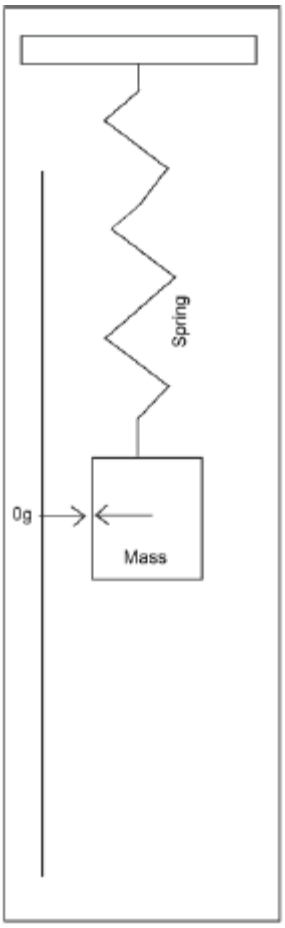
XXI. Akcelelrometry

Akcelerometry stanowią dobre wprowadzenie do klasy elementów elektronicznych zwanych układami mikroelektromechanicznymi (MEMS). Akcelerometr może być jednoosiowy, dwuosiowy lub trójosiowy. Oznacza to, ile różnych kierunków może jednocześnie mierzyć przyspieszenie. Większość urządzeń do gier ma akcelerometry trójosiowe. Jeśli chodzi o tworzenie gier, wartości przyspieszenia są zazwyczaj dostarczane do programu za pośrednictwem interfejsu API z jednostkami w wielokrotnościach g. Jeden g jest równy przyspieszeniu spowodowanemu grawitacją na Ziemi lub 9,8 m / s2. Udawajmy, że mamy akcelerometr jednoosiowy i orientujemy go tak, że oś jest skierowana w stronę środka ziemi. Zapisałby 1g. Teraz, jeśli będziemy podróżować z dala od jakiejkolwiek masy, tak że nie będzie grawitacji, akcelerometr odczyta 0. Jeśli wtedy przyśpieszymy tak, że w ciągu jednej sekundy od 0 m/ do 9,8 m/s, przyspieszeniomierz będzie odczytywać stały 1g podczas tej jednej sekundy przerwy. Rzeczywiście, niemożliwe jest odróżnienie przyspieszenia od grawitacji i przyspieszenia ze względu na zmianę prędkości. Prawdziwy ruch na ogół jest niestały. W zależności od celów aplikacji może być konieczne zastosowanie różnych funkcji wygładzania, takich jak filtry górnoprzepustowe lub dolnoprzepustowe. Sprowadza się to do cyfrowego przetwarzania sygnału, tematu, który pochłonął całe teksty. Ponadto wiele akcelerometrów ma metodę ustawiania szybkości odpytywania lub liczbę razy na sekundę, o którą program żąda aktualizacji z przyspieszeniomierza. Nazywa się to częstotliwością i jest podane w hercach (Hz). Ten parametr może być użyty do zwiększenia wydajności programu, gdy nie jest potrzebna dokładna rozdzielczość przyspieszenia w czasie. Po zaakceptowaniu danych wejściowych z akcelerometru lub innego przetwarzania sygnału musisz zaakceptować, że dane wejściowe nie będą przychodzić dokładnie wtedy, kiedy tego chcesz. Systemy operacyjne zwykle używane w grach - Windows, OS X, Linux - nie są środowiskiem w czasie rzeczywistym. Oznacza to, że chociaż ustawiasz stawkę za odpytywanie raz na sekundę, gwarantuje to tylko, że dane zostaną dostarczone nie wcześniej niż raz na sekundę. Jeśli coś rozprasza system operacyjny, na przykład pojawienie się pakietów w sieci, sygnał, który otrzymasz z akcelerometru, może być opóźniony.

Teoria przyspieszeniomierza

Sposób w jaki MEMS mierzy akcelerometry jest z zasady bardziej podstawowy niż ty mogą myśleć. Najważniejszym osiągnięciem jest miniaturyzacja technologii, dopóki nie zmieści się w chipie komputera! Aby jasno zilustrować podstawową zasadę, pokażemy najpierw jej mechanikę w makroskaliowej wersji znanej masy i wiosny. Powiedzmy zbudujesz coś podobnego do tego pokazanego na rysunku 21-1 i przeniesiesz je do windy w miejscu, w którym nie ma grawitacji. Za chwilę będziemy się martwić efektami grawitacji.



Jak widać, urządzenie składa się ze znanej masy na końcu sprężyny obok przyrząd pomiarowy. Gdy winda nie przyspiesza, masa jest na znaku 0. Kiedy winda przyspiesza w górę lub w dół, masa na końcu sprężyny opiera się temu przyspieszeniu i ma tendencję do pozostawania w spoczynku. To pierwsze prawo Newtona w działaniu. Obciążenie inercyjne powoduje rozciąganie lub ściskanie sprężyny. Jeśli winda przyspiesza do góry, masa spowoduje rozciągnięcie sprężyny w dół. Przypomnij sobie z Części 3, że siła działająca na sprężynę jest liniowo zależna od przesunięcia masy przez równanie:

$F_n = kd$

Możemy bezpośrednio zmierzyć przemieszczenie, d, abyśmy mogli określić siłę w tym kierunek, n. Ponieważ masa jest znana, voila!

$a_n = m/F_n$

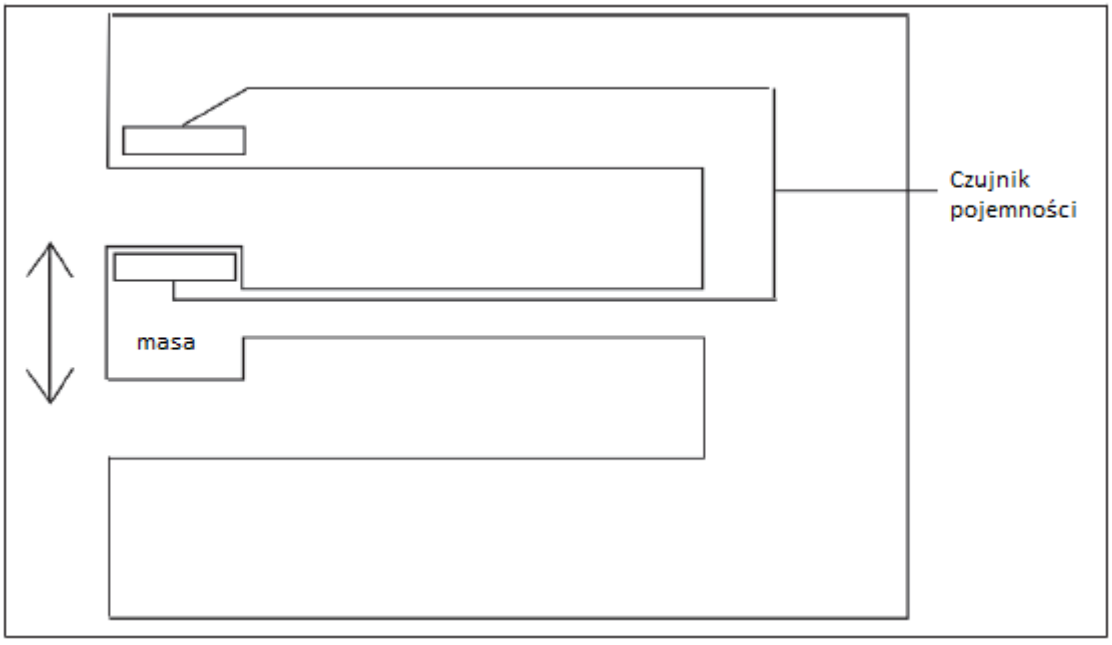
Na marginesie, fakt, że można stwierdzić, że przyspieszasz bez konieczności patrzenia na windę, sprawia, że jest to "nieinicjalny układ odniesienia", o którym mowa w części 2. Nie martw się, jeśli nie do końca to rozumiesz; nie jest ważne dla tego, o czym tu mówimy. Teraz, przenieśmy naszą windę z powrotem na ziemię. Przy tym samym urządzeniu masa nie będzie miała wartości 0, nawet jeśli winda nie przyspiesza, ponieważ siła grawitacji powoduje jej obniżenie. Wcześniej używaliśmy jednostek cali, które następnie konwertowaliśmy, aby wymusić i ostatecznie przyspieszyć. Jednak teraz mamy bezpośrednią miarę przyspieszenia z powodu grawitacji i możemy łatwo umieścić znak na miejscu, gdzie masa jest i nazwać ją 1g. Również możemy umieszczać znaki wzdłuż linijki w tych samych odstępach czasu. Teraz zwiększamy prędkość windy do 9,8 $m/s^{2.}$ Masa powinna przesunąć się w dół skali do 2g, a każdy stojący w windzie poczuje się dwukrotnie cięższy niż normalnie. Powiedzmy, że chcieliśmy przyspieszyć windę w dół na 9,8 m/s^2. Możemy to łatwo zrobić, po prostu zwalniając

hamulce i pozwalając grawitacji wykonać pracę. Teraz podczas spadania nie odczuwamy grawitacji w ogóle, prawda? Dzieje się tak dlatego, że przyspieszenie w dół anuluje przyspieszenie z powodu grawitacji. Masa powróci do 0, podobnie jak w kosmosie, z dala od ciał grawitacyjnych. Właśnie z tego powodu, a nie z braku grawitacji, astronauci unoszą się w powietrzu. Są w opadach na całej ziemi.

Wreszcie, jeśli przyśpieszymy windą w dół o 2g, masa przesunie się do -1g ,zaznacz na linijce. Jest tak dlatego, że przyspieszenie w dół jest teraz przytłaczające. Ci w windzie znajdowaliby się na suficie dokładnie tak, jak na ziemi! W rzeczywistości jednym z osiągnięć Einsteina było pokazanie, że nie można odróżnić grawitacji od inercyjnych przyspieszeń. Zostawimy szczegóły tego dla niezależnego badania i wrócimy do akcelerometrów w postaci MEMS.

Przyspieszeniomierze MEMS

Przyspieszeniomierze w mikroskali nie różnią się zbytnio od opisanej poprzednio maszyny, ale generalnie używają belki wspornikowej zamiast sprężyny. Aby śledzić więcej niż jedną oś, czasami trzy dyskretne akcelerometry są umieszczane poza płaszczyzną względem siebie. Alternatywnie, bardziej złożone modele wykorzystują elementy, które mogą wykrywać wszystkie trzy kierunki w jednym zintegrowanym czujniku. Generalnie dają lepsze wyniki. Jedyną istotną różnicą w stosunku do wyżej wymienionych przykładów, poza tym, że MEMS ma tysiące razy mniejszą skalę niż masa i sprężyna, jest sposób pomiaru ugięcia badanej masy. Istnieją trzy popularne metody stosowane w akcelerometrach. W przypadku większości urządzeń do gier, w których nie jest wymagana skrajna dokładność, odchylenie jest zwykle mierzone jako zmiana pojemności. Jest to trochę tak samo, jak działają pojemnościowe ekrany dotykowe, jak opisano w części 20, i pokazano na rysunku 21-2.



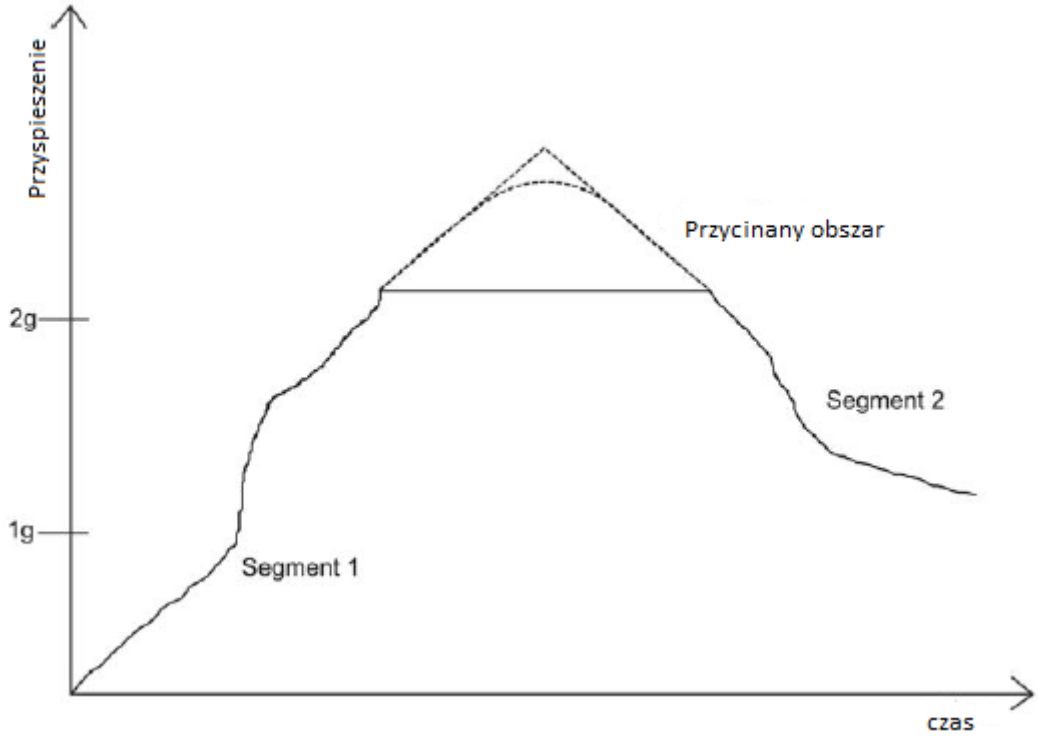
Strumień odchyla się pod wpływem zewnętrznych przyspieszeń masy próbnej i przynosi dwie naładowane płytki dalej lub bliżej siebie. To zmienia pojemność systemu. Ta zmiana może być następnie skalibrowana do narzuconego przyspieszenia. Inne metody obejmują integrację piezorezystora w samej wiązce tak, że ugięcie wiązki zmienia opór obwodu. Chociaż to ostatecznie daje lepsze wyniki, są one trudniejsze do wyprodukowania. Dla najbardziej wymagających zastosowań dostępne są akcelerometry wykorzystujące elementy piezoelektryczne oparte na kryształach kwarcu. Są one bardzo czułe nawet podczas zmian przyspieszenia o wysokiej częstotliwości, ale generalnie nie są wykorzystywane do wykrywania ruchu wejścia człowieka.

Powszechne akcelerometry

Aby pomóc Ci lepiej eksperymentować z akcelerometrami, zebraliśmy specyfikacje w przypadku kilku najpopularniejszych przyspieszeniomierzy używanych w czasie pisania. Limit 2g dla telefonów może powodować problemy podczas próby nagrywania ruchu. To ograniczenie zostanie omówione w dalszej części tego rozdziału. Większa gama kontrolerów Wii i Sony pokazuje, że są przeznaczone do gier, gdzie większe przyspieszenia są oczekiwane.

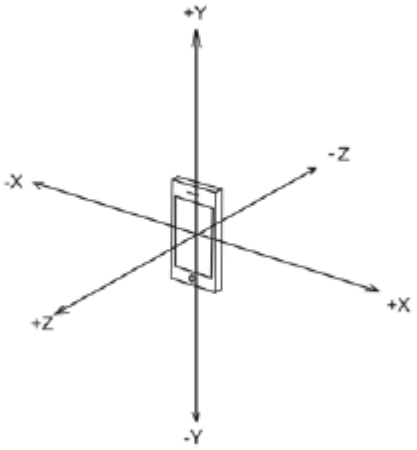
Przycinanie danych

Ludzkie ramię może łatwo przekroczyć zakres ± 2 g czujnika iPhone'a. Wartości zgłaszane przez API przekroczą w rzeczywistości 2 g do około 2,3 g. Dokładność tych wartości, które przekraczają specyfikację, jest nieznana. Bez względu na to, są prawdopodobnie co najmniej tak dokładne, jak opcja próbowania odtworzenia danych, więc w razie potrzeby mogą być użyte. Wszystkie wartości powyżej tego górnego limitu będą zgłaszane jako górny limit tak, że gdyby wykreślić wartości, wyglądałyby jak na rysunku 21-3.



Istnieje kilka różnych sposobów radzenia sobie z obcinaniem danych. Jedną z nich jest odrzucenie danych i powiadomienie użytkownika, że przekroczył dostępny zakres. Innym jest próba odtworzenia brakujących danych. Jeśli zapisujesz dane do późniejszego przetworzenia, możesz użyć zarówno segmentu 1, jak i segmentu 2, aby dopasować krzywą między punktem, w którym dane zaczęły być przycinane, a punktem, w którym ważne gromadzenie danych jest wznawiane. Jest to wysoce zależne od zastosowania, a krzywa użyta do dopasowania danych będzie musiała być dopasowana do wykonywanej czynności. Jeśli nagrywasz dane do późniejszego przetworzenia, możesz użyć zarówno segmentu 1, jak i segmentu 2, aby podać swoje dane. Jeśli próbujesz przetworzyć sygnał w czasie rzeczywistym, będziesz miał tylko segment 1 do pracy. Może to spowodować nieciągłość, gdy wznowione zostanie ważne gromadzenie danych, i będziesz musiał zdecydować, jak sobie z tym poradzić, biorąc pod uwagę szczegóły dotyczące tego, co robisz z danymi. Wykrywanie orientacji

Wykrywanie obrotu w trzech stopniach swobody oznacza wyczuwanie orientacji sztywnego ciała i jest złożonym problemem, którego nie można w pełni rozwiązać za pomocą tylko przyspieszeniomierzy. Pomyśl o trzymaniu urządzenia w pionie. Jeśli obrócisz urządzenie wokół osi opisanej przez wektor grawitacji, żaden z akcelerometrów nie zmierzy żadnej zmiany siły działającej na ich masy testowe. Nie możemy zmierzyć tego stopnia wolności. Aby to zrobić, musimy naprawić żyroskop do urządzenia, a nawet te napotykają problemy, gdy ciało może swobodnie obracać się wokół wszystkich trzech osi. Teraz omówmy, co możemy zrobić. Najpierw Rysunek 21-4



pokazuje układ współrzędnych, którego użyjemy; rzeczywisty użyty układ współrzędnych zostanie określony przez producenta urządzenia, dlatego należy sprawdzić jego dokumentację. Teraz, jeśli podejmiemy pewne założenia w oparciu o to, jak użytkownik będzie trzymał nasze urządzenie, możemy określić pewne orientacje "grubsze". Jest tu kilka rzeczy do zapamiętania. Po pierwsze, jeśli trzymasz telefon w tych orientacjach ręką, przyspieszeniomierz jest wystarczająco czuły, aby wykryć niewielkie odchylenia od rzeczywistej pionowej. Zastanawiamy się nad tymi "grubymi" orientacjami, tak aby te małe odchylenia były ignorowane.

Wyczuwając Przechyły

Chociaż nie możemy dokładnie określić, jaki kąt użytkownik trzyma telefon wokół wszystkich trzech osi, możemy wybrać jedną oś, zakładając, że jest skierowana w dół, a następnie znaleźć zmianę kąta od tego założenia w czasie. Na przykład, jeśli telefon leży na stole, średnie przyspieszenie w kierunku Z wynosi -1, aw pozostałych kierunkach 0. Nawet jeśli użytkownik obróci telefon, wartości pozostaną takie, jak wcześniej podano, a my nie wyczuwa tego obrotu. Jeśli jednak użytkownik podniesie jedną krawędź ze stołu - nazwijmy to przechylaniem - wtedy akcelerometr zarejestruje różne wartości. Część przyspieszenia wywołana grawitacją będzie działała na pozostałe dwie osie. Dzięki wyczuciu tej zmiany akcelerometr pozwoli nam określić, pod jakim kątem urządzenie jest przechylone.

Używanie Tilt do kontrolowania Sprite

Tutaj pokażemy, jak zaimplementować kod do prostej gry, która prosi użytkownika o przeniesienie awatara do celu poprzez przechylenie telefonu. Najpierw krótko pokażemy przykład określenia obrotu wokół pojedynczej osi. Załóżmy, że mamy akcelerometr obrócony pod dowolnym kątem, α, o to właśnie rozwiąże nasz algorytm. Jak już wcześniej wspomniano, akcelerometry generalnie zgłaszają wartości jako wielokrotności grawitacji bliskiej ziemi, g. W poniższym przykładzie zajmujemy się tylko wartościami osi X i Y, odpowiednio ax i ay. Gdyby urządzenie znajdowało się w pozycji "pionowej", wówczas ax byłby równy 0, a ay byłby równy 1. Po obróceniu urządzenia, zobaczylibyśmy różne wartości, które są związane z naszym kątem α przy użyciu funkcji arcus tangens. W tym przypadku, ponieważ funkcja pojedynczego argumentu zawarta w większości języków programowania nie

rozróżnia diametralnie przeciwnych kierunków, korzystne jest użycie funkcji dwuargumentowej. Odpowiedni kod C wygląda następująco:

```
#define PI 3.14159

float find2dAngle(void){

//LOCAL VARIABLES

float alpha,

double ax, ay;

//POLL ACCELEROMETER FOR ACCELERATIONS, API SPECIFIC

ax = getXacceleration();

ay = getYacceleration();

//FIND ANGLE

alpha = atan2(ay,ax);

if (alpha >= 0){

alpha = alpha * (180/PI);

else {

alpha = alpha * (-180/PI) + 180;

}

return alpha;

}
```

Jest to dość proste, ale jest kilka rzeczy do podkreślenia. Po pierwsze, w drodze który twój program uzyska wyniki z akcelerometru będzie się znacznie różnić platformy, więc zamknęliśmy ten specyficzny dla API kod w getXacceleration () funkcjonować. W rzeczywistości większość systemów operacyjnych będzie stale pobierać akcelerometr w osobnym wątku, więc będziesz musiał mieć logicznego operatora, który mówi obiektowi akcelerometru, kiedy rzeczywiście chcesz zobaczyć wartości przekazane do twojego programu. Przykładowy kod celu C dla akcelerometru w iPhonie zostanie pokazany później. Po drugie, zauważysz, że używamy instrukcji if, która zmienia radiany na stopnie w taki sposób, aby zwracać prawidłowe odpowiedzi 0 ° -360 °. Pozwala to uniknąć zwracania uwagi na znak, ponieważ atan2 zwraca tylko odpowiedzi od 0 ° do 180 °, używając wartości ujemnej do reprezentowania drugiej połowy zakresu. Na przykład wyjście o wartości 0 ° oznacza, że urządzenie jest pionowe, a wyjście o wartości 90 ° oznacza, że urządzenie obraca się o 90 ° w lewo, a sygnał wyjściowy o wartości 180 ° oznacza, że urządzenie jest odwrócone. Teraz rozszerzmy to na dwa wymiary. Dzięki temu dowiemy się nie tylko, jak daleko znajduje się telefon od pionu wokół jednej osi, ale także od jego nachylenia względem osi Y.

Dwa stopnie swobody

Teraz powiedzmy, że chcemy stworzyć grę, w której kontrolujemy duszka poruszającą się w świecie 2D. Użytkownik trzymałby urządzenie tak, jakby leżało na stole i patrzyło z góry. Następnie przechyli telefon z tej płaszczyzny, aby skłonić duszka do poruszania się w pożądanym kierunku. Frakcja

grawitacji, której akcelerometr obecnie doświadcza w kierunkach x i y, będzie wprowadzana do naszej symulacji. Przykład zostanie zademonstrowany przy użyciu kodu Objective-C dla iPhone'a, a my będziemy z wykorzystaniem architektury graficznej Qwartz2D. Jeśli nie znasz się na Objective-C, nie rób tego martw się - wyjaśnimy, co robimy na każdym etapie, i możesz przesłać ten kod do niezależnie od języka, w którym pracujesz. Pierwszym krokiem będzie ustawienie naszego akcelerometru. W tym przypadku zamierzamy go zainicjować w naszym pliku tiltViewController.m, abyśmy:

```
- (void) viewDidLoad

{

Akcelerometr UIAccelerometer * = [UIAccelerometer sharedAccelerometer];

accelerometer.delegate = self;

accelerometer.updateInterval = kPollingRate;

[super viewDidLoad];

}
```

Ważną koncepcją jest to, że zdefiniowaliśmy nazwę dla naszego obiektu przyspieszeniomierza, akcelerometru i ustawiliśmy jego właściwość updateInterval na kPollingRate. Ta stała została zdefiniowana w tiltViewController.h jako (1.0f / 60.0f), co odpowiada 60 Hz. Innymi słowy, mówi to systemowi operacyjnemu, aby aktualizował obiekt akcelerometru naszego programu 60 razy na sekundę. Również w tiltViewController.m, piszemy, co dzieje się, gdy obiekt akcelerometr zostanie zaktualizowany za pomocą funkcji didAccelerate: przyspieszeniomierza w następujący sposób:

```
- (void) akcelerometr: przyspieszeniomierz (UIAccelerometer *)

didAccelerate: przyspieszenie (UIAcceleration *) {

[(SpriteView *) self.view setAcceleration: przyspieszenie];

[(SpriteView *) rysowanie self.view];

}
```

Ta funkcja jest wywoływana za każdym razem, gdy obiekt przyspieszenia jest aktualizowany i robi dwie rzeczy. Najpierw pobiera dane przyspieszenia z przyspieszeniomierza i przekazuje je do klasy Sprite View, o czym porozmawiamy za chwilę. Następnie informuje SpriteView, aby kontynuował i przerysowywał siebie. Klasa SpriteView jest miejscem, gdzie odbywa się akcja i składa się z pliku nagłówkowego Sprite View.h, w którym definiujemy następujące zmienne globalne:

```
UImage * sprite
```

Wskaźnik do obrazu, który będzie używany do reprezentowania naszego duszka na ekranie.

```
currentPos
```

Pozycja na ekranie, na której chcemy narysować duszek.

```
prevPos
```

Poprzednia pozycja ikonki na ekranie. Wykorzystamy to, aby przekazać losowanie funkcji, której części ekranu trzeba przerysować.

Przyspieszenie UIAcceleration *

Specjalny typ danych Objective-C do przechowywania danych z przyspieszeniomierza.

CGFloat xVelocity i CGFloat yVelocity

Zmienne zmienne utrzymujące bieżącą prędkość odpowiednio w kierunku x i y.

CGFloat convertX i CGFloat convertY

Zmienne zmienne utrzymujące współczynniki konwersji wyników silnika fizyki w metrach do pikseli na podstawie założonego rozmiaru światowego.

Dodatkowo zdefiniowaliśmy następujące stałe globalne:

g

W pobliżu ziemskiej wartości grawitacji ustawionej na 9,8 m / s2. Spowoduje to konwersję wartości przyspieszeniomierza od g do m/s^2 do wykorzystania w obliczaniu prędkości. Można to również dostroić do reprezentacji. W pobliżu ziemskiej wartości grawitacji ustawionej na 9,8 m / s^2. Spowoduje to zamianę wartości przyspieszeniomierza z g na m/s^2 w celu obliczenia prędkości. Może to być również dostrojone, aby reprezentować arbitralne przyspieszenie, zamiast tylko używać siły grawitacji jako siły (na przykład procentu ciągu silnika odrzutowego).

kWorldHeight i kWorldWidth

Wartości te są używane, aby umożliwić programistom zmianę założonych wymiarów światowych. Wyższe wartości oznaczają, że każdy piksel to większa odległość w metrach. Świat zawsze będzie skalowany w celu dopasowania do ekranu, tak duży świat oznacza ikonka pojawi się poruszać wolniej (kilka pikseli na raz) dla danego przyspieszenia. Zauważ, że nasz obecny kod nie skaluje ikonki. Teraz pokażemy, w jaki sposób używamy tych zmiennych w SpriteView.m, aby przenieść nasz ikonkę na nasz ekran w wyniku wartości akcelerometru. Po pierwsze, mamy trochę inicjalizacji, która ma miejsce w initWithCoder: metodzie, która uruchamia się po wczytaniu widoku po raz pierwszy:

```
- (id) initWithCoder: (NSCoder *) koder {

if ((self = [super initWithCoder: coder])) {

self.sprite = [UIImage imageNamed: @ "sprite.png"];

self.currentPos = CGPointMake ((self.bounds.size.width / 2.0f) +

(sprite.size.width / 2.0f), (self.bounds.size.height /2.0f)+(sprite.size.height /2.0f));

xVelocity = 0.0f;

yVelcoity = 0.0f;

convertX = self.bounds.size.width / kWorldWidth;

convertY = self.bounds.size.height / kWorldHeight;

}

return self;

}
```

Większość z nich jest dość prosta. Opowiadamy naszemu programowi, gdzie znajduje się wybrany przez nas obraz ikonki, i ustawiamy jego początkową pozycję na środku ekranu. Ustawiamy również prędkość początkową na 0 w obu kierunkach. Następnie uruchamiamy zmienne convertX i convertY w oparciu o właściwość self.bounds.size, która określa granice widoku w pikselach. Pokażemy dokładnie, jak wpłynie to na nasz program później. Następnie napiszemy niestandardowy mutator dla zmiennej CurrentPos:

```
- (void) setCurrentPos: (CGPoint) newPos {

prevPos = currentPos;

currentPos = newPos;

if (currentPos.x <0) {

currentPos.x = 0;

xVelocity = 0.0f;

}

if (currentPos.y <0) {

currentPos.y = 0;

yVelcoity = 0.0f;

}

if (currentPos.x> self.bounds.size.width - sprite.size.width) {

currentPos.x = self.bounds.size.width - sprite.size.width;

xVelocity = 0.0f;

}

if (currentPos.y> self.bounds.size.height - sprite.size.height) {

currentPos.y = self.bounds.size.height - sprite.size.height;

yVelocity = 0.0f;

}

CGRect curSpriteRect = CGRectMake (currentPos.x, currentPos.y,

currentPos.x + sprite.size.width, currentPos.y + sprite.size.height);

CGRect prevSpriteRect = CGRectMake (prevPos.x, prevPos.y,

prevPos.x + sprite.size.width, currentPos.y + sprite.size.height);

[Self setNeedsDisplayInRect: CGRectUnion (curSpriteRect, prevSpriteRect)];

}
```

Jeśli nie jesteś obeznany z Objective-C, kiedy zdefiniujesz zmienną instancji klasy, automatycznie zdefiniujesz mutator, który po prostu zaktualizuje wartość zmiennej do wartości, którą przekazujesz. Jednak w poprzednim przykładzie nadpisujemy ten mutator, aby wykonać dodatkową pracę. Najpierw

ustawiamy zmienną prevPos na bieżącą pozycję ikonki, a następnie aktualizujemy currentPos wartością, którą podał mutator. Jednak nasz silnik fizyki nie będzie uwzględniał reakcji kolizji z granicami ekranu, więc sprawdzamy, czy ikonka osiągnęła krawędź ekranu. Jeśli tak, po prostu mówimy programowi, aby zostawił go na krawędzi i ustawił prędkość w tym kierunku na 0. Na koniec definiujemy kilka prostokątów na podstawie nowej pozycji ikonki i starej pozycji ikonki. Po połączeniu tych prostokątów razem, mówimy systemowi operacyjnemu, aby przerysował ekran w tym obszarze za pomocą metody setNeedDis playInRect :. Jak pewnie sobie przypominasz, nasz akcelerometr wywołuje metodę losowania za każdym razem, gdy jest aktualizowana, i to właśnie w tej metodzie umieścimy nasz silnik fizyki:

```
- (void)draw {

static NSDate *lastUpdateTime;

if (lastUpdateTime != nil) {

NSTimeInterval secondsSinceUpdate = -([lastUpdateTime

timeIntervalSinceNow]); //calculates interval in seconds from last update

//Calculate displacement

CGFloat deltaX = xVelocity * secondsSinceUpdate +

((acceleration.x*g*secondsSinceUpdate*secondsSinceUpdate)/2); // METERS

CGFloat deltaY = yVelocity * secondsSinceUpdate +

((acceleration.y*g*secondsSinceUpdate*secondsSinceUpdate)/2); // METERS

//Converts from meters to pixels based on defined World size

deltaX = deltaX * convertX;

deltaY = deltaY * convertY;

//Calculate new velocity at new current position

xVelocity = xVelocity + acceleration.x * g * secondsSinceUpdate; //assumes

acceleration was constant over last update interval

yVelocity = yVelocity - (acceleration.y * g * secondsSinceUpdate); //assumes

acceleration was constant over last update interval

//Mutate currentPos which will update screen

self.currentPos = CGPointMake(self.currentPos.x + deltaX,

self.currentPos.y + deltaY);

}

[lastUpdateTime release];

lastUpdateTime = [[NSDate alloc] init];

}
```

Wcześniej omawialiśmy problemy z czasem podczas pracy z danymi z akcelerometru. W takim przypadku Objective-C bardzo ułatwia uzyskanie właściwego czasu w sekundach. Najpierw definiujemy zmienną statyczną lastUpdateTime jako typ NSDate. Ten typ ma wbudowaną funkcję, która daje przedział czasu w sekundach od teraz, który przypisujemy zmiennej NSTime Interval. Przechodząc do ostatnich dwóch linii, po prostu aktualizujemy ostatni czas aktualizacji, zwalniając i reinicjując zmienną. Ponieważ jest statyczny, pozostanie nawet po powrocie funkcji. Jeśli używasz języka niższego poziomu, być może będziesz musiał napisać własną funkcję timeIntervalSinceNow, która bierze pod uwagę określoną częstotliwość zegara w systemie. Teraz, gdy mamy czas w sekundach, możemy obliczyć naszą nową pozycję. Przypomnijmy z części 2:

$s_2 = s_1 + v_1 t + (a t^2) / 2$

które przestawiliśmy na:

$\Delta s = s_2 - s_1 = v_1 t + (a t^2) / 2$

To zostanie zaprogramowane jako:

CGFloat deltaX = xVelocity * secondsSinceUpdate + ((acceleration.x * g * secondsSinceUpdate * secondsSinceUpdate) / 2); // METRY

Następnie przeliczamy to przemieszczenie w metrach na przemieszczenie w pikselach, stosując odpowiedni stosunek do wielkości naszego świata. Zanim będziemy mogli przejść dalej, musimy obliczyć naszą nową prędkość w naszej nowej pozycji. Ponownie zakładamy przyspieszenie jako stałą w okresie aktualizacji i przywołując:

$v_2 = v_1 + a \Delta t$

z części 2 możemy rozwiązać nowy xVelocity z:

xVelocity = xVelocity + acceleration.x * g * secondsSinceUpdate;

Jak widać z pełnego opisu metody, kod kierunku y jest podobny. Na koniec wywołujemy mutator currentPos, aby ustawić nową pozycję na podstawie zmiany przemieszczeń. Przypomnij sobie, że jest to niestandardowy mutator, który informuje system operacyjny o konieczności aktualizacji wyświetlacza. Po zakończeniu procesu rysowania przyspieszeniomierz oczekuje 1/60 sekundy, a następnie wywołuje ją ponownie. Możesz rozszerzyć ten program, dodając tarcie, odporność na płyny i kolizje z granicami ekranu, używając metod opisanych w innych rozdziałach tego tekstu.