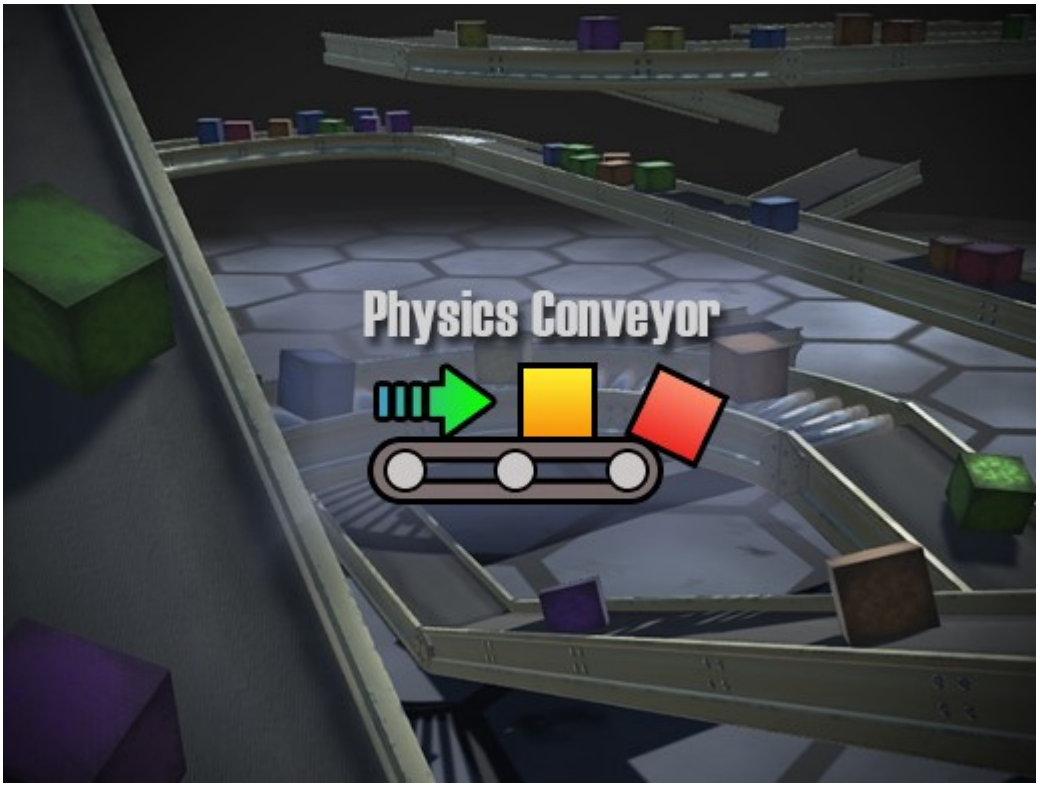


An Asset for Unity by Nathan Brower

Package Contents

Physics Conveyor is a professional modular physics based conveyor system that realistically moves rigidbodies along its surface in the way you would expect. This game ready package contains everything you need to get moving!

Features:

- Easy to use, just place the prefabs in your scene!

- 21 prefabs, including thin variations and decoration.

- 7 PBR materials with texture sizes up to 2048x2048.

- Conveyor Belts and Rollers.

- Belt script supports any shader for custom solutions.

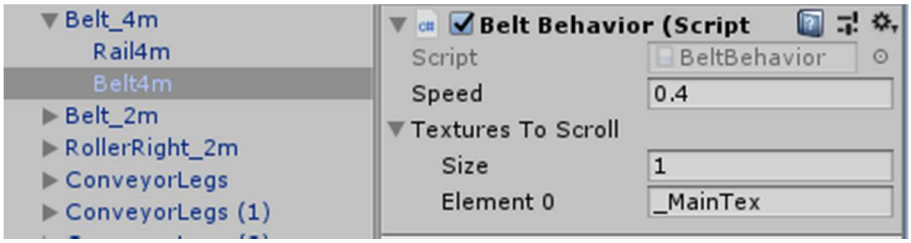
- Clean and well commented code.

- Script for automatically handling rigidbody interpolation modes.

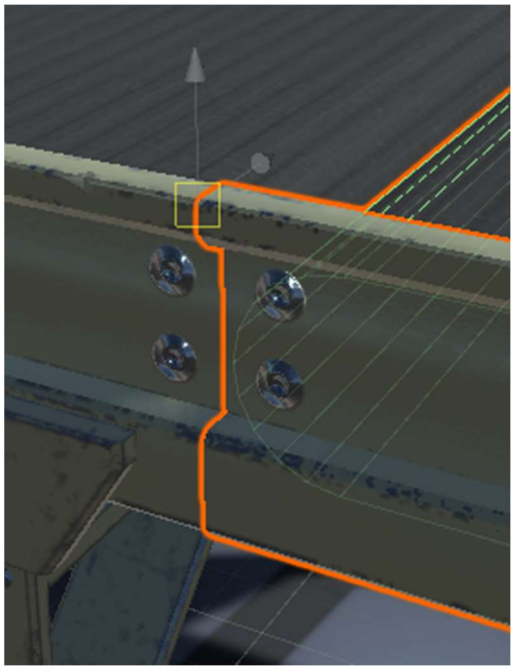
- Demo scene showing how you can easily change the conveyor speed at runtime.

Getting Started

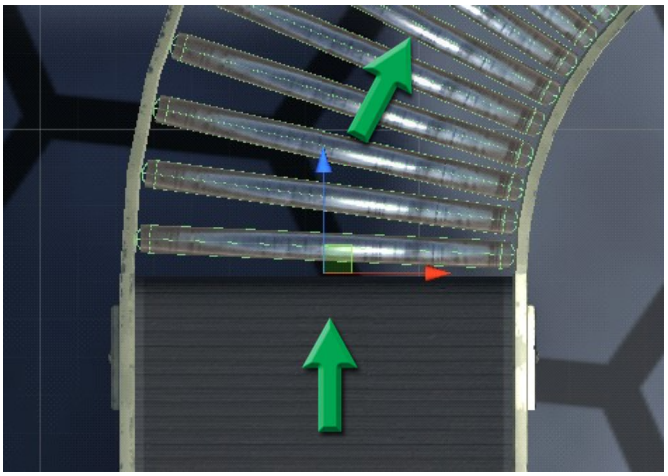
To get started using Physics Conveyor, simply place an included prefab in your scene, place some rigidbodies above it, and press play. The conveyor should be working! To change the speed, just alter the speed variable on the Belt Behavior (or Roller Behavior for rollers) component found nested in the prefab.



To easily connect up conveyor prefabs, hold 'V' before moving the GameObject to use vertex snapping.



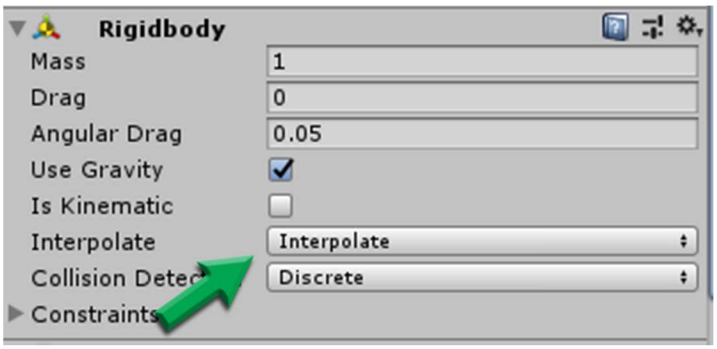
Conveyors will move rigidbodies along its positive local Z axis. Giving a conveyor a negative speed will cause it to move in the opposite (Z negative) direction.

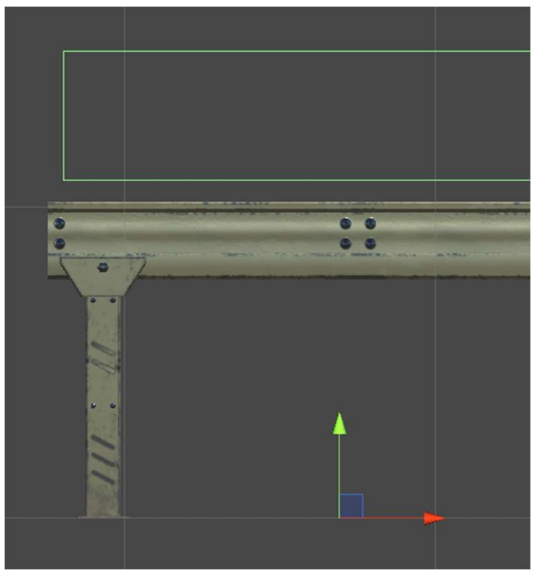


The 'Textures to Scroll' array is the list of texture names used by the shader that should be scrolled by the Belt Behavior component to give the illusion of a turning conveyor belt. If you are using the built in Standard Shader (which all of the included materials do) this array only needs to contain '_MainTex', and the rest of the textures will scroll for you. This is not a very well documented feature of the Standard Shader, but at the time of writing, this is all that needs to be done. If you are using a different shader, you will need to find the internal names of the textures used by that shader that you wish to have scroll and add those to the array. (e.g. _NormalTex, _MetallicTex, etc.)

Rigidbody Interpolation

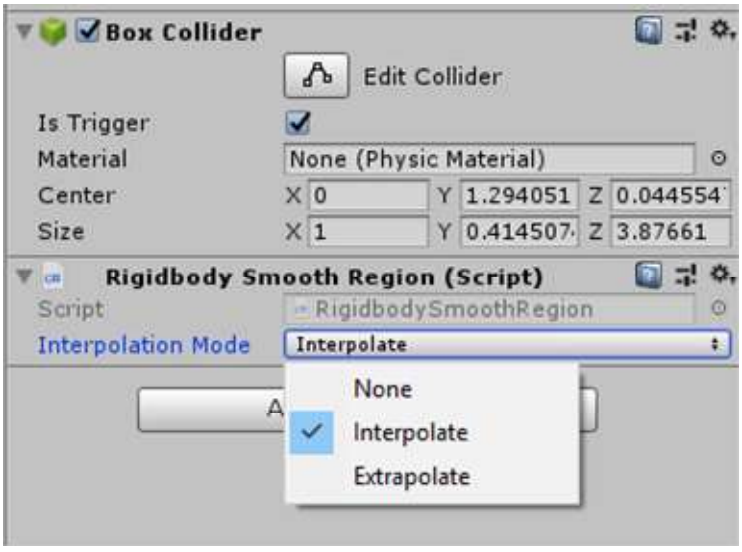
For the best looking results, rigidbodies intended to ride along a conveyor belt should have their Interpolation mode set to Interpolate (or Extrapolate). Leaving it set to 'None' will cause the rigidbody to appear to jitter slightly while riding on a belt when viewed up close. Setting all of the rigidbodies in your game that might find itself riding a conveyor belt to Interpolate is a fine and easy solution. However, if your project cannot afford the extra overhead involved with interpolating all rigidbodies in your game, this package contains a solution.





For an example of this problem and an implementation of the solution, see the included 'SmoothRegionDemo' scene. In this scene, you will find two conveyor lines. One line has a couple of Rigidbody Smooth Regions, while the other does not. All boxes being spawned in this scene have their Interpolation mode set to none. If you look closely, the bodies riding on the conveyor belts with no smooth regions appear to slightly jitter. For each line, the roller segment does not cause the bodies to jitter, so the smooth regions are only used for belts.

For various reasons, these Rigidbody Smooth Regions are not included in any conveyor prefab. Instead, they must be added manually. Doing so is easy. Simply add a GameObject, give it a Box Collider, and a Rigidbody Smooth Region component. Edit the collider so it is positioned above (but not touching!) the conveyor belt as shown on the left, and make sure you check 'Is Trigger'. Set the Interpolation Mode to Interpolate (or Extrapolate) and you're done!



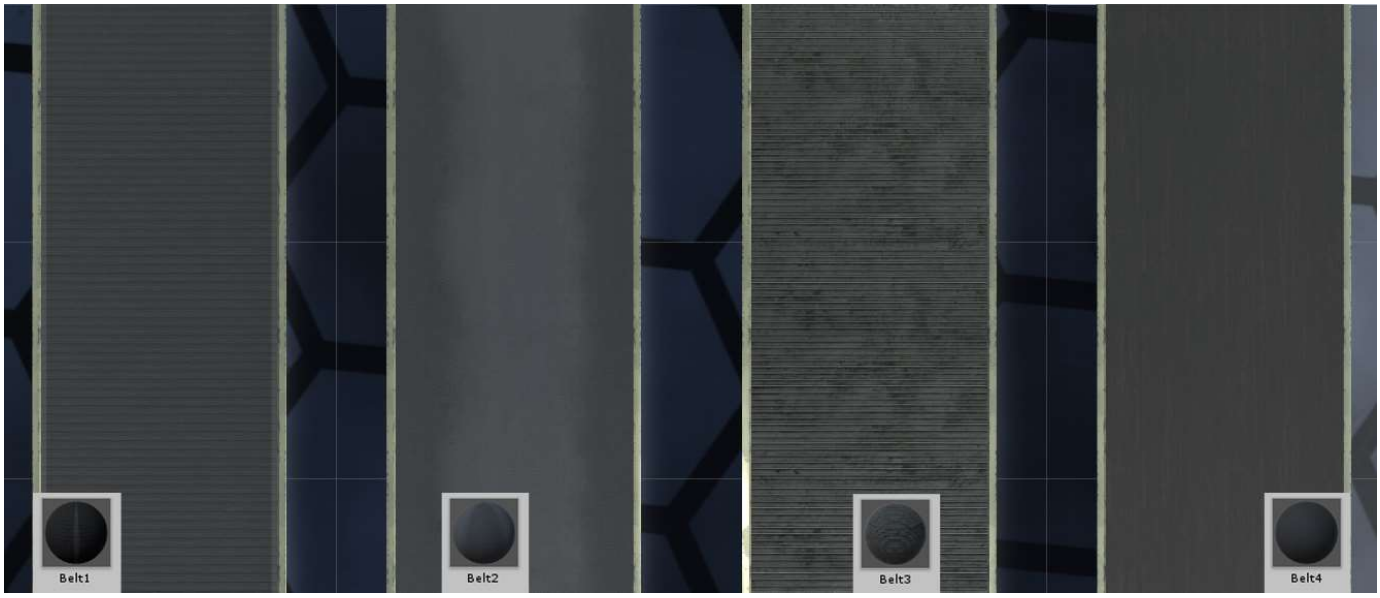
Now, any rigidbody passed into this region will have its previous Interpolation Mode cached and set to whatever is specified. Upon exit, Interpolation Mode will be reverted.

Important note! For straight belt segments with more than one conveyor connected in a line, you should only use one smooth region for the entire

straightaway. If a rigidbody enters an adjacent region before fully exiting the one it is already in, the newly entered region will cache the rigidbody's interpolation mode as not necessarily what it started with. While not game breaking, this does defeat the purpose of using the smooth regions in the first place.

Material Variations

Included in this package are four material variations for the belt.



Adjusting Speed at Runtime

Both BeltBehavior.cs and RollerBehavior.cs extend the base class ConveyorBehavior.cs. ConveyorBehavior.cs contains the public float 'speed'. Adjusting this will seamlessly adjust the speed of the belt or roller at runtime. See the main demo scene and DemoManager.cs for an example as to how this could be implemented in your game. In that demo, each conveyor line that should be controlled separately (main, right fork, and left fork) are all represented by a parent GameObject with each conveyor GameObject as its child. The Demo Manager finds each of the Conveyor Behaviors in each of the conveyor lines' children, and adds them to an array. This array is then looped through and each Conveyor Behavior is updated with the UI Slider value.

```
foreach (ConveyorBehavior behavior in mainConveyors)
    behavior.speed = mainValue;

foreach (ConveyorBehavior behavior in rightConveyors)
    behavior.speed = rightValue;

foreach (ConveyorBehavior behavior in leftConveyors)
    behavior.speed = leftValue;
```

There are of course many ways this can be done, and I hope this example gets you started in implementing something that suits your game!

Contact Information

Thank you for your support, and I hope this package is useful to you. If you have any questions or comments, please don't hesitate to contact me.

Cheers!

Nathan Brower

Email – NDBrower@Gmail.com

ArtStation - https://www.artstation.com/nathanbrower

One or more textures on this 3D model have been created with photographs from Textures.com. These photographs may not be redistributed by default; please visit www.textures.com for more information.