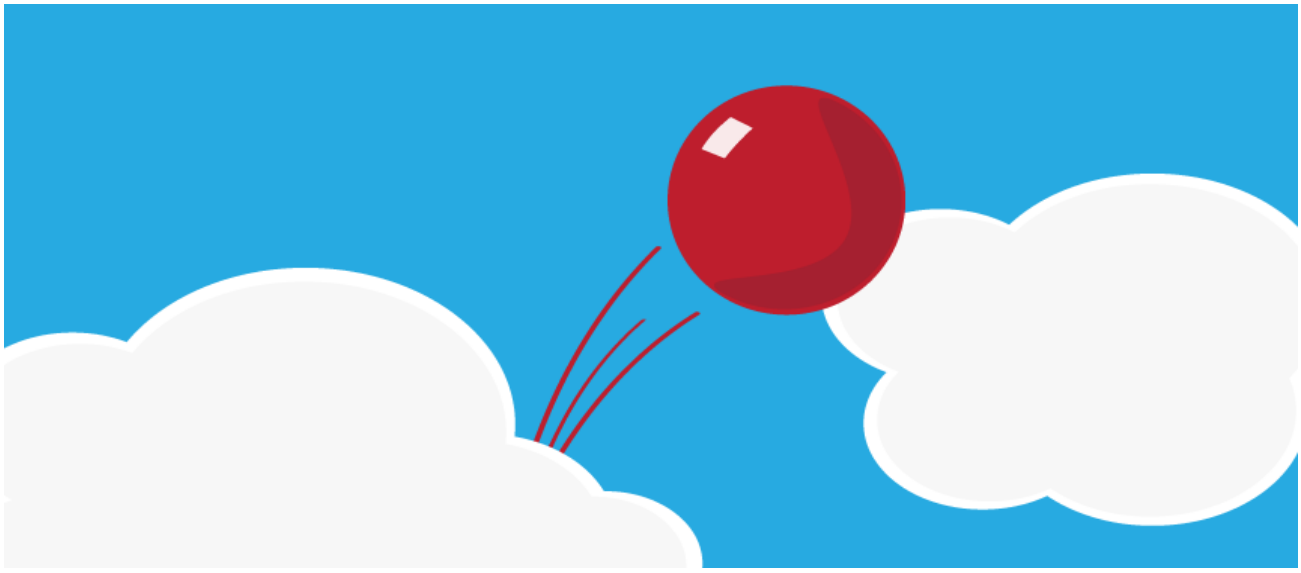

Propulsion Physics v1.3

1. Getting Started

Hello and thanks for purchasing Propulsion Physics! Prefabs are included with the tool to let you quickly start trying out the propulsion scripts. To try this out in your scene, drag on the Propulsion Pad prefab and then drag on the Propulsion Target prefab into your scene. Click on the Propulsion Pad and then drag and drop the Propulsion Target from the hierarchy pane onto the Target field on the Propulsion Pad.

In the scene view, you should see an arrow connecting the Pad to the Target. That's the trajectory the object will take when it gets propelled off the Propulsion Pad. Now let's propel a Rigidbody! Create a sphere and add a Rigidbody to it. Next add the tsg_PropelRigidBody script to the sphere. This script tells the Propulsion Pad how to handle propelling this object when they touch. Now position the sphere over the Pad and let it fall. You should see it fly into the air and eventually land on the target.

That's it! Now you can start using Propulsion Physics in your game and enjoy watching objects get shot around. :)

2. Settings

- **Target** is the Transform object that the propulsion script will propel the object to. The included Propulsion Target Prefab is a fast way to set targets for your propulsions.

- **Reach Time** lets you set how long the propelled object should take to reach the target based on the editor's gravity.
- **Trajectory Color** does what it says. :) This is really handy when you have multiple propulsions in a scene.
- **Show Trajectory** lets you either show or hide the trajectory in the scene view.
- **Vertical Only Min** tells the propulsion script that when the target position gets within this distance from the pad, it should calculate a vertical only trajectory.

3. Extending

It's easy to extend Propulsion Physics to work with other types of objects. Say you want a character controller to be propelled, just implement the tsg_IPropelBehavior interface and define a React method. This React method will be called once your custom script touches the Propulsion Pad trigger. You can then define how to handle what happens to the character. To see an example of this, check out the tsg_PropelRigidBody script.

Another quick example of extending is to have a sound play when an object hits a pad. Create a new propulsion physics class that inherits from tsg_PropulsionPhysics and override the PropelObject method. Upon a successful propel, tell the Audio Source to play:

```
using UnityEngine;
using System.Collections;

public class CustomPropulsionPhysics : tsg_PropulsionPhysics
{
   protected override bool PropelObject(GameObject propelObject, Vector3 velocity)
   {
       if(base.PropelObject(propelObject, velocity))
       {
           audio.Play();
       }

       return true;
   }
}
```

4. Support

If you have any questions about this tool or suggestions that could make it better, please let me know! You can contact me at caleb@polycrime.com.

5. Thanks

I'd like to thank AnomalusUndrdog and kOrc from the Unity forums for helping make this code possible. :)

6. Changes

v1.3 -

- removed audio source from Propulsion Pad prefab to be more agnostic
- added virtual methods to PropulsionPhysics class to allow easy extending
- simplified how the propulsion pad target is rendered
- documentation update

v1.2 - documentation update

v1.1 - description update

v1.0 - initial release