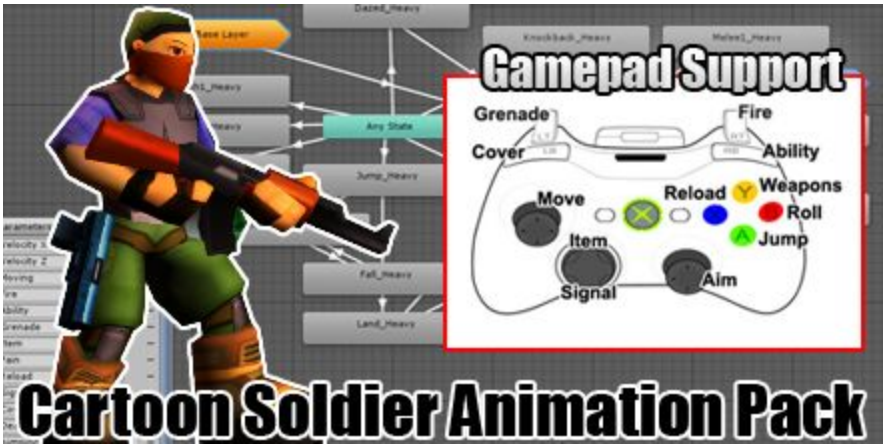


Version 2.6 - Last Updated *2015-11-7* by Kristopher Peterson

Hey, first off thanks for purchasing and using this pack!

Update 2.6 - Switched Rig type to Generic. Added transition from Run to PushPull. Tweaked Firing blend.

Update 2.5 - PushPull addition. Default key is “H”. Hold while in yellow volume. Currently no implementation of attaching or moving any objects, or doing anything with the weapons.

Update 2.4 - Shotgun Fire animation added as alternate to Rifle Fire. Not currently set to play, but can be enabled through the Soldier Animator as a replacement for the Rifle Fire, or as a new weapon type.

Update 2.3 - Swimming has been added, and the code has been majorly re-worked to accommodate this. Input has been split out of its previous functions and included only in Update(), whereas before it was in a couple different places. Character rotation has also been moved to its own function since it was being used in multiple points.

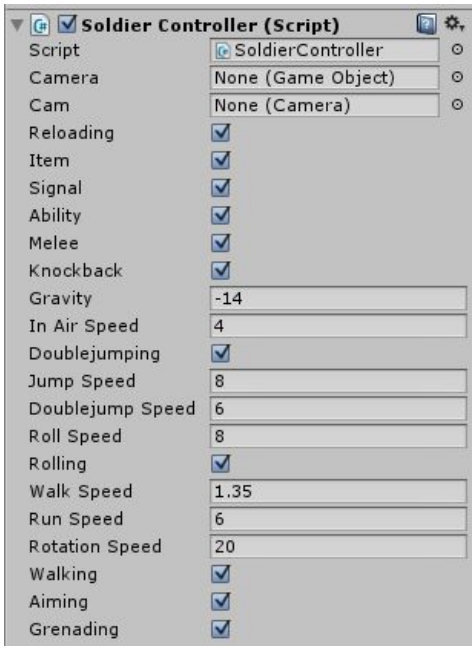
Update 2.2 - Minor update to fix some Materials warnings and other misc fixes.

Update 2.1 - This update adds improved support for animated or rigid body driven platforms and some minor code changes and clean up. Note that the gravity function is now under FixedUpdate().

Update 2 - This update adds significant changes from version 1.8, including a modified

skeleton for Humanoid rig type with neck, shoulder, and toe bones, Avatar setup, multiple AvatarMasks for upper body and head control, a full Mecanim Animation Controller with all 104 animations included, and a custom SoldierController script with Xbox360 controller support for all the movement and actions.

There is also a Blobshadow in the latest version that scales upon distance from the ground, and also aligns to the normal of the surface below the soldier, If you are using dynamic shadows or some other form of lighting, you can simply delete the script from the character, and remove the Blobshadow object from the hierarchy.



Before attempting to use the pack, you must first ensure that the tags and inputs are correctly defined. There is an included **InputManager.asset** and **TagManager.asset** included in the **ProjectSettings.zip** file that contains all the settings. Copy these to your ProjectSettings folder.

All of the actions are toggle-able via the inspector, and the common attributes are visible for adjustment. In the script, the different sub control methods such as movement, aiming, jumping, and cover are broken up into their own functions and then included in Update() for easy modification, or removal if you don't plan on using them in your game.

With the 4 current weapon types, you can either replace them with your own weapons, or easily add more by correlating them to a weaponType, and then adjusting this code to continue incrementing:

```
if (weaponType == 5)
                {
                        //enables pistol, disables other weapons
                        pistol.SetActive(true);
                        rifle.SetActive(false);
                        launcher.SetActive(false);
                        heavy.SetActive(false);
                        StartCoroutine( COSwitchWeapon("Weapon", 1));
                        weaponType = 1;
                }
```

All the other animations have their own methods so that you can easily insert your own functions in for your game:

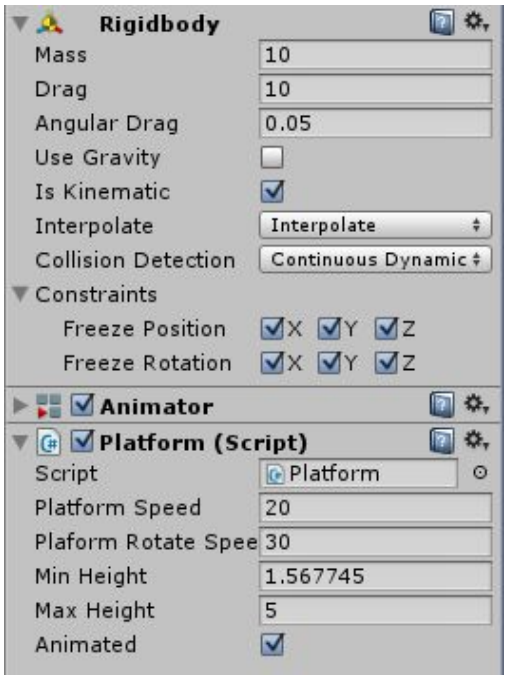
```
void Fire()
	{
		StartCoroutine( COPlayOneShot("Fire") );
	}

void Grenade()
	{
		StartCoroutine( COGrenade() );
		isGrenading = true;
	}

void Item()
	{
		StartCoroutine( COPlayOneShot("Item") );
	}
```

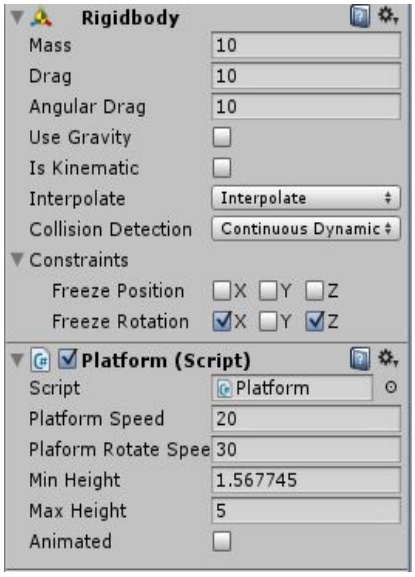
Platform.cs

In order to interact with a platform, you'll need to attach Platform.cs. If the platform is animated with keyframes, you can check 'Animated' and leave the the other settings in the script. If 'Animated' make sure the Rigidbody is marked 'Is Kinematic' and all the Constraints for Freeze Position and Freeze Rotation are checked. Setting 'Interpolate' or 'Continuous Dynamic' is recommended if your player or platforms move at a high rate of speed. The Mass, Drag, and Angular Drag should not matter if it is animated.




```
	void Update()
	{
		if (animated)
		{
			velocity = ((transform.position -
previous)) / Time.deltaTime;
			previous = transform.position;
		}
```

The Vector3 'velocity' is what is calculated to approximate the platforms rigidbody velocity if it is animated, and then that value is passed to the SoldierController where it is applied to the character while he is standing on the platform.



If the platform is moved via rigidbody scripting, then these are the settings that should be used. Any positions or rotations that are not adjusted by the script should be frozen. The Platform.cs provides a simple rigidbody movement to test out the characters interaction with the platform, which you can replace with your own movement or waypoint system.

Note: Any platform must also be tagged 'Platform' for the script to work properly.

Everything in the script is highly commented, so you should have no problem adapting it to suit your needs.

Please direct any questions, comments, issues, or requests to the Unity thread at:
http://forum.unity3d.com/threads/186856-Soldier-Animation-Pack

After you've gotten a chance to use the Pack, please take a moment and rate it if purchased from the Unity Asset Store, and if you don't think its deserving of a 5 star rating, please let me know what I can do to change that for you?
https://www.assetstore.unity3d.com/#/content/9246

Thank you!
-Kristopher, CEO @ Explosive LLC