Rigidbody là gì?
Rigidbody có tác dụng điều khiển vị trí của vật thể thông qua hệ thống mô phỏng vật lý của Unity, không cần bất kỳ script nào nó cũng có thể di chuyển và xử lý hướng khi va chạm (nếu có Collider)

Đồng thời chúng ta cũng có thể tác động vào Rigidbody bằng script như điều chỉnh vận tốc (velocity), vận tốc xoay (angular velocity),…

Lưu ý khi sử dụng Rigidbody(2D), chúng ta nên hiểu và sử dụng hàm FixedUpdate() thay vì Update()

Rigidbody2D và Transform
Khi một GameObject được gắn Rigidbody2D thì nó (Rigidbody2D) sẽ “ghi đè” lên Transform tức là cập nhật position và rotation theo vật lý của Rigidbody2D. Vì thế đã xài tới Rigidbody/2D thì không nên di chuyển hay xoay GameObject bằng Transform như kiểu:

```
transform.position += direction * moveSpeed * Time.deltaTime;
```

```
transform.Translate(direction * moveSpeed * Time.deltaTime);
```

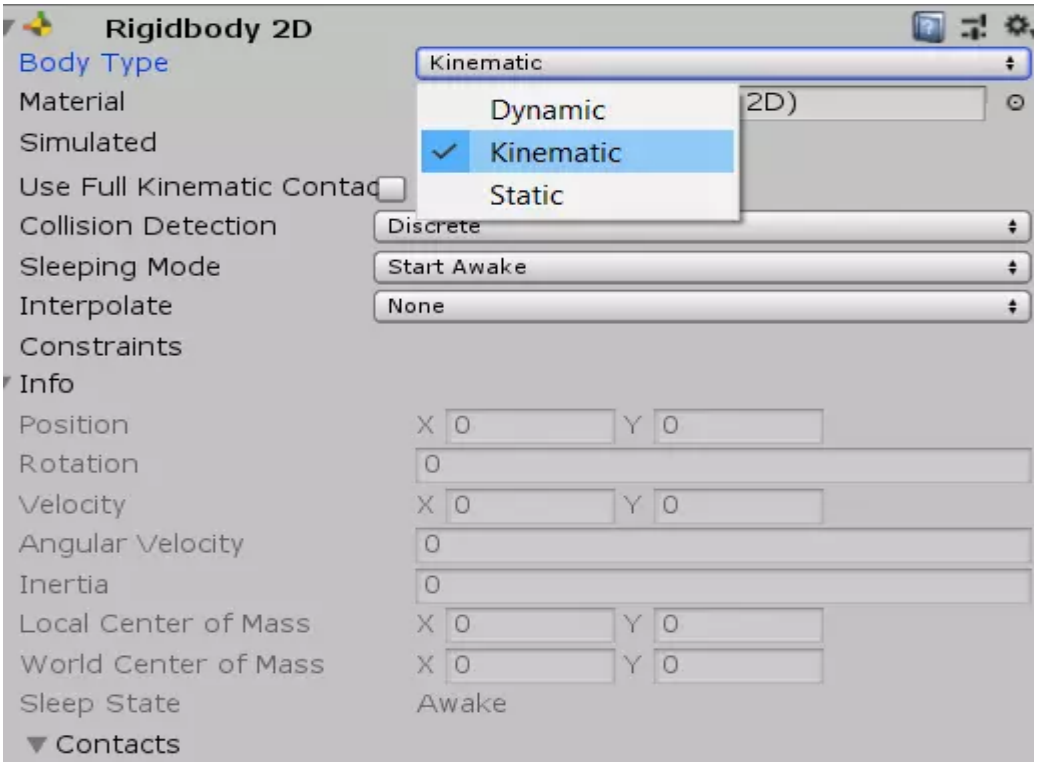
```
transform.eulerAngles += rotateDirection * rotateSpeed * Time.deltaTime;
```

“Sao lại không, mình vẫn xài được bình thường khi có Rigidbody mà?”

Tất nhiên chúng ta vẫn sử dụng được, nhưng không nên, logic vật lý sẽ sai, có thể các vật thể sẽ xuyên qua nhau, thông, chồng lên nhau,…

Kinematic Body
Kinematic Body Type là loại Rigidbody không chịu ảnh hưởng bởi các loại lực như: lực tác động bởi các vật thể trong game, tác động lực bằng script, trọng lực. Cũng có thể xem khối

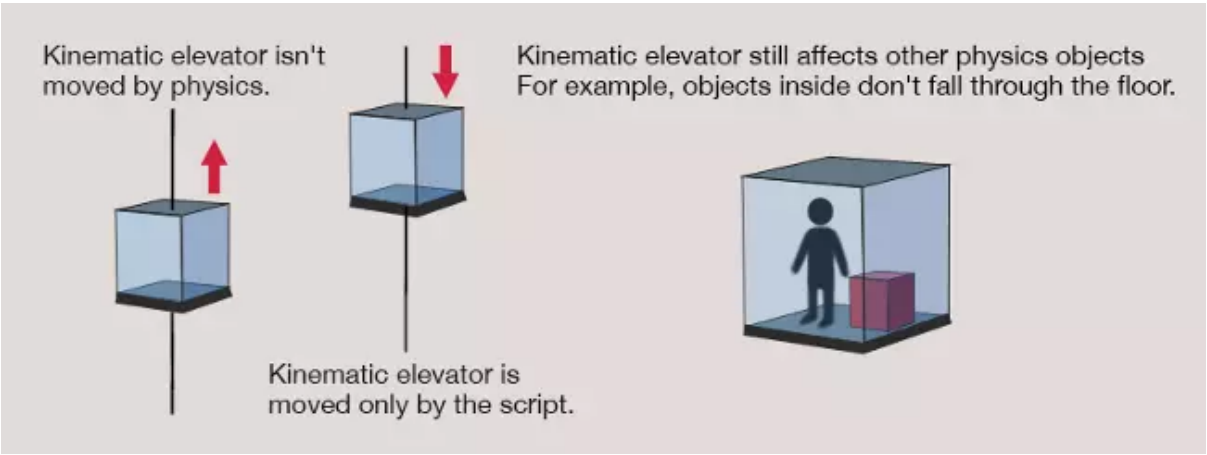
lượng của Kinematic Body là vô cực và không chịu ảnh hưởng của trọng lực.



Mặc dù không chịu tác động bởi lực, nhưng Kinematic Body vẫn có thể có vận tốc (velocity) và nếu không có tác động từ script thì vận tốc đó là không đổi bởi không có lực gì tác động.

“Vậy sao không sử dụng Dynamic Rigidbody? Có full chức năng không tốt hơn hay sao?”

Tùy mục đích mình sử dụng, giả sử Kinematic là cái thang máy chẳng hạn, nó đi lên/ xuống không theo vật lý trong game mà vẫn có va chạm, sử dụng Kinematic Body sẽ hợp lý hơn. Đồng thời Kinematic Body về performance hiển nhiên sẽ tốt hơn Dynamic Rigidbody.

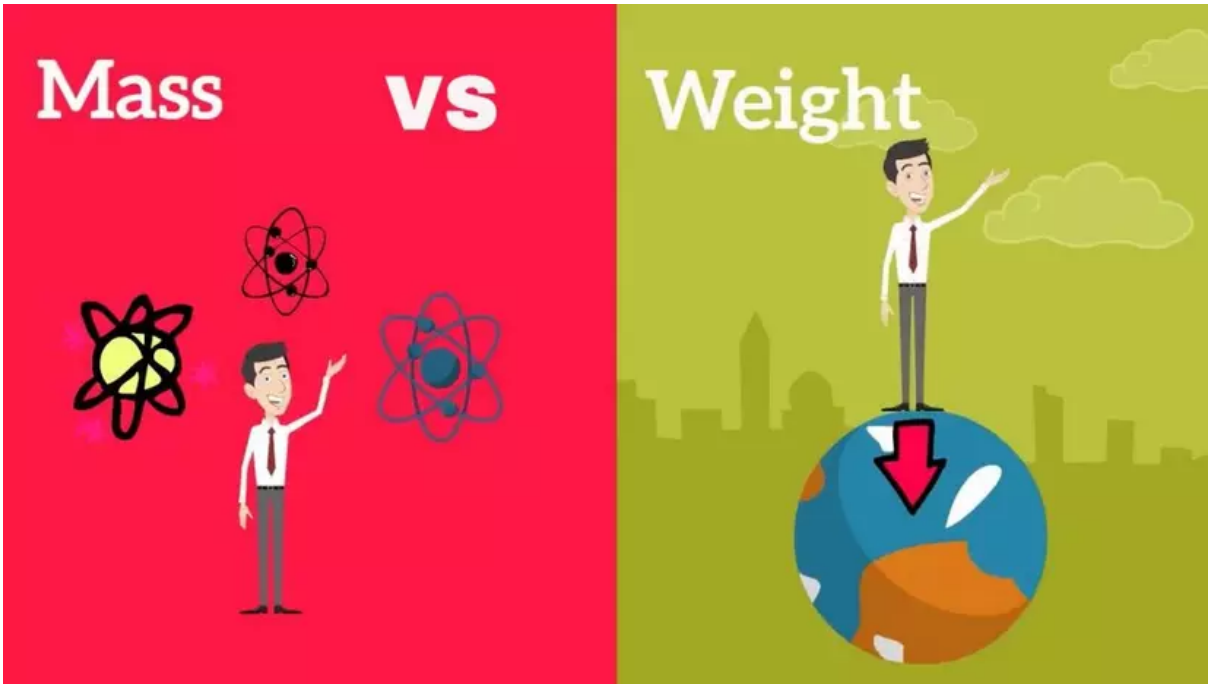


Mass và Auto Mass

“Quá dễ, Mass là cân nặng của một vật thể chứ gì nữa, còn Auto Mass thì Unity dựa vào độ “to” của vật thể như sprite hay collider gì đấy rồi tính toán sao cho phù hợp”

“Mass có phải là cân nặng không? Sao không đặt là Weight?”

Phần này mình nói thêm cho bài có chút thú vị, nghiêng về tiếng Anh. Mass thật ra không phải là cân nặng, tưởng tượng các phụ tùng “stuffs“, cơ quan trong cơ thể bạn là mass, còn cân nặng của bạn là 60kg.
60kg là con số “không xác định”, cùng một thời điểm, giả sử bạn ở trên mặt đất là 60kg, nhưng ở trên mặt trăng thì con số này sẽ giảm còn 2-3kg. Mass thì không thay đổi, các phụ tùng cơ quan vẫn ở trong cơ thể bạn.



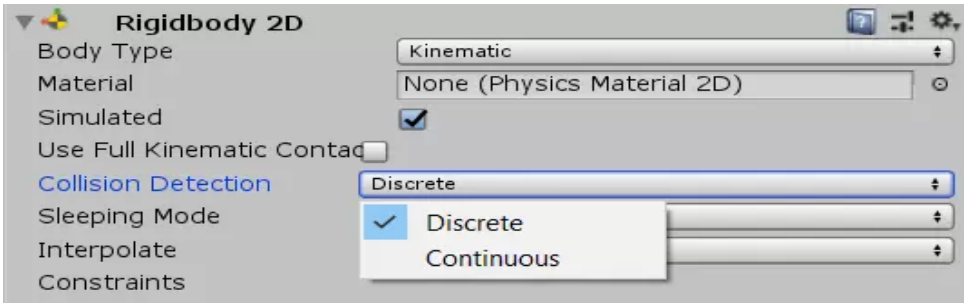
Mass và Weight
“Nếu mass đã khác thì Auto Mass chắc không đơn giản như trên đâu hé?”

Nó vẫn sẽ đơn giản nhưng mà khác ở trên một chút, Auto Mass hoạt động có dính dáng tới một nhân vật nữa là “Collider/2D”.

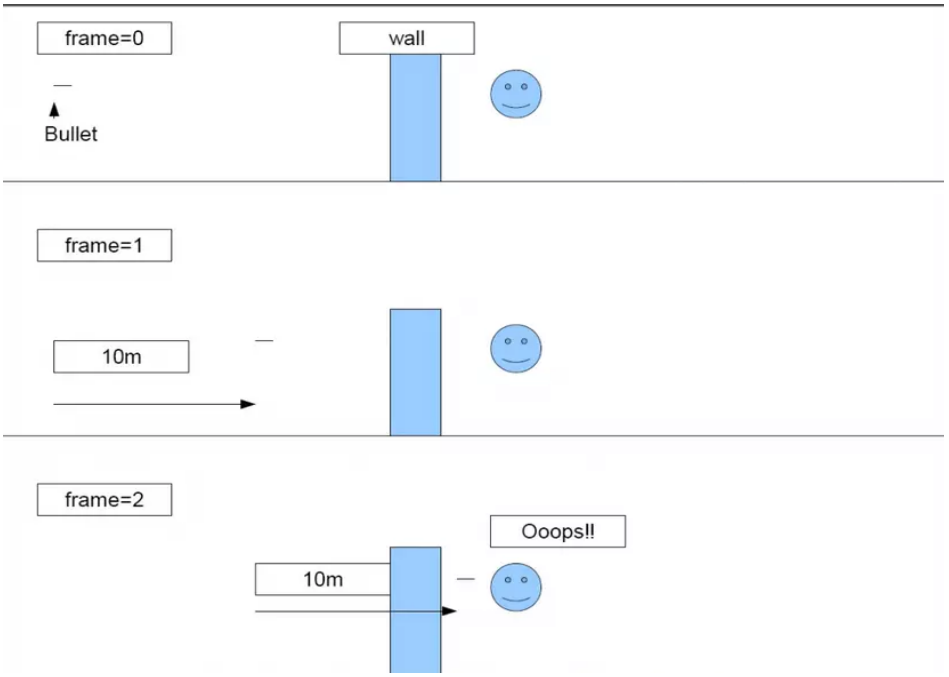
Khi bạn tick vào checkbox Auto Mass của Rigidbody2D, các Collider liên kết với nó sẽ mở rộng thêm một thuộc tính gọi là Density (độ dày đặc). Như vậy Auto Mass sẽ tính Mass dựa trên 2 thông số là Area (độ “to”) và Density (độ dày đặc)

Collision Detection
Đây là một thuộc tính mà mình ban đầu không để ý cho lắm và ít khi sử dụng, tuy nhiên thuộc tính này rất hữu ích trong vật lý của Unity.

Nó có 2 loại: Discrete (rời rạc) và Continuous (liên tục)



Ưu điểm	Khuyết điểm	
Discrete	Ít tốn tài nguyên, performance tốt hơn so với Continuous	Khi một object A di chuyển đủ nhanh, nó có thể xuyên qua hoặc chồng lên object B nào đó.
Continuous	Xử lý được trường hợp object di chuyển nhanh hoặc fixedDeltaTime lớn. Một thuật toán tương tự có thể tham khảo là SweptAABB, anh em có thể đọc thêm	Performance thấp hơn Discrete



Sleeping Mode
Cũng như Collision Detection, mình cũng ít sử dụng thuộc tính này bởi mặc định của mode này là Start Awake đã khá là hợp lý.

Khi một Rigidbody di chuyển hoặc xoay với tốc độ chậm hơn so với mức tiêu chuẩn “rất nhỏ” được quy định trước, engine vật lý sẽ giả định rằng rigidbody này tạm ngừng lại và tiến vào Sleeping Mode, nghĩa là nó sẽ không di chuyển hay xoay trở lại cho tới khi nhận va chạm (hoặc lực) tác động. Thuộc tính này nhằm tối ưu tài nguyên sử dụng cho Rigidbody.

“Sleeping Mode tức là mode ngủ hả? Nếu một Rigidbody2D ngủ thì làm sao đánh thức nó dậy?”

Có 4 cách mà mình research được:

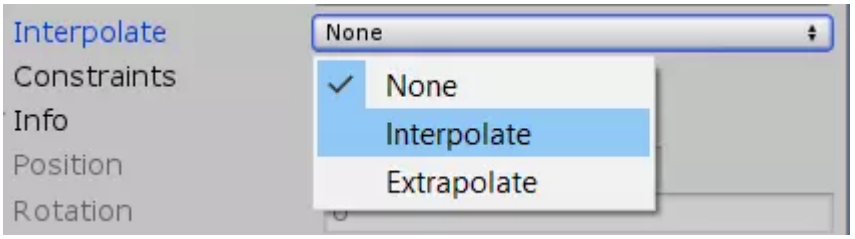
Rigidbody2Ds khác va chạm với nó (phải diễn ra sự va chạm)
Sleeping Rigidbody2D này được kết nối với các Joint2Ds đang di chuyển
Khi sửa các thuộc tính của Rigidbody2D
Khi tạo lực (AddForce)
Interpolate
Okay, đây là thuộc tính đủ phức tạp để kết thúc bài viết này.

Interpolate hiểu đơn giản là phép suy vị trí của vật thể trong trường hợp FixedUpdate chạy chậm hơn so với Update. À thì có thể không đơn giản lắm…

Giải thích theo cách khác, khi Graphic muốn vẽ hình lên screen nhưng ông FixedUpdate chậm chạp mãi chưa kịp cập nhật vị trí cho các vật thể do vật thể có sử dụng vật lý, thì Unity cung cấp một thuộc tínhInterpolate hỗ trợ làm mượt chuyển động.

“Khoannn, FixedUpdate thì liên quan gì tới vật lý vậy?”

Tất cả các tính toán vật lý, cập nhật sẽ được thực thi ngay sau các FixedUpdate. Để hiểu rõ hơn về FixedUpdate thì anh em đọc Execution of Order trong Unity nhé



Hoạt động	
None	Object sẽ bị giật vị trí khi render nếu fixedDeltaTime lớn
Interpolate	Phép nội suy, Graphic sẽ vẽ GameObject tại vị trí dựa trên các vị trí của GameObject này ở frame trư
Extrapolate	Phép ngoại suy, Graphic vẽ GameObject tại vị trí ước tính của GameObject tại frame tiếp theo