Aplikacja mobilna umożliwiająca umieszczenie wirtualnej grafiki w rzeczywistym położeniu.

Michał Aniserowicz
Wydział Elektroniki i Technik Informacyjnych
M.Aniserowicz@stud.elka.pw.edu.pl

Streszczenie *abstract.*

1. Wstęp wstęp

2. Opis aplikacji

3. Odczytywanie pozycji telefonu Komponent *PositionTracking* korzysta z metod odczytywania pozycji telefonu na kuli ziemskiej zaimplementowanych w API platformy Android. System udostępnia możliwość odczytywania pozycji z następujących źródeł [**?**]:

3.1. GPS Odczyt bezpośrednio z odbiornika GPS zamontowanego w urządzeniu. Jest to najdokładniejszy sposób określania pozycji telefonu - jego dokładność wynosi od kilku do kilkunastu metrów. Ma jednak znaczące ograniczenia:

- ze względu na konstrukcję systemu GPS, do poprawnego działania wymaga otwartej przestrzeni - używanie go w budynkach lub w ich pobliżu skutkuje niedokładnymi odczytami,
- przed uzyskaniem pierwszego odczytu musi nawiązać kontakt z kilkoma (ok. 7) satelitami GPS, co może potrwać nawet do kilku minut,
- pobiera najwięcej energii spośród wszystkich źródeł.

3.2. Assisted GPS Metoda Assisted GPS (AGPS, wspomagany GPS) oprócz odbiornika GPS wykorzystuje sygnał sieci komórkowej. Telefon otrzymuje od pobliskich stacji bazowych informacje na temat dostępności satelitów GPS oraz przybliżonych pozycji tych satelitów. Znacząco przyśpiesza to uzyskanie pierwszego odczytu, a także eliminuje wymóg otwartej przestrzeni. Co więcej, AGPS pobiera mniej energii niż bezpośredni odczyt GPS. Jego wadą jest jednak dokładność - posiłkowanie się przybliżonymi danymi pochodzącymi ze stacji bazowych powoduje jej spadek do kilkudziesięciu metrów.

3.3. Odczyt pasywny Metoda odczytu pasywnego nie korzysta z odbiornika GPS - jest więc najmniej dokładna (jej dokładność wynosi od kilkudziesięciu do kilkuset metrów). Polega na porównaniu siły sygnału odbieranego przez urządzenie od pobliskich stacji bazowych. Znając lokalizajcję stacji bazowych możliwe jest przybliżone określenie pozycji urządzenia. Dodatkowym źródłem informacji jest siła sygnału pochodzącego od sieci Wi-Fi wykrywanych przez urządzenie[1].

Wyboru źródła odczytu dokonuje programista, poprzez jawne wskazanie, którego dostawcy lokalizacji (ang. *LocationProvider*) chce użyć. Może również zdać się na system, formułując jedynie kryteria wyboru, takie jak wymagana dokładność odczytu, możliwość uzyskania informacji o prędkości poruszania się urządzenia i kierunku jego ruchu, czy zezwolenie na użycie metody, która wiąże się z naliczeniem opłat przez operatora sieci. PositionTracking używa domyślnego sposobu odczytu, podając domyślne kryteria wyboru. Pozwala to systemowi wybrać najdokładniejszy w danych warunkach bezpłatny *LocationProvider*.

Użytkownik komponentu może zlecić poje-

[1] Firma Google, podobnie jak inne firmy działające w sektorze mobilnym, zbiera i przechowuje informacje o lokalizacji sieci bezprzewodowych [**?**].

dynczy odczyt lub odczyt cykliczny z zadanym interwałem czasowym. W przypadku wyboru drugiej możliwości, powinien pamiętać o jawnym zakończeniu odczytu, gdy przestanie go potrzebować.

4. Problem niedokładności odczytów Błąd odczytu odbiornika GPS zaobserwowany w trakcie testowania *PositionTracking* w optymalnych warunkach, tj. na otwartym terenie, wahał się od kilku do kilkunastu metrów, w zależności od miejsca odczytu. O ile błąd na poziomie dwóch-trzech metrów można uznać za akceptowalny, o tyle niedokładność rzędu dwunastu-piętnastu metrów może bardzo niekorzystanie wpływać na doświadczenia użytkowników aplikacji stworzonych przy pomocy *fARmework*.

Za przykład posłużyć może gra w "policjantów i złodziei", w której osoba goniąca przez cały czas jest informowana o tym, jaki dystans powinna przebyć i w jakim kierunku się poruszać, aby dotrzeć do osoby uciekającej. Dystans ten typowo wynosił będzie od zera do kilkuset metrów.

4.1. Przykład W danym momencie policjant znajduje się w okolicach wydziału Elektroniki i Technik Informacyjnych (punkt A(52,219505° N, 21,012028° E)), a złodziej - na ulicy Lwowskiej (w punkcie B(52,220458° N, 21,012281° E)). Odległość między punktami wynosi 103 m, a azymut - 9,24°. Następują odczyty pozycji policjanta i złodzieja, oba z niedokładnością około dziesięciu metrów. Według odczytów, policjant znajduje się w punkcie A'(52,219518° N, 21,011892° E), a złodziej w punkcie B'(52,220466° N, 21,012431° E).

Obliczenia dla tych punktów wskazują, że policjant powinien przebyć 115,3 m w kierunku 19,2°. Niedokładność wskazań jest na tyle duża, że może zmylić policjanta - zamiast na ulicę Lwowską, prawdopodobnie pobiegnie on w kierunku ulicy Śniadeckich (sytuację obrazuje Rysunek **??**).

Jak widać, dokładność jest istotna nawet w sytuacjach, w których odległość pomiędzy telefonami wynosi ponad sto metrów. W przypadku metod odczytu udostępnianych przez system Android nie jest możliwe całkowite usunięcie błędu odczytu - wynika on z jakości odbiorników GPS montowanych w smartfonach. Można jednakże próbować go zmniejszyć poprzez rozszerzenie istniejących metod o operacje korygujące takie jak filtrowanie lub uśrednianie kolejnych odczytów. Alternatywnym podejściem jest zaimplementowanie nowej metody dającej w określonych warunkach lepsze wyniki niż metody podstawowe. To właśnie podejście zastosowano po stworzeniu podstawowej wersji komponentu, czego wynikiem było opracowanie i zaimplementowanie dodatkowej metody odczytu pozycji.

5. Odczytywanie orientacji telefonu Źródłem wiedzy o orientacji urządzenia są odczyty z wbudowanych w telefon sensorów: akcelerometru i magnetometru.

5.0.1 Akcelerometr

Dostarcza informacji o przyśpieszeniach działających na telefon. Przykładowo jeśli telefon trzymany jest pionowo w miejscu, to działa na niego jedynie siła grawitacji, dając odczyt wynoszący około $9,81\frac{m}{s^2}$ na osi y. Zorientowanie osi obrazuje Rysunek **??**.

Na podstawie informacji o tym, na jakie osie i w jakim stopniu działa siła grawitacji, można wnioskować o nachyleniu telefonu. Jako że akcelerometr mierzy przyśpieszenie wypadkowe, podczas ruchu urządzenia wnioskowanie to jest obarczone dużym błędem. Aby zminimalizować błąd, w systemie Android zastosowano filtr dolnoprzepustowy, który częściowo odcina wpływ poruszania na dokładność odczytów.

5.0.2 Magnetometr

Odczytuje wartości pola magnetycznego otaczającego urządzenie. Na podstawie tych wartości można określić, w którą stronę świata zwrócony jest telefon. Sensor podaje odczyty dla wszystkich trzech osi - orientacja urządzenia nie jest więc przeszkodą. Należy jednak zwrócić uwagę na to, że odczyty nie mają szansy być bardzo dokładne - magnetometr mierzy nie tylko pole magnetyczne Ziemi, ale także pola generowane przez inne

podzespoły telefonu, na przykład głośnika.

Klasa *SensorManager* należąca do API systemu Android udostępnia operację (*getRotationMatrix*) pozwalają wyznaczyć macierz obrotów telefonu. Macierz tę można zastosować do konwersji współrzędnych punktu z układu współrzędnych telefonu do układu współrzędnych jego otoczenia (zorientowanie osi otoczenia przedstawia Rysunek **??**). Aby ją uzyskać, należy pobrać odczyty akcelerometru i magnetometru i przekazać je metodzie *getRotationMatrix*.

6. Rozpoznawanie obrazu

7. Wyświetlanie obrazu Do generowania grafiki trójwymiarowej system Android wykorzystuje biblioteki OpenGL ES (OpenGL for Embedded Systems). Jest to odmiana biblioteki OpenGL dedykowana dla systemów wbudowanych używanych między innymi w konsolach i telefonach [**?**]. Android oferuje wsparcie dla dwóch standardów biblioteki:

- OpenGL ES 1.x - wykorzystuje potok ustalony (ang. fixed pipeline). Oznacza to, że transformacje oraz obliczenia koloru wierzchołków wykonywane w celu imitacji cieniowania i oświetlenia wykonywane są przez kartę graficzną w sposób narzucony z góry. Utrudnia to lub uniemożliwia uzyskanie wielu zaawansowanych efektów graficznych, takich jak mapowanie wypukłości[2], płaszczyzny refleksji[3] [**?**] czy renderowanie z użyciem szerokiego zakresu dynamicznego[4] [**?**].

- OpenGL ES 2.x - wykorzystuje potok programowalny (ang. programmable pipeline). Umożliwia pisanie krótkich programów w języku GLSL[5] nazywanych shaderami, które definiują sposób wykonywania obliczeń na wierzchołkach. Umożliwia to uzyskiwanie zaawansowanych efektów graficznych. Jest jednak trudne do zrozumienia dla początkujących programistów, gdyż wykonywanie transformacji lub oświetlenie sceny wymaga podania kodu źródłowego tych operacji jako wartości zmiennych napisowych (*String*) - poprawność tego kodu nie jest sprawdzana przez kompilator, co znacznie utrudnia wykrycie ewentualnego błędu.

Ze względu na brak wykorzystania efektów wymagających użycia potoku programowalnego, *fARmework* wykorzystuje bibliotekę OpenGL ES w standardzie 1.x. Klasy jednakże zostały wydzielone w sposób, który umożliwia transparentną wymianę wersji biblioteki bez konieczności redefiniowania modelu.

7.1. Definicja modelu Renderowany na ekranie obiekt rozszerza klasę *Model* i jest opisany trójkątnymi ścianami. Do zdefiniowania modelu wymagane są następujące informacje:

- tablica współrzędnych wierzchołków obiektu - każdy wierzchołek opisany jest trzema współrzędnymi (x, y, z),

- tablica kolorów wierzchołków obiektu - każdy wierzchołek opisany jest czterema wartościami (r, g, b, a) definiującymi jasności kanałów: czerwonego, zielonego, niebieskiego i przezroczystości.

W komponencie zdefiniowany jest również specjalny typ modelu - *PhasingModel*. Ma on zdolność automatycznej transformacji koloru wierzchołków na podstawie zadanego współczynnika z zakresu $[0, 1]$. Przed wyświetleniem, wybrane przez programistę składowe barw wierzchołków modelu są mnożone przez ten współczynik. Dzięki temu, wraz ze zmianą wartości współczynnika, model automatycznie zmienia swój kolor.

[2]Mapowanie wypukłości (ang. bump mapping) - technika umożliwiająca symulowanie obiektów o porowatej powierzchni.

[3]Płaszczyzny refleksji (ang. reflection planes) - technika umożliwiająca uzyskanie efektu lustrzanego odbicia sceny na danej powierzchni.

[4]Renderowanie z użyciem szerokiego zakresu dynamicznego (ang. high dynamic range rendering) - technika umożliwiająca uzyskanie realistycznego oświetlenia przez wierne odwzorowanie bardzo jasnych i ciemnych elementów sceny.

[5]GLSL - OpenGL Shading Language, http://www.opengl.org/documentation/glsl/.

Komponent zawiera jeden przykładowy model typu *PhasingModel*, przygotowany na potrzeby gry *Hare and Houds*. Jest to strzałka, która wskazuje użytkownikowi kierunek poruszania się. Przykłady kolorowania tego modelu prezentuje Rysunek **??**.

7.2. Proces renderowania Dla każdego zdefiniowanego modelu moduł renderujący grafikę automatycznie:

1. Oblicza i normalizuje wartości wektorów normalnych dla każdej ściany, co umożliwia poprawne oświetlenie obiektu.
2. Ustawia położenie kamery oraz definiuje sposób wyświetlania sceny:
   - kamera jest automatycznie pozycjonowana ponad zdefiniowanym obiektem,
   - scena wyświetlana jest domyślnie z użyciem projekcji ortogonalnej, co zwiększa czytelność modelu.
3. Dokonuje obrotu modelu zgodnie z przekazaną do modułu macierzą obrotów. Pozwala to wypozycjonować go tak, aby niezależnie od orientacji telefonu pozostawał nieruchomy względem otoczenia.
4. Dokonuje obrotu modelu zgodnie z przekazanym do modułu azymutem. Pozwala to na orientację obiektu tak, aby wskazywał zadany kąt względem magnetycznej północy.
5. Definiuje kolor wierzchołków (w przypadku modelu typu *PhasingModel*).
6. Rysuje zdefiniowany obiekt.
7. Oświetla scenę.

Dzięki temu programista może swobodnie używać komponentu bez znajomości biblioteki OpenGL. Wystarczy, że wygeneruje współrzędne oraz kolory wierzchołków trójwymiarowego obiektu (co można wykonać automatycznie narzędziami takimi jak 3ds Max[6]) i jako model przekaże je do komponentu (patrz sekcja **??**), a reszta pracy zostanie wykonana samoczynnie.

7.3. Obracanie modelu Aby utrzymać model w pozycji pozornie nieruchomej względem otoczenia telefonu, niezbędna jest wiedza o aktualnej orientacji urządzenia i obracanie modelu przeciwnie do tej orientacji.

7.4. Orientacja urządzenia a tryb wyświetlania Macierz obrotu zwracana przez metodę *getRotationMatrix* przechowuje wartości określające fizyczną orientację telefonu. Obrót modelu według tych wartości będzie niepoprawny, jeśli ekran urządzenia będzie wyświetlał treści w trybie innym niż domyślny. Dwa podstawowe tryby wyświetlania, to:

- portrait mode (tryb portretu, domyślny) - tryb używany, gdy telefon znajduje się pozycji pionowej,
- landscape mode (tryb krajobrazu) - tryb używany w pozycji poziomej; interfejs graficzny aplikacji jest odpowiednio obrócony.

Przełączanie pomiędzy trybami odbywa się automatycznie w zależności od pozycji, w jakiej znajduje się urządzenie. Zachowanie to można wyłączyć korzystając z ustawień systemu, a także kontrolować z poziomu aplikacji - dlatego wyznaczenie poprawnej macierzy obrotów jedynie na podstawie odczytów sensorów nie zawsze jest możliwe.

Android udostępnia informacje o aktualnie stosowanym trybie wyświetlania (dostarcza ich metoda *getRotation* klasy *Display*). W przypadku, gdy jest on inny od domyślnego, macierz należy skorygować, tzn. zamienić miejscami dwie z trzech osi. Na przykład przy obrocie ekranu o 90° w prawo, oś y wskazuje tam, gdzie poprzednio wskazywała oś x, a oś x - w kierunku przeciwnym do pierwotnie wskazywanego przez oś y. Podając macierz obrotu i sposób zamiany osi jako argumenty operacji *remapCoordinateSystem* klasy *SensorManager*, komponent uzyskuje skorygowaną macierz.

7.5. Obrót modelu Skorygowana macierz obrotu bez dalszych modyfikacji może zostać użyta do obrócenia wyświetlanego modelu tak, aby zniwelować obrót urządzenia.

[6] 3ds Max - oprogramowanie służące do modelowania, renderowania i animowania grafiki 3D. Strona produktu dostępna jest pod adresem: http://usa.autodesk.com/3ds-max/.

Przed każdorazowym wyrenderowaniem modelu na ekranie telefonu, pobierana jest ostatnio odczytana macierz. Przy pomocy funkcji OpenGL model jest obracany zgodnie z wartościami macierzy, a następnie rysowany. Rysunek **??** prezentuje kilka przykładów wyświetlania obróconego modelu strzałki.

Model strzałki domyślnie wskazuje północ. Wskazywany kierunek można jednak dowolnie zmieniać - obrotem modelu może sterować komponent zewnętrzny.

8. Podsumowanie

Literatura

[1] *Wielki słownik ortograficzny PWN*, pod red. Edwarda Polańskiego, Wyd. 3 popr. i uzup., Warszawa, Wydawnictwo Naukowe PWN, 2012, ISBN 978-83-01-16405-8.