**Opis działania fizyki w grze**

**High-Speed Rocket Rave**

**w klasie “PlayerController”**

**W klasie “PlayerController” znajduje się między innymi sterowanie graczem (rakietą), a więc również fizyka, którą chcę omówić.**

****

**Na początku potrzebowałem wektora “direction”, a więc różnicy między wektorem “rotTarget”, czyli kierunkiem, a właściwie przyszłym przesunięciem względem osi x (tutaj w zależności od ruchu gracza w prawo lub w lewo) i y oraz aktualną pozycją gracza (transform.position).**

****

**Następnie potrzebowałem kąta (angle), który powstaje między osią x, a współrzędnymi przesunięcia (direction.y, direction.x). Do tego użyłem funkcji Mathf.Atan2(). Oczywiście wartość musiałem zmienić na stopnie, gdyż domyślnie funkcja zwraca wartość w radianach.**

****

**Kolejno uzyskałem rotacje rakiety (rotation), używając funkcji Quaternion.AngleAxis(), wcześniej uzyskanego kąta, oraz osi z, która odpowiada w Unity 2D za rotacje. Użyłam Vector3.foward, ponieważ to to samo co Vector3(0, 0, 1), gdzie trzecia wartość odpowiada za oś z, której jak wcześniej pisałem potrzebowałem do rotacji obiektu.**

****

**Na koniec, aby uzyskać gładki ruch obrotu rakiety, użyłem funkcji Quaternion.Slerp(), która generuje płynną interpolację między aktualnym obrotem rakiety (transform.rotation), a kolejno uzyskaną rotacją (rotation) oraz szybkością przejścia tych rotacji (rotationSpeed) pomnożoną razy Time.deltaTime (czas w sekundach od ukończenia ostatniej klatki), aby zapewnić stałą, płynna rotajcę, niezlaęnie od ilości klatek na urządzeniu, które uruchamia grę.**