# Week 9 Allie Arnott

Repository Link: https://github.com/Xiomaraze/Physics-Exploration.git

Q1: What does the element do?

Q2: What is an example of what it can be used for?

Q3: How could you incorporate it into your previous Space Shooter assignment?

Q4: How useful does the concept seem on a scale of 1-10?

## Rigidbody 2D Constraints:

Q1: Allows objects and associated rigidbodies to be confined to specific axes of motion and rotation. Example: a ridigbody constrained to the Y axis will only be able to move "up" and "down" along the Y axis, never left and right along the X or Z. Also only be able to rotate along the Y axis keeping it's orientation to the X and Z axes static.

Q2: Allows for controlled exacting motion for objects like along a track, or a 3rd person camera panning. By disallowing the other axes of movements the likelihood of the object getting lost in the scene or looking the wrong way is much smaller. Can also be used to help control direction during collision events.

Q3: By constraining the player ship to the X and Y axes, it would allow for simpler rotation and movement to their sprite and object without possibility of the player sprite/object being lost on the Z axis.

Q4: On a scale of 1-10, this is easily a 10, great concept. Takes a lot of the back end calculations out of human hands and removes a lot of possibility of calculation errors.

## Rigidbody Type 2D

Q1: Controls how the object reacts to physics present within the scene, like gravity, forces, and player control. Dynamic may be affected by gravity and other dynamic and kinematic rigidbodies. It's collider is affected by all other rigidbodies. Kinematic is not affected by gravity, or other static and kinematic rigidbodies. It can be moved by repositioning, or velocity methods, and will only collide with dynamic rigidbodies unless explicitly set to collide with all rigidbody 2D types. Static is a non-moving rigidbody, it will not be affected by gravity or other rigidbodies, but will affect dynamic type of rigidbody, this is explicitly designed to never be moved and should not be moved while a scene is active.

Q2: Allows for physics based movement and object contact types, creating a sense of mass in the world. Example: A building will have a static rigidbody in most cases and will not move no matter how much the player pushes with their dynamic rigidbody.

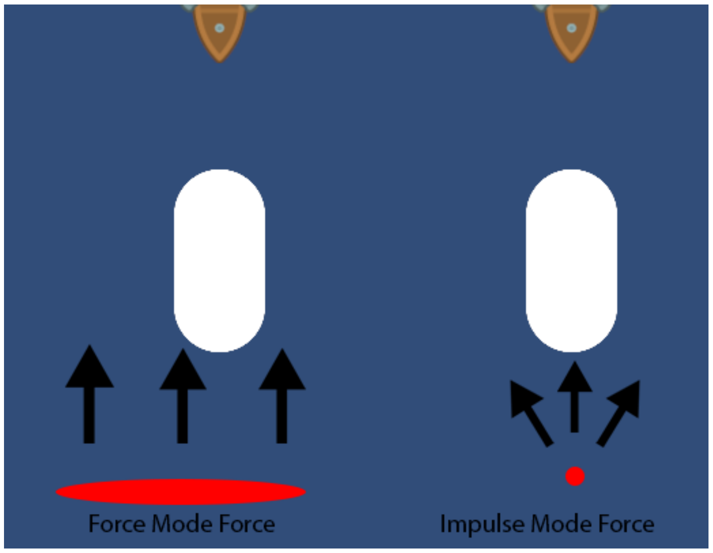
Q3: The Asteroids in the space shooter could have been handled by a kinematic rigidbody, allowing for collider interaction based movement instead of human programmed coordinate based movement.

Q4: On a scale of 1-10 this is a 9, very useful but can cause some frustrations if attempting to explicitly reposition objects via scene editor instead of using forces while the scene is active.

## Rigidbody 2D Add Force At Position

Used references and test script at https://docs.unity3d.com/ScriptReference/ForceMode2D.html

Q1: Applies a force to a rigidbody based on a world space position specified. This allows the force to be external and affect the rigidbody in a way that may result in rotation of the object. This may be applied in 2 methods, 1 with constant consistent force increasing velocity from the position specified, and the other being a burst of force from the specified position increasing velocity.



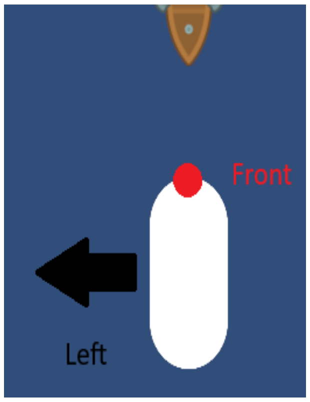
Q2: Allows for consistent speed applied to objects, or an impulse type force which is a possible exponential type force applied to an object. Example: A wind applying force vs An explosion applying force.

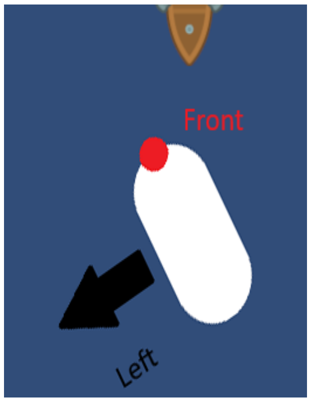
Q3: Could have been used to handle enemy movement in a consistent chase aspect, or even redirecting the asteroids via shielding as with my proposed additions to the project.

Q4: On a scale of 1-10 this is a 7, useful but seems overly complicated to control movement for the player. Would acceleration and deceleration HAVE to be handled by the impulse type force? Or is there a way to handle it with the force mode type force.

## Rigidbody 2D Add Relative Force

Q1: As far as I can tell this just applies force in the same way as the previous one. It says something about applying it in the rotated coordinate space of the object, but I am not sure what that means exactly. Is rotated coordinate space the object based on it's rotation? Or based on the orientation of the camera?





I understand now, it refers to the object in it's rotation. I was using a specific vector like (0.1, 2.5) before as the force, instead of the vector2.left type of stuff. As shown in the image, left means left to the object, not the world space / camera.

Q2: Player movement using rotation without having to calculate angle based on world space.

Q3: Would have been way more useful in coding how the player's ship moves, how the asteroids move, how to handle their reflection off the shields.

Q4: Usefulness is decided as 30 out of 10, I love this now that I understand it. Math is not my forte when it comes to quaternions and rotations, and I tend to give myself headaches. By keeping the movement relative to the object's rotation rather than worldspace it makes so much more sense to me.