## SubmarineBehavior.cs

The purpose of this revised SubmarineBehavior script is to control the basic movements of the sub in a more physically realistic manner. To improve the realism and real-time tuning capability of the Submarine, I've created a number of variables:

```
private Rigidbody rb;

private float BallastTanksFull;
private float MotorSpeed;
private float RudderDirection;

public float RudderForce;
public float PropellerForce;
public float BallastForce;
```

**TIP:** Don't forget to initialize the values of your private variables in `Start()`.

### Use Keyboard Input to Control Motion

We will need to check if the user is pressing forward/backward keys. If they are, then we change the speed of the motor.

```
if(Input.GetKey(KeyCode.UpArrow))
{
    MotorSpeed = MotorSpeed + 0.01f;
    if(MotorSpeed > 1.0f) {
        MotorSpeed = 1.0f;
    }
}
```

We will need to check if the user is pressing directional keys. If they are, then we rotate the rudder.

```
if (Input.GetKey(KeyCode.RightArrow))
{
    RudderDirection = RudderDirection + 0.01f;
    if(RudderDirection > 1.0f) {
        RudderDirection = 1.0f;
    }
}
```

**Tip:** Think carefully about what the values of MotorSpeed and RudderDirection should be when going the opposite direction!

NEXT PAGE →

## Adding Some Elevation Control

Be sure to create a variable that keeps track of the percentage of the Ballast Tanks that are filled. Then you can use the keyboard to empty or fill the tanks.

```
if (Input.GetKey(KeyCode.Space))
{
    BallastTanksFull = BallastTanksFull + 0.01f;
    if (BallastTanksFull > 1.0f) {
        BallastTanksFull = 1.0f;
    }
}
```

Use the current state of the tanks to determine the upward force of the Submarine.

**TIP:** Don't forget to process your user input in **`Update()`**.

## Making the Sub Move

Now that our sub has sum control memory, we can use the physics engine to make it move.

```
void ProcessPhysics() {
    rb.AddRelativeForce (new Vector3 (0, BallastTanksFull * BallastForce, 0), ForceMode.Force);
    rb.AddRelativeTorque (0, RudderForce, 0, ForceMode.Force);
    rb.AddRelativeForce (new Vector3 (0, 0, PropellerForce * MotorSpeed), ForceMode.Force);
}
```

**TIP:** Don't forget to call your physics method in **`FixedUpdate()`**. As a general rule, physics calculations shouldn't be placed in the **`Update()`** method.