*Sebastian Worek*

# Box2D – dwa wymiary fizyki

Box2D jest silnikiem do symulacji fizyki dwuwymiarowej. Jest lekki, prosty w wykorzystaniu, efektywny, łatwo przenośny i darmowy, także do zastosowań komercyjnych. Biblioteka została napisana w języku C++ przez Erina Catto i jest obecnie dostępna na prawie wszystkich liczących się systemach operacyjnych (Windows, MacOS, Linux, Android czy iOS). Została także przepisana przez różnych autorów na praktycznie wszystkie popularne języki programowania, jak C#, Flash, Java czy Python. Niniejszy artykuł opiera się na wersji 2.3.0 wspomnianej biblioteki.

## WSTĘP

Rozpoczynając pracę z biblioteką do symulacji fizyki, warto najpierw zapoznać się z przyjętymi przez jej twórcę wymaganiami dotyczącymi formatu danych, jakich wymaga silnik do poprawnej symulacji, i w jakim formacie dane są z silnika zwracane. Kwestia istotna, gdyż Box2D jest silnikiem tylko i wyłącznie do symulacji fizyki, dlatego aby cokolwiek zobaczyć na ekranie, musimy napisać własną implementację rysującą poszczególne kształty lub skorzystać z innego gotowego rozwiązania, na przykład z biblioteki do grafiki 2d SFML, co niekoniecznie pokryje się z formatem danych biblioteki. Wysłanie do silnika nieprawidłowych danych, na przykład wysłanie obrotu w stopniach zamiast w radianach, z których korzysta Box2D, spowoduje nieprawidłowe zachowanie (niezgodne z oczekiwanym) symulowanego obiektu.

### *Piksele i metry*

Pierwsza ważna kwestia to przyjęcie wspólnej dla wszystkich obiektów, jakie będą tworzone wielkości obiektów – oczywiście wielkości umownych. Box2D został przygotowany do operowania w metrach (nie pikselach), co oznacza, że prędkość podajemy w metrach na sekundę, a wagę obiektu w kilogramach. Należy pamiętać, aby trzymać się tej ustalonej koncepcji dla wszystkich obiektów. Musimy także ustalić współczynnik przyjęty do obliczania wielkości naszych obiektów. Pozostaje nam do ustalenia, ile wynosi metr w przeliczeniu na piksele, bo na nich operuje grafika 2D widoczna na ekranie naszego monitora. Zastosowanie przelicznika jeden piksel równa się jeden metr nie byłby dobry. Dla obiektu wielkości 64 pikseli na ekranie (czyli obiektu dość małego w stosunku do rozdzielczości ekranu choćby 800 na 600, a przecież wiele gier pracuje w rozdzielczości 1920 na 1080) będzie to 64 metrowy kolos do obliczania przez Box2D (przeliczając 3 metry na piętro rzeczywistego budynku, daje to 20 piętro + parter). Do symulacji ruchomych ciał zaleca się (nie jest to oczywiście warunek konieczny) używanie obiektów o wielkości od 0.1 do 10 metrów. Jeżeli ustalimy, że 64 piksele są równe jeden metr, to kod takiego przelicznika będzie prezentował się następująco:

```
#define piksele_na_metr 64.0f
#define metr_na_piksele (1.0f/piksele_na_metr)
// skrócone nazwy
#define PNM piksele_na_metr
#define MNP metry_na_piksele

//Zamiana danych do przeliczania dla Box2D
Box2dPosPlayer = b2Vec2(posPlayer.x * MNP, posPlayer.y * MNP);

//biblioteka wykonuje obliczenia, po czym zwracany wynik
//przekazujemy do funkcji rysującej tym razem wykonujemy
//konwersję w drugą stronę
posPlayer = vec2(Box2dPosPlayer.y * PNM, Box2dPosPlayer.y *PNM);
```

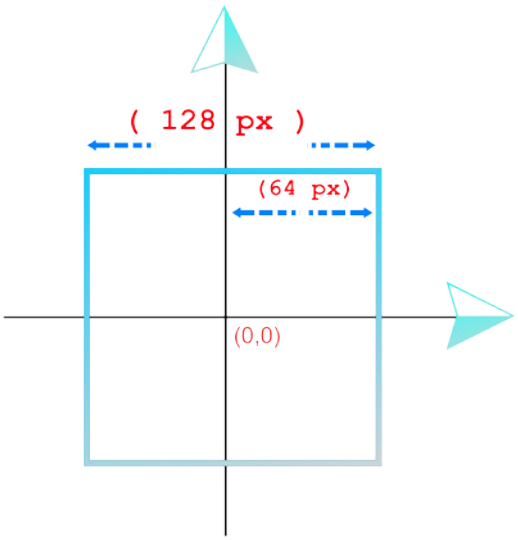
#### Obroty

Box2D używa radianów do obliczania obrotu obiektów. Jeżeli nasza aplikacja wykorzystuje do reprezentowania obrotów dla grafiki stopnie, to także w tym przypadku musimy sami zadbać o zamianę w obie strony, czyli zamienić stopnie na radiany i radiany na stopnie. Przelicznik jest bardzo prosty i sprowadza się do dwóch prostych operacji:

```
Stopnie = (180.0 * (radiany / PI));
Radiany = (PI * (stopnie / 180.0));
```

### *Środek obiektu*

Biblioteka oczekuje rzeczywistego środka obiektu, w innym przypadku symulacja będzie daleka od oczekiwanej. Tu również policzenie nie sprawia żadnej trudności. Jeżeli wielkość naszego obiektu w pikselach wynosi 128 na 128 piksele, a jego środek jest w punkcie 0,0 (Rysunek 1 przedstawia opisywaną sytuację), to dla box2D rzeczywistym środkiem obiektu jest połowa wysokości i połowa szerokości, czyli 64 na 64 (nie 0,0). I taką wartość należy przekazać do biblioteki dla wybranego obiektu.




*Rysunek 1. Rzeczywisty środek obiektu graficznego stworzonego na potrzeby np. naszej gry wynosi 0,0, natomiast rozmiar wynosi 128 pikseli. Do silnika Box2D musimy podać rzeczywisty środek obiektu, czyli połowę wysokości i połowę szerokości, która wynosi w tym przypadku (64,64)*

## BOX2D, CZYLI FIZYKA 2D WEDŁUG ERINA CATTO

Oczywiście autor nie zmienia praw fizyki, a jedynie ułatwia jej wykorzystanie (poprzez wygodny interfejs dostarczany przez bibliotekę) w grach czy symulacjach fizycznych, do których biblioteka została stworzona. Mimo pewnego uproszczenia modelu w celu zapobiegania niedokładności (skończona pre-