XXII. Gry z jednego miejsca na drugie

Kiedy narzędzie miało pomóc Stanom Zjednoczonym w prowadzeniu międzykontynentalnych rakiet balistycznych, Globalny System Pozycjonowania (GPS) ewoluował, by stać się częścią naszego codziennego życia. Obecne pokolenie nigdy nie poznałoby świata, w którym zgubienie byłoby czymś, czego nie można było naprawić, trilaterując swoją pozycję między satelitami krążącymi wokół planety. Chociaż GPS stał się powszechny w świecie nawigacyjnym, rozprzestrzenianie się smartfonów właśnie otwiera drzwi do gier GPS. Podczas gdy ten gatunek właśnie się pojawia, chcielibyśmy dać wam wprowadzenie do fizyki stojącej za GPS i aktualnymi aplikacjami w świecie gier. Przypomnijmy, że pozycje w pobliżu powierzchni ziemi są zwykle podawane w geograficznym układzie współrzędnych, częściej określane są jako szerokość, długość i wysokość. Szerokość geograficzna jest miarą stopnia oddalenia się północy lub południa od równika. Długość geograficzna to miara tego, jak daleko na wschód lub na zachód jesteś od głównego południka. Południk jest linią o stałej szerokości geograficznej biegnącą od bieguna północnego do bieguna południowego. Główny południk jest arbitralnie definiowany jako południk, który przechodzi przez Obserwatorium Greenwich w Wielkiej Brytanii. Wysokość jest zwykle podawana jako miara tego, jak daleko powyżej lub poniżej poziomu morza znajdujesz się w punkcie opisanym przez szerokość i długość geograficzną.

Gry oparte na lokalizacji

Zanim przejdziemy do fizyki stojącej za GPSem, chcielibyśmy poświęcić chwilę na omówienie sposobu implementacji GPS w grach. W tej chwili jest to rynek wschodzący, który dopiero zaczyna się rozwijać. Istnieje kilka szerokich kategorii, w które wchodzą gry. Kolejny krok poza tym, co zrobił akcelerometr, GPS umożliwia użytkownikom przenoszenie gier komputerowych nie tylko poza kanapą, ale także na zewnątrz.

Geocaching i odwrotne geocaching

Geocaching to najstarsza forma gier z udziałem GPS. Powstał po selektywnym dostępność została usunięta z GPS, dzięki czemu jest bardziej dokładna, w roku 2000. W jego najbardziej podstawową formą, jest proces polowania na "cache" przy użyciu dostarczonych współrzędnych GPS. Pamięć podręczna zwykle ma dziennik i może zawierać inne przedmioty, takie jak monety z numerami seryjnymi, które znalazca może przenieść do innej pamięci podręcznej i śledzić online. Ze względu na dużą liczbę ustawień związanych z wdrażaniem gry geocaching na skalę komercyjną, większość implementacji jest oparta na społeczności. Jednak odwrotne geocaching jest bardziej obiecujące dla branży gier. W tym wariancie nic nie ma w dostarczonych współrzędnych, ale podróżowanie do nich jest wymagane, aby wykonać pewne działanie. Pomyśl o tym, jako o przenoszeniu pamięci podręcznej, której nie można odblokować, dopóki nie znajdzie się w zasięgu określonej współrzędnej. Może to zostać wykorzystane do zmuszania użytkowników do podróży w celu odblokowania przedmiotu gry. Na przykład, aby uzyskać umiejętność posługiwania się mieczem w grze, użytkownik musi udać się do najbliższego sklepu z artykułami sportowymi. Możliwości komercyjne korporacyjnych powiązań są oczywistym plusem.

Mieszana rzeczywistość

Gry o mieszanej rzeczywistości są podobne do geocaching. Wykraczają poza zwykłe wykorzystanie współrzędnych użytkownika do wywoływania zdarzeń, do korzystania z lokalnych mieszkańców. Obecnym przykładem jest Famiglia Gbangi. W tej grze ruch w realnym świecie pozwala odkryć wirtualne zakłady w świecie gry. Oddziela je od rzeczywistych fizycznych lokalizacji zgłaszanych przez GPS, ale wymaga przemieszczania się między lokalizacjami w świecie rzeczywistym, aby przenieść swoją postać w wirtualnym świecie. Popularna jest teraz aplikacja FourSquare na urządzenia mobilne.

Jest to najprostsza możliwa implementacja gier w mieszanym świecie. FourSquare pozwala użytkownikowi zostać burmistrzem danego miejsca, jeśli "sprawdza się" w lokalizacji bardziej niż ktokolwiek inny.

Gry uliczne

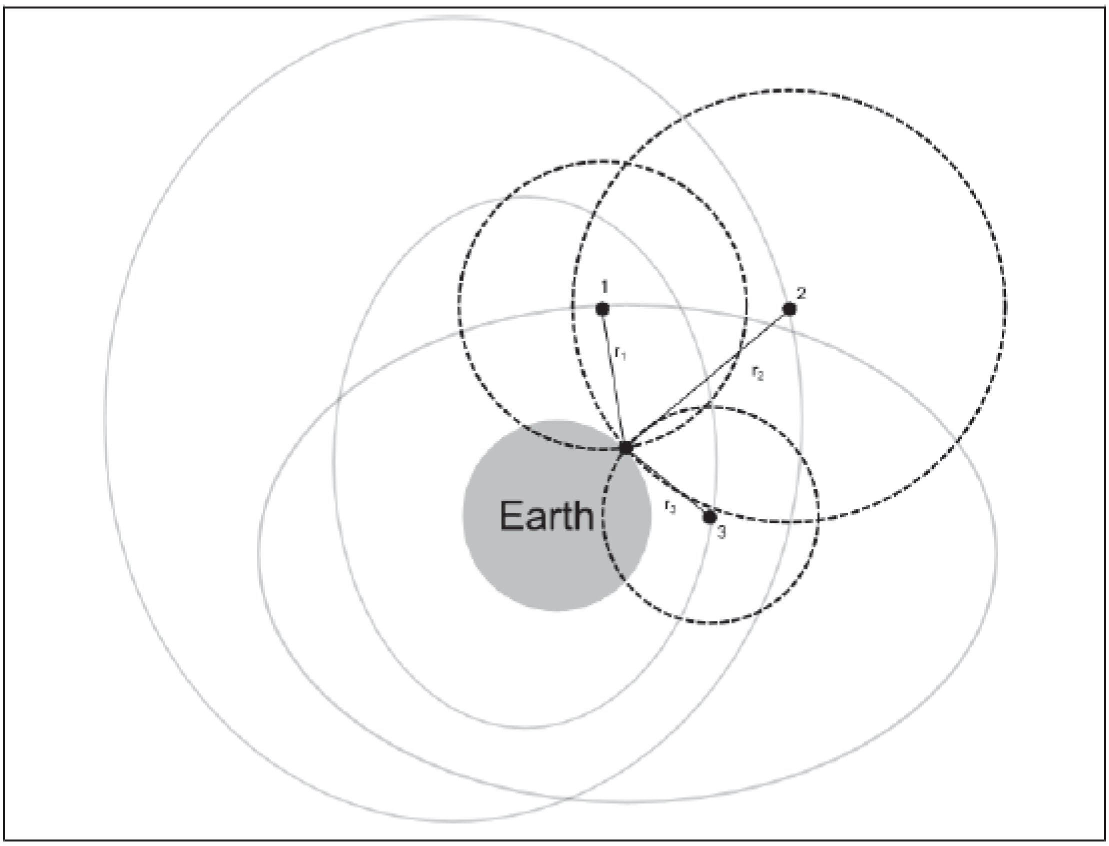
Gry uliczne to kolejny krok poza rzeczywistość mieszaną. Zmieniają one otoczenie użytkownika na wirtualną planszę. Jednym z przykładów jest niedawna gra multiplayer Pac-Manhattan wykorzystująca GPS w smartfonach do grania na żywo Pac-Mana w Washington Square Park. Ogólnie rzecz biorąc, chodzi o stworzenie sądu do gry z wykorzystaniem środowiska otaczającego użytkownika. Relacja między użytkownikami jest śledzona w wirtualnej przestrzeni gry i zapewnia elementy interaktywne.

Która godzina?

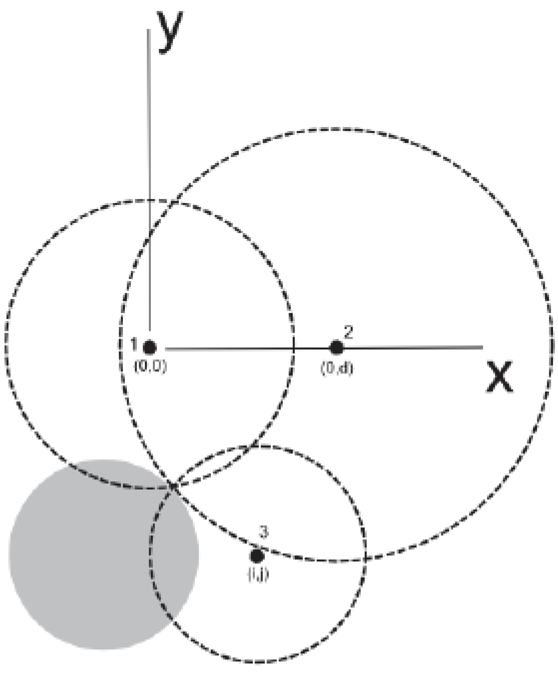
Historia GPS zaczyna się od nagrody zaoferowanej przez brytyjski rząd w 1717 roku dla tego kto w prosty sposób określi swoją długość geograficzną. Przyznano w 1773 roku przyjęte rozwiązanie było porównanie południa lokalnego z oficjalnym południem obserwowanym w Greenwich Observatory. Różnica między tymi dwoma czasami pozwoli powiedzieć, jak daleko byłeś za obserwatorium. SSatelity krążą wokół Ziemi, emitując wiadomości ze znacznikiem czasu. Obliczając różnicę między czasem otrzymania wiadomości a czasem jej przesłania, możemy obliczyć naszą odległość od satelity. W obu przypadkach potrzebujesz dokładnego sposobu na zachowanie lub zmierzenie czasu. Dla żeglarzy w 1800 roku, urządzenie było nowo wymyślonym chronometrem. Dla nas jest to zegar atomowy. Ponieważ sygnały z satelity GPS poruszają się z prędkością światła, potrzebny jest bardzo dokładny zegar, aby śledzić, jak długo trwało podróżowanie do Ciebie. Na przykład, jeśli zegar, którego używasz do czasu, kiedy nadejdzie sygnał, jest wyłączony w ciągu 1 mikrosekund, oszacujesz odległość przekraczającą 900 mil. Po stronie podaży sygnału, każdy satelita ma zegar atomowy, a wewnętrzny czas GPS jest dokładny do około 14 nanosekund. Problem polega na tym, że potrzebny jest również bardzo dokładny zegar w odbiorniku i byłoby dość trudno dopasować zegar atomowy do telefonu ekonomicznie. Aby obejść ten problem, odbiornik musi ustalić aktualny czas w oparciu o sygnały z satelitów.

Dwuwymiarowe przedstawienie matematyczne

W tej sekcji dowiesz się, w jaki sposób systemy GPS określają swoją lokalizację. To tło pomoże w wielu zastosowaniach geometrii w grach ogólnie, ale większość urządzeń GPS wykonuje intensywne podnoszenie i raportuje za pośrednictwem interfejsu API swoją szerokość i długość geograficzną. Niektóre interfejsy API mogą zawierać więcej informacji - na przykład obecny interfejs API iOS, nazwany Core Location, podaje aktualną szerokość i długość geograficzną, kierunek podróży, przebytą odległość i odległość w metrach do danej współrzędnej. Podaje także szacunkową wartość błędu związanego z jego pozycją w metrach. Jednym ze sposobów zdobycia pozycji dzięki informacjom dostarczanym przez GPS jest technika zwana trilateracją. Zamierzamy potraktować ten problem jako matematyczny zabieg w dwóch wymiarach. Możesz rozszerzyć to do trzech wymiarów, używając kulek zamiast kół. Na początek możemy wymienić nasze niewiadome: naszą współrzędną x i współrzędną y w przestrzeni oraz błąd w zegarze odbiorcy (lub odchylenie), b. W dwuwymiarowej płaszczyźnie trilateracja pomiędzy trzema okręgami daje dokładną pozycję; w przestrzeni trójwymiarowej wymagane są cztery sfery do określenia wszystkich trzech specjalnych współrzędnych. Zauważmy, że gdybyśmy przyjęli założenie o byciu na powierzchni jakiegoś geometrycznego kształtu, takiego jak ziemia, moglibyśmy zmniejszyć liczbę niewiadomych. Żadne uproszczenie nie jest tu stosowane, aby zapewnić najbardziej ogólny przypadek. W naszym przykładzie, jesteśmy gdzieś na powierzchni dwuwymiarowej ziemi, pokazanej na rysunku 22-1 jako jasnoszary dysk twardy.



Ten dysk jest orbitowany przez kilka satelitów GPS. Orbity satelitów są regularne, a ich pozycje w dowolnym czasie są zestawione w almanach przechowywany w odbiorniku. Czas transmisji jest zakodowany w sygnale tak, że dane są x_i, y_i i t_i, gdzie i = 1,2,3. Aby nam ułatwić, porzucimy układ współrzędnych Ziemi i użyjemy układu współrzędnych wyznaczonego przez nasze trzy satelity. Początek będzie na satelicie 1, oś X będzie bezpośrednio z satelity 1 prosto do satelity 2, a oś y będzie prostopadła do tej. Pokazano to na rysunku 22-2.



Równania trzech okręgów są zatem następujące:

$$r^2_1 = x^2+y^2$$
$$r^2_2 = (x-d)^2+y^2$$
$$r^2_3 = (x-i)^2+(y-j)^2$$

Każdy promień znajduje się przez odjęcie czasu transmisji, ti od bieżącego czasu i pomnożenie przez prędkość światła. Ponieważ prędkość światła jest bardzo duża, a nasz obecny czas jest tylko przybliżonym oszacowaniem, promienie te są powszechnie nazywane - pseudonimami, aby przypomnieć nam, że są nadal przybliżone. Wartości xiy, które spełniają te równania, są naszą obecną lokalizacją w przestrzeni dwuwymiarowej. Teraz odejmujemy drugie równanie od pierwszego:

$$r_1^2 - r_2^2 = x^2 + y^2 - (x\text{-}d)^2 - y^2$$
$$r_1^2 - r_2^2 = x^2 - (x\text{-}d)^2$$

Rozwiązanie dla x daje

$$x = (r_1^2 - r_2^2+d^2) / 2d$$

gdzie d jest odległością między znanymi lokalizacjami satelity 1 i satelity 2. Teraz zamień naszą współrzędną x z powrotem na równanie pierwszego koła:

$$y^2 = r_1^2 - [((r_1^2 - r_2^2 + d^2)^2) / (4d^2)]$$

i wreszcie po pobraniu pierwiastka kwadratowego:

$$y \quad = \quad \pm\sqrt{r_1^2 - \frac{(r_1^2 - r_2^2 + d^2)^2}{4d^2}}$$

zauważ, że wartość y jest teraz wyrażana jako dodatni lub ujemny pierwiastek kwadratowy. Oznacza to, że może istnieć zero, jeden lub dwa rozwiązania liczb rzeczywistych. Jeśli kręgi się nie przecinają, to ilość pod pierwiastkiem kwadratowym będzie ujemna, a wartość y będzie miała zero rzeczywistych rozwiązań. Jest to mało prawdopodobne w przypadku pierwszych dwóch satelitów, ponieważ otrzymałeś już od nich pseudonimy w postaci r_1 i r_2, które algorytm zakłada zerowy błąd. Jeśli dwa kręgi zdarzą się przecinać tylko w jednym punkcie stycznym, y będzie miało jedno rozwiązanie i będzie równe 0. Jest to również mało prawdopodobne. Najbardziej prawdopodobnym wynikiem będzie to, że y będzie zbiorem dwóch wartości, plus lub minus pierwiastek kwadratowy z wartości dodatniej, i że te dwa punkty będą szeroko rozdzielone. Teraz, jeśli uwzględnimy założenie, że jesteśmy na powierzchni Ziemi, możemy już zerwać więź między tymi dwoma punktami, wybierając obszar najbliższy powierzchni Ziemi. Jednakże nadal będziemy musieli radzić sobie z prawdopodobieństwem wystąpienia dużego błędu w naszej pozycji, biorąc pod uwagę nieprecyzyjny zegar w odbiorniku. Możemy naprawić naszą pozycję (x, y) bez żadnych założeń i uwzględnić odchylenie zegara, wprowadzając trzeci punkt i jego pseudonim. Teraz było bardzo prawdopodobne, że kręgi uzyskane z dwóch pierwszych satelitów przecinają się ze względu na sposób rozmieszczenia satelitów GPS wokół Ziemi. Ponieważ nasze obliczenia używają pseudonimów, jest mało prawdopodobne, że trzecie koło przejdzie bezpośrednio przez jeden z nich dwa punkty zdefiniowane przez przecięcie dwóch pierwszych okręgów. Aby usunąć takie błędy związane z odległościami, najpierw obliczamy, który punkt (x, y) lub (x, -y) jest bliżej (i, j) i wybieramy, że jest to nasza zakładana lokalizacja. Różnica między mniejszą z tych dwóch odległości a pseudorange r3 jest wtedy naszą korektą odległości, da. Ponieważ sygnał porusza się z prędkością światła, następujący iloraz zapewnia oszacowanie błędu między właściwym czasem a czasem odbiorcy:

$b = d_a/c$

Ponieważ wszystkie satelity GPS mają zsynchronizowane zegary atomowe, dla każdego z nich istnieje takie samo odchylenie. Oznacza to, że obliczone przez nas odchylenie wpłynęłoby w rzeczywistości na pierwsze pseudonimy, których użyliśmy do znalezienia naszego początkowego błędu. Dlatego konieczne jest podejście iteracyjne, aby dostosować wszystkie zmienne w czasie rzeczywistym, aż do zejścia. Bardziej bezpośrednie, ale mniej oczywiste, algebraiczne rozwiązanie, które nie wymaga żadnej iteracji, zostało opracowane przez Stephena Bancroft. Poza błędami zegara, inne błędy są wprowadzane przez atmosferę, sygnały odbijają się od ziemi iz powrotem do odbiornika, efekty relatywistyczne i dryf zegara atomowego. Wszystkie one są uwzględniane w modelach matematycznych zastosowanych do nieprzetworzonych danych pozycji. Na przykład, zegary GPS tracą około 7214 nanosekund dziennie z powodu ich prędkości zgodnie ze szczególną teorią względności. Ponieważ jednak są one wyższe w studni grawitacyjnej Ziemi, zyskują 45.850 nanosekund dziennie, zgodnie z ogólną teorią względności. Efekt netto można znaleźć, dodając te wartości razem: przebiegają one o 38,640 nanosekundy szybciej każdego dnia, co powodowałoby około 10 kilometrów niedokładności budowania każdego dnia na orbicie. Aby to uwzględnić, zegary w odbiornikach GPS są wstępnie wyregulowane z 10,23 MHz na 10,229999999543 MHz. Fakt, że dajemy liczbę do 11 miejsc po przecinku, pokazuje stopień dokładności, z jakim cieszy się współczesny wiek. Po rozwiązaniu błędu i usunięciu wszystkich innych możliwych błędów, rozwiązanie konwergentne można przełożyć z powrotem do dowolnego układu współrzędnych, który jest wygodny do podania użytkownikowi końcowemu. Zwykle jest to szerokość, długość i wysokość. Następnie nauczymy się obliczać różne ilości w oparciu o geograficzny układ współrzędnych.

Lokalizacja, lokalizacja i lokalizacja

Poświęćmy chwilę, aby omówić odległość między dwiema współrzędnymi szerokości i długości geograficznej. Możesz mieć ochotę obliczyć go jako odległość między dwoma punktami. W przypadku bardzo małych odległości to przybliżenie jest prawdopodobnie wystarczająco dokładne. Ponieważ jednak ziemia jest w rzeczywistości kulą, na dużych odległościach obliczona trasa będzie znacznie krótsza niż faktyczna odległość wzdłuż powierzchni. Najkrótsza odległość między dwoma punktami na kuli, szczególnie w problemach nawigacji, nazywana jest wielkim kręgiem. Wielki okrąg to przecięcie sfery i płaszczyzny zdefiniowanej przez punkt środkowy kuli, punkt początkowy i cel. Wynikowy kurs faktycznie ma nagłówek, który ciągle się zmienia. Na statkach można tego uniknąć korzystania z linii loksodromy, która jest najkrótszą drogą stałego kursu. Ułatwia to nawigację kosztem czasu. Samoloty jednak podążają szlakami wielkich kół, aby zminimalizować spalanie paliwa.

Odległość

Istnieje kilka sposobów obliczania odległości wzdłuż wielkiego koła. Omówimy tutaj formułę haversine. Istnieją inne metody, takie jak sferyczne prawo cosinusów i formuła Vincenty, ale haversine jest bardziej dokładna dla małych odległości niż sferyczne prawo cosinusów, pozostając znacznie prostszym niż formuła Vincent'a. Formuła haversine dla odległości to:

$d = (R)\ (c)$

gdzie R = promień Ziemi, a c to odległość kątowa w radianach podana przez:

$$c = 2 \arcsin\left(\sqrt{a}\right)$$

Tutaj, a jest kwadratem o połowę długości cięciwy między dwoma punktami obliczonymi jako:

$$a = \sin^2(\Delta_{lat}/2) + \cos(lat_1)\cos(lat_2)\sin^2(\Delta_{long}/2)$$

W końcu

$$\Delta_{long} = long_2 - long_1$$
$$\Delta_{lat} = lat_2 - lat_1$$

Pamiętaj, aby najpierw przekonwertować kąty na radiany przed użyciem ich w funkcji trygonometrycznej. Następnie zaczniemy pokazywać implementację kilku różnych formuł w Objective-C; jednak powinny one tłumaczyć się do C z niewielką modyfikacją. Wszystkie one wykorzystują następującą strukturę danych do przechowywania informacji o szerokości i długości geograficznej:

```
typedef enum {

float lat;

float lon;

} Coordinate2D;
```

Biorąc to pod uwagę, implementacja haversine wyglądałaby następująco:

```
float distanceGreatCircle(Coordinate2D startPoint, Coordinate2D endPoint){

//Convert location from degrees to radians

float lat1 = (M_PI/180.) * startPoint.lat;

float lon1 = (M_PI/180.) * endPoint.longi;

float lat2 = (M_PI/180.) * endPoint.lat;

float lon2 = (M_PI/180.) * endPoint.longi;

//Calculate deltas

float dLat = lat2 - lat1;

float dLon = lon2 - lon1;

//Calculate half chord legnth

float a = sin(dLat/2) * sin(dLat/2) + cos(lat1) * cos(lat2) * sin(dLon/2)

* sin(dLon/2);

//Calculate angular distance

float C = 2 * atan(sqrt(a)/sqrt(1-a));
```

```
//Find arclength
float distance = 6371 * C; //6371 is radius of earth in km
return distance;
}
```

Jednym z ograniczeń poprzedniej metody jest to, że jeśli te dwie lokalizacje są prawie antipodalne - to jest po przeciwnych stronach Ziemi - wówczas formuła haversine może mieć zaokrąglenia, co może skutkować błędami rzędu 2 km. Będą one jednak na dystansie 20 000 km. Jeśli wymagana jest skrajna dokładność w przypadku prawie współrzędnych antipodalnych, można cofnąć się do kulistego prawa cosinusów, które najlepiej nadaje się na duże odległości, takie jak przypadek antypodalny.

Nagłówek Wielkiego Kręgu

Jak wspomniano wcześniej, aby podążać najkrótszą ścieżką między dwoma punktami na kuli, musisz podróżować po wielkim kręgu. Wymaga to jednak ciągłego zmieniania pozycji w czasie. Formuła do obliczenia początkowego kursu lub azymutu do przodu to:

$$\Theta_i = \operatorname{atan2}[\sin(\Delta_{long})\cos(lat_2), \cos(lat_1)\sin(lat_2) - \sin(lat_1)\cos(lat_2)\cos(\Delta_{long})]$$

Przypomnijmy, że atan2 jest dwuargumentową odmianą funkcji arcus tangens. Zwraca znormalizowany kąt w radianach między -π a π (-180 ° i 180 °). Kod, który oblicza wartość i zwraca namiar kompasu jest następujący:

```
float initialBearing (Coordinate2D startPoint, Coordinate2D endPoint){
//Convert location from degrees to radians
float lat1 = (M_PI/180.) * startPoint.lat;
float lon1 = (M_PI/180.) * startPoint.longi;
float lat2 = (M_PI/180.) * endPoint.lat;
float lon2 = (M_PI/180.) * endPoint.longi;
//Calculate deltas
float dLat = lat2 - lat1;
float dLon = lon2 - lon1;
// Calculate bearing in radians
float theta = atan2f( sin(dlon) * cos(lat2), cos(lat1)*sin(lat2)-sin(lat1)*cos(lat2)
*cos(dlon));
//Convert to compass bearing
Float bearing = theta * (180 / M_PI); //radians to degrees
bearing = ( bearing > 0 ? bearing : (360.0 + bearing)); //fix range
```

```
return bearing;

}
```

Kąt ujemny obejmuje rozpoczynanie od 0 ° i obracanie w kierunku malejącym, ale kompasy nie są oznaczone wartościami ujemnymi! Aby to naprawić, wiersz z komentarzem "Zakres ustalania wartości" używa operatora potrójnego do stwierdzenia, że jeśli namiar ma mniej niż 0, zwróć wartość, którą odczytałby kompas. Na przykład, jeśli łożysko wynosiło -10 °, to łożysko kompasu wynosi -10 ° + 360 ° = 350 °. Jeśli wartość jest dodatnia, to po prostu zwraca tę samą wartość. Aby znaleźć ostateczne łożysko, po prostu bierzemy pierwsze łożysko idące od punktu końcowego do punktu początkowego, a następnie odwracamy je. Kod jest tworzony w następujący sposób:

```
float finalBearing (Coordinate2D startPoint, Coordinate2D endPoint){

//Convert location from degrees to radians

float lat1 = (M_PI/180.) * endPoint.lat;

float lon1 = (M_PI/180.) * endPoint.longi;

float lat2 = (M_PI/180.) * startPoint.lat;

float lon2 = (M_PI/180.) * startPoint.longi;

//Calculate deltas

float dLat = lat2 - lat1;

float dLon = lon2 - lon1;

//Calculate bearing in radians

float theta = atan2f( sin(dlon) * cos(lat2), cos(lat1)*sin(lat2)-sin(lat1)*cos(lat2)

*cos(dlon));

//Convert to compass bearing

float bearing = theta * (180 / M_PI); //radians to degrees

bearing = ( bearing > 0 ? bearing : (360.0 + bearing)); //fix range

bearing = ((bearing + 180) % 360) //reverse heading

return bearing;

}
```

Różnica polega na tym, że odwróciliśmy lat1, long1 i lat2, long2, konwertując lokalizacje na radiany. Ponadto, zanim zwrócimy wartość łożyska, odwracamy ją, dodając do niej 180 stopni. Operator modulo (%) zapewnia, że wartości powyżej 360 ° są przenoszone na współrzędne kompasu. Na przykład, jeśli obliczysz namiar o 350 ° i dodasz do niego 180 °, otrzymamy 530 ° stopni. Jeśli zaczniesz od 0 ° i zmienisz około 530 °, skończysz na 170 °. Operator modulo spowoduje obliczenie namiaru z tą prawidłową wartością kompasu.

Loksodorma

Jak wspomniano wcześniej, czasami lepiej jest wybrać dłuższą ścieżkę stałego kursu, zwaną linią loksodromy, w porównaniu do ciągłego zmieniania pozycji w celu podążania ścieżką wielkiego koła. Linia rombu będzie dłuższa niż wielkie koło, a odległość od trasy wielkiego koła w dowolnym momencie nazywana jest błędem poprzecznym. Przekroczenie Atlantyku jest o 5% dłuższe, jeśli podążymy za rombem. Skrajny przykład przejścia ze Wschodniego Wybrzeża Stanów Zjednoczonych do Chin jest o około 30% dłuższy. Jednak tak duże kary są rzadko spotykane, ponieważ statki muszą zmienić kurs, aby uniknąć lądowania! To sprawia, że przykłady "w linii prostej" są nierealistyczne. Jeśli twoja gra dostarcza informacji nawigacyjnych każdemu, prócz pilotów, prawdopodobnie użyje loksodromy. Poniżej przedstawiono formuły stosowane do obliczania odległości i namiaru między dwiema współrzędnymi na linii rombowej. Najprostszym sposobem na rozpoczęcie jest spłaszczenie globu. W rzucie Mercatora linie rombu są proste. W rzeczywistości sprawia to, że graficzne rozwiązanie problemu jest bardzo proste. Używasz linijki. Matematyczne rzeczy stają się nieco bardziej skomplikowane. Następujące równanie daje Δϕ, co jest różnicą szerokości po uwzględnieniu, że rozciągnęliśmy je w celu spłaszczenia kuli:

$$\Delta\varphi = \ln[\tan(\mathrm{lat2}/2 + \pi/4) / \tan(\mathrm{lat1}/2 + \pi/4)]$$

Odległość między dwoma punktami na linii rombowej określa:

$$D = \sqrt{\Delta_{lat}^2 + q^2 * \Delta_{long}^2} * R$$

Zmienna q jest wartością, której formuła zależy od Δϕ. Jeśli Δϕ jest równe 0, oznacza to, że obliczony kurs będzie albo bezpośrednio na wschód, albo na zachód. W takim przypadku wartość pośrednia q wynosi:

q = cos (lat1)

jeśli Δϕ nie jest równe 0, wówczas:

q = Δlat / Δϕ

Możesz zobaczyć, że jeśli nie zostanie poprawnie zaimplementowany, bezpośredni kurs na wschód lub zachód spowoduje podział o 0. Na koniec, stałe położenie to:

$$\Theta_{\mathrm{rhumb}} = \mathrm{atan2}(\Delta_{\mathrm{long}}, \Delta\varphi)$$

W rzeczywistości istnieje nieskończona liczba lloksodormy, które doprowadzą nas do naszego punktu końcowego. Jednak ci, którzy są dłuższi, pokonają nasz glob lub spiralę na całym świecie, zanim uderzą w nasz punkt końcowy. W każdym razie najkrótszą trasą będzie ta, w której Δl_{ong} jest mniejsze niż 180 °. Powyższy jest zaimplementowany w Objective-C w następujący sposób:

```
float rhumbBearing (Coordinate2D startPoint, Coordinate2D endPoint) {

// Konwertuj lokalizację ze stopni na radiany

float lat1 = (M_PI / 180.) * startPoint.lat;

float lon1 = (M_PI / 180.) * startPoint.longi;

float lat2 = (M_PI / 180.) * endPoint.lat;
```

```
float lon2 = (M_PI / 180.) * endPoint.longi;

// Oblicz delty

float dLat = lat2 - lat1;

float dLon = lon2 - lon1;

// znajdź delta phi

float deltaPhi = log (tan (lat2 / 2 + (M_PI) / 4) / tan (lat1 + M_PI / 4))

float q = (deltaPhi == 0? dlat / deltaPhi: cos (lat1); // unika dzielenia przez 0

if (abs (dLon)> M_PI) {

dLon = (dLon> 0? - (2 * (M_PI-dLon) :( 2 * M_PI + dLon));

}

float D = sqrt (dLat * dat + q * q * dLon * dLo) * 6371;

float theta = atan2f (dLon, deltaPhi);

// teraz zamień na nagłówek kompasu

łożysko pływające = theta * (180 / M_PI); // radian do stopni

bearing = (bearing> 0? bearing: (360.0 + bearing)); // ustal zakres

bearing  return;

}
```

Jest kilka rzeczy, które warto podkreślić. Po pierwsze, używamy funkcji trójskładnikowej w wierszu skomentowanym przez "unika podziału przez 0", aby zająć się przypadkiem, gdy delta Phi jest równe 0. Jeśli jest równe 0, q jest ustawione na cos (lat1); jeśli nie, to jest ustawiony na dlat / deltaPhi. Instrukcja if bezpośrednio po gwarantuje, że jeśli dLon jest większy niż π (180 °), w związku z tym kładziemy na dłużej niż wymaganą loksodromę, a następnie powinniśmy poprawić wartość odpowiadająca najkrótszej trasie. Osiąga się to poprzez trójskładnik, który zapewnia, że dLon jest mniejszy niż π i nieujemny. Na koniec przeliczamy znormalizowaną odpowiedź radian na kierunek kompasu. Teraz, gdy masz już dobre pojęcie o tym, jak obliczyć pozycję i odległość w geograficznym układzie współrzędnych, możesz użyć wcześniejszych rozdziałów, aby określić inne wielkości, takie jak prędkość i przyspieszenie.