacja, grafika

SFEROGEBRA

Summary: Article describes an application, written in HTML, for the visualization of geometric phenomena occurring on the surface of a sphere. The work was inspired by numerous implementations of 2D systems (including Geogebra) used to learn geometry.

Keywords: software, program, engineering, sphere, geometry, informatics, HTML, visualisation, graphics

1. O projekcie

Sferogebra to oryginalna nazwa projektu, jaki organizowałem w ramach nauki języka HTML. Każdy z uczniów miał przygotować dowolny projekt z dziedziny matematyki, fizyki lub informatyki w owym języku, a że działo się to podczas lekcji online, miałem nieco więcej czasu wolnego niż zwykle, a więc postanowiłem zrobić coś bardziej ambitnego. Jako przewodniczący szkolnego koła astronomicznego, oraz osoba silnie związana i zainteresowana tą dziedziną fizyki, postanowiłem zrobić projekt właśnie o tej tematyce, i zamiast zaprojektować zwykłą symulację układu słonecznego, postanowiłem stworzyć interaktywne narzędzie do rysowania figur geometrycznych na sferze.

[1] V Liceum Ogólnokształcące w Bielsku-Białej, jaro.socha@gmail.com
[2] mgr inż. V Liceum Ogólnokształcące w Bielsku-Białej, bartek@zywczak.pl
[3] dr inż., Katedra Inżynierii Produkcji, WBMiI ATH, sherma@ath.bielsko.pl
[4] dr , Katedra Matematyki, WBMiI ATH, dkotrys@ath.bielsko.pl

Nazwę „Sferogebra" stworzono łącząc słowa „Sfera" oraz nazwę znanego programu do rysowania na płaszczyźnie dwu i trójwymiarowej „Geogebra".

2. Główne problemy

Projekt był tworzony w ramach nauki języka HTML, CSS oraz JavaScript. Podstawowym problemem było stworzenie środowiska trójwymiarowego za ich pomocą. Aby to osiągnąć, użyto wolno dostępnej obszernej biblioteki „three.js", pozwalającej tworzyć właśnie takie środowisko. Biblioteka ta składa się z licznych klas i komend, które okazały się użyteczne dla wykonania niniejszej pracy.

Podczas procesu tworzenia samej aplikacji pojawiło się kilka problemów. Pierwszym z nich było stworzenie odpowiedniego systemu sterowania, który pozwoliłby na obrót wokół sfery oraz przybliżanie i oddalanie. Stworzenie takiego mechanizmu sterowania znacznie wydłużyłoby sam proces, dlatego wykorzystano wolno dostępny program używający biblioteki THREEjs „OrbitControls", który został zaimplementowany po niewielkiej modyfikacji.

Po stworzeniu sfery jednostkowej, na którą użytkownik będzie mógł nanosić obiekty, i opanowaniu problemów związanych z „raycastingiem" skonstruowano podstawową funkcję tworzenia punktów na sferze, w postaci drobnych sfer. W trakcie prac natrafiono na kilka problemów głównych, związanych z funkcjami aplikacji:

Tworzenie kół wielkich na sferze

Koło wielkie na sferze to okrąg o środku w środku sfery. Aby zdefiniować koło wielkie potrzebne są dwa punkty, przez które ono przechodzi. Wykorzystywana biblioteka „three.js" niestety nie posiada funkcji tworzenia koła przechodzącego przez dwa punkty. Wobec tego skorzystano z wbudowanej klasy wektorów oraz funkcji mnożenia wektorowego. Wykorzystując ten fakt otrzymano wektor prostopadły do płaszczyzny okręgu, po czym dokonano obrotu okręgu w stronę tego wektora.

Tworzenie kół małych

Koła małe to koła o środku poza środkiem sfery, o promieniu mniejszym od promienia sfery. Są one definiowane przez trzy dowolne punkty. W ich przypadku należało odnaleźć nie tylko ich obrót, ale także współrzędne środka i promień. Wiedząc, że rzut środka okręgu na powierzchnie sfery będzie leżał na kole wielkim, prostopadłym do koła wielkiego przechodzącego przez dwa dowolne jego punkty, stworzono poniższe funkcje:

```
function CreateCircleSetup (point1, point2){
 const v1 = new THREE.Vector3().crossVectors(point1,point2);
 const v2 = new THREE.Vector3().addVectors(point1,point2);
  v1.cross(v2);
  v1.normalize();
  return v1;
}
function CreateCircleData (point1, point2, point3) {
    const v1 = CreateCircleSetup(point1, point2);
    const v2 = CreateCircleSetup(point2, point3);
```

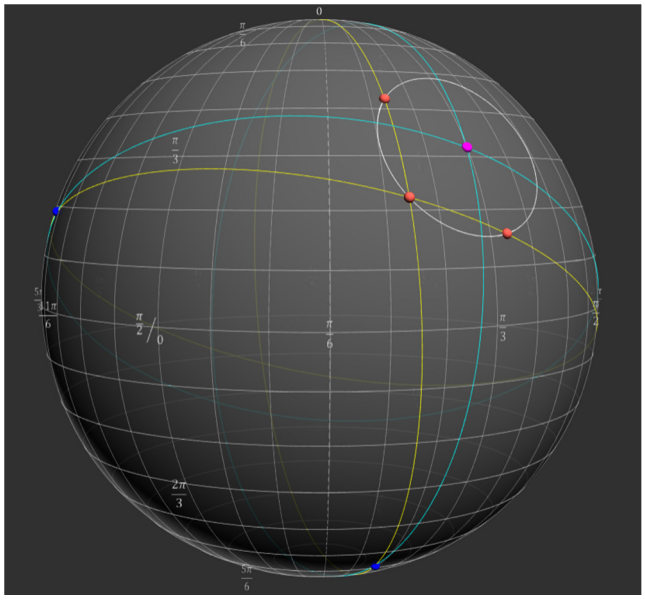
```
    const srodek = new THREE.Vector3().crossVectors(v1, v2).
normalize();

    const qw = new THREE.Quaternion();
    qw.setFromUnitVectors(unitVector, srodek);

    const Angle = point2.angleTo(srodek);
    const radius = Math.sin(Angle);
    srodek.multiplyScalar(Math.cos(Angle));

    return {
        qw: qw,
        radius: radius,
        offset: srodek,
    };}
```

W powyższym listingu funkcja „*cross*” oznacza mnożenie wektorowe, „*Vector3*” - klasę wektorów trójwymiarowych, „*normalize*” funkcję skalującą wektory tak aby miały długość 1, a „*quaternion*” służy do wyznaczenia kierunku koła.



Rysunek 1. Przykład działania aplikacji

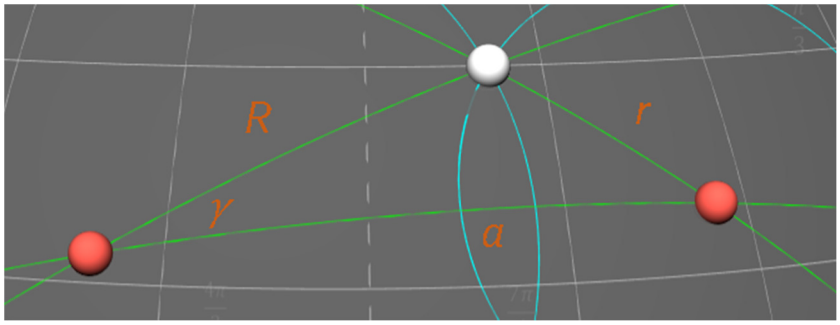
Na Rysunku 1 przedstawiono przykład działania fragmentu kodu, gdzie niebieskie punkty otrzymywane są za pomocą funkcji „*CreateCircleSetup*”, a ich znormalizowany iloczyn wektorowy to punkt rzutu środka koła na sferę.

Pomnożenie i dodanie wektorów w odpowiedniej kolejności skutkuje otrzymaniem współrzędnych środka koła, jego promienia i kierunku obrotu.

Znajdowanie punktów przecięcia dwóch kół

W przypadku kół wielkich nie jest to trudne, gdyż punkty przecięcia to iloczyny wektorowe wektorów kierunku kół wielkich (pierwszego z drugim i drugiego

z pierwszym). Trudniej jest jednak obliczyć współrzędne przecięcia kół małych. Aby to zrobić należy najpierw wyznaczyć kąty między środkami kół oraz kąty od środków kół do ich brzegów, a skoro znamy ich położenie oraz promienie to można tego dokonać.



Rysunek 2. Oznaczenie kątów przecięcia

Na Rysunku 2 przedstawiono wygląd przykładowego przecięcia wraz z oznaczeniami kątów, gdzie R i r to kąty między środkami kół i ich bokami, a to kąt między środkami kół, a γ to kąt między dwoma zielonymi ramionami zielonego trójkąta. Następnie korzystając z twierdzeniem cosinusów dla sfery, które w tym przypadku wygląda następująco:

$$\cos R = \cos r \cos a + \sin r \sin a \cos \gamma \tag{1}$$

otrzymano kąt γ. Następnie za pomocą funkcji obrotów wektorów o dany kąt wokół siebie otrzymano współrzędne punktów przecięcia. Gdy kąt a jest mniejszy od sumy R i r to punkty nie powstają.

System zależności

W momencie zmiany pozycję koła, na sferze, punkty przecięcia z innymi kołami, tak jak i inne oparte na tych kołach obiekty, również powinny ulec zmianie. Stworzony system został przygotowany na dodawanie kolejnych funkcji i nowych rodzajów obiektów w ten sposób, że każdy obiekt posiada własną listę obiektów od niego zależnych (zmieniających się gdy i on się zmienia) oraz obiektów od których zależy (gdy one się zmieniają, on także). Stworzone zostały również funkcje, które tak samo jak te tworzące kształty, za pomocą zależności z innymi obiektami, wyznaczają nowe współrzędne obiektom zależnym.

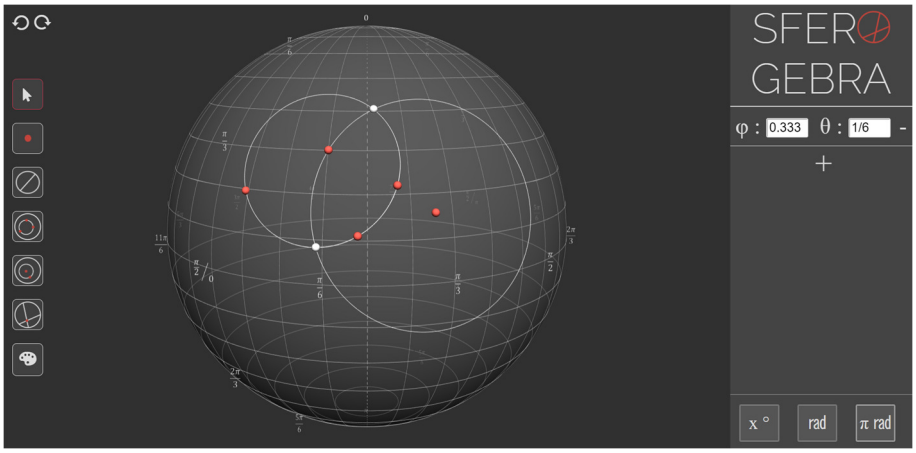
Cofanie i przywracanie obiektów

Każdemu obiektowi nadano miejsce w tabeli kolejności, a gdy jedna funkcja tworzyła kilka obiektów na raz, były one powiązane, aby podczas cofania znikać wspólnie. Stworzono też funkcję przywracania, oddającą cofnięte obiekty, przy czym jeżeli obiekty, od których były zależne zmieniły położenie, to przywrócone obiekty będą na nowych miejscach tak, jak zostały zdefiniowane podczas tworzenia. Gdy stworzony zostanie obiekt po cofnięciu innego, cofnięte obiekty zostaną zniszczone i zwolnią pamięć.

Dodawanie punktów na sferze za pomocą współrzędnych biegunowych

Tworzenie punktów o danych współrzędnych na sferze było jednym z największych problemów, ponieważ punkty można dodawać i kasować, a więc jeżeli punkt zostałby skasowany, a były na nim oparte jakiekolwiek inne obiekty, te muszą zostać także skasowane, razem z obiektami powiązanymi. Gdy użytkownik postanowi punkt przywrócić, muszą one powrócić razem z nim. Dodatkowo po zmianie jakiejkolwiek wartości współrzędnej trzeba było sprawdzić, dla którego punktu się zmieniła a następnie zaimplementować i zmienić położenia obiektów zależnych.

3. Autorski interfejs użytkownika oraz funkcje systemu



Rysunek 3. Interfejs aplikacji

Na Rysunku 3 przedstawiono wygląd interfejsu autorskiej aplikacji. Samą sferę obracać można za pomocą przytrzymania prawego przycisku myszy (niezależnie od używanej funkcji), oraz oddalać używając „rolki" myszy.

W lewym górnym rogu znajduje się funkcja cofania oraz przywracania obiektów. Po lewej stronie ustawione są funkcje, odpowiednio od góry:

1. Kursor – pozwala na przemieszczanie punktów
2. Tworzenie punktów – punkty można osadzać na kołach, ograniczając ich ruch tylko do nich samych,
3. Koło wielkie – tworzy koło wielkie przez dwa różne punkty (podgląd wyglądu koła wyświetla się po wybraniu pierwszego punktu)
4. Koło małe - tworzy koło małe przez trzy różne punkty (podgląd wyglądu koła wyświetla się po wybraniu drugiego punktu)
5. Koło małe ze środka - tworzy koło małe przez wybranie środka a następnie punktu boku (podgląd wyglądu koła wyświetla się po wybraniu środka)
6. Przecięcia kół – tworzy punkty przecięcia dwóch kół (domyślnie białe, aby odróżnić od punktów niezależnych – czerwonych, jeżeli koła się nie

przecinają punkty będą niewidoczne, ale pojawią się jeżeli zmienimy parametry kół tak, żeby się przecinały)

7. Paleta kolorów – po naciśnięciu obiektu wyświetla się menu kilkunastu kolorów, dzięki któremu można dostosować kolor obiektu dla wygody korzystania z programu.

Z prawej strony pod logiem znajduje się pole punktów o danych współrzędnych, gdzie można je tworzyć (+) i usuwać (-) oraz wpisywać ich współrzędne w układzie biegunowym. W ramki można wpisać liczbę dziesiętną bądź ułamek (jego wartość jest liczona za pomocą odrębnie stworzonej funkcji. Wpisanie wartości nie będącej liczbą ani ułamkiem skutkuje podkreśleniem ramki na kolor czerwony. W przypadku przekroczenia dostępnego miejsca ilością punktów pojawia się pasek przewijania z prawej strony. Na samym dole pola można zmienić jednostki ze stopni na radiany oraz wielokrotności π radianów.

Wielkości przycisków i funkcji dostosowują się dynamicznie wraz ze zmianą wielkości i kształtu okna. Wygląd strony oraz dobór wielkości i kolorów jest rozwiązaniem autorskim, a dla menu funkcji inspiracją była wspomniana wcześniej „Geogebra".

4. Podsumowanie

W przyszłości, program należałoby rozszerzyć o funkcje obliczania pól trójkątów sferycznych, wyświetlania kątów odcinków i kątów między kołami, wyświetlania wartości współrzędnych wybranego punktu, oraz możliwość wyboru trybu, w którym użytkownik jest wewnątrz sfery, aby można było prościej rozwiązywać zadania z dziedziny astronomii sferycznej.

Stworzony program nadaje się do rozwiązywania problemów matematycznych na sferze np. w zakresie astronomii sferycznej. Program dodatkowo można wykorzystać w celach edukacyjnych do nauki geometrii na sferze, z uwagi na intuicyjność. Program mógłby stanowić ułatwienie w interaktywnym poznawaniu geometrii sfery oraz jej tajników. Ponadto pozwoliłby nauczycielowi na przeniesienie dwuwymiarowych rysunków z tablicy w interaktywną przestrzeń 3D. Choć istnieją inne programy z zakresu geometrii trójwymiarowej, to z dydaktycznego i metodycznego punktu widzenia, zapoznawanie uczniów z geometrią sfery, właściwiej byłoby rozpocząć właśnie od dedykowanej aplikacji, pozbawionej zbędnego nadmiaru funkcji. Fakt, że aplikacja została stworzona z wykorzystaniem języka HTML, może również stanowić inspirację do tworzenia nieszablonowych rozwiązań edukacyjnych.

LITERATURA

1. Strona z listą funkcji, przykładami oraz podręcznikiem biblioteki three.js *threejs.org/*
2. Aplikacja "Geogebra" *geogebra.org/*