Whipstitch Platformer Tools

Whipstitch Games

Version 1.8

For additional support or feedback, contact *dsnettleton@whipstitchgames.com*

Overview

Whipstitch Platformer Tools is designed to be a useful toolkit for building 2D / 2.5D platformer games. It contains numerous character movement options, as well as a variety of scripts and object templates for building a rich, feature-filled world for your characters to interact with. The toolkit is ideal for sidescrollers, smash-style fighting games, endless runners, and anything else the developer can imagine. As of version **1.8**, a list of mechanics includes:

- ➢ Gravity Switching
- ➢ Swinging from grappling hooks
- ➢ Ledge Grabbing
- ➢ Double-Jumping
- ➢ Horizontal Air-Dashing
- ➢ Bounceable Objects
- ➢ One-Way Platforms
- ➢ Wall Jumps
- ➢ Moving Platforms
- ➢ Falling Platforms
- ➢ Buoy Platforms
- ➢ Libra (Mirrored) Platforms
- ➢ Slopes
- ➢ Running
- ➢ Jumping
- ➢ Crouching

All of these features are implemented both in Unity's 2D and 3D physics, and can be customized to suit your individual needs. All the source code is C#, open and well-documented for modification.

For additional support or feedback, contact *dsnettleton@whipstitchgames.com*

Revision History

1.8 - July 31, 2015

Added feature for gravity switching camera rotation with player, and crouch-toggle. Also performed significant code cleanup.

1.7 - Jul 17, 2015

Added new feature: grappling hook. Character can now swing on a rope. Also, smoothed turning for 3d objects.

1.5 - Jun 24, 2015

Implemented Unity's 3D physics as an alternative to the Physics2D platform.

1.4 - Mar 22, 2015

Added new feature: Libra platforms, replaced Ms. Whipstitch with a character built in Chibi Studio.

1.3 - Feb 16, 2015

Added information pertaining to new features: falling platforms and buoy platforms.

1.20 - Jan 20, 2015

Cleanup for Asset Store release, documenting layer and tag settings. Simplified number of layers required.

1.0 - Jan 12, 2015

Initial draft.

For additional support or feedback, contact *dsnettleton@whipstitchgames.com*

Setup

Physics Types

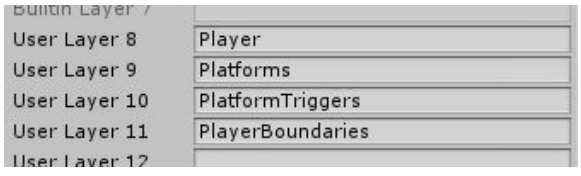
Many of the prefabs in Whipstitch Platformer tools have implementations for both 2D and 3D physics. If you would like to use Unity's 3D physics instead of the simpler 2D implementation, select the prefabs ending in "3D," where applicable. There is a directory under /WhipstitchPlatformerTools/Prefabs/ called "Physics3D." This contains all the necessary prefab duplicates, other than the player character.

Creating a Character

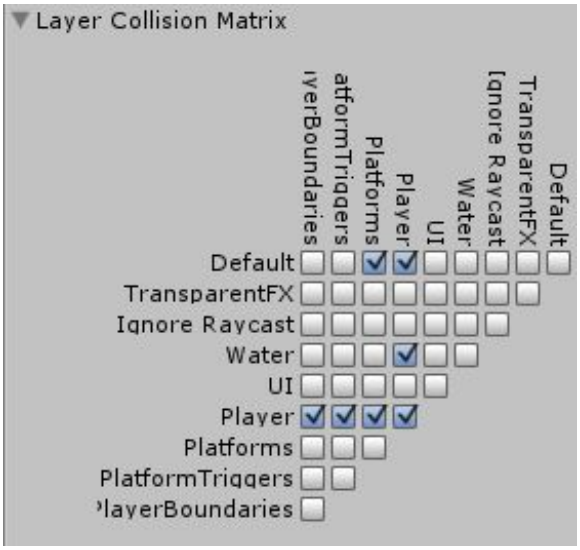
Platformer characters do not have to be the player. In fact, effort was given to the ability to allow numerous characters in the game at once without issue. A template has been provided using the Mr. Platformer (ninja) character and some default settings and animations. The character can be found in your assets folder under /WhipstitchPlatformerTools/ChibiStudio_Objects/PlayerCharacter.prefab. If you would like to use 3D physics, select PlayerCharacter3D.prefab from the same directory. Most of the work happens in the script **wsPlatformCharacter.cs**, while user input is handled by **wsPlatformer2DUserControl.cs**. This easily decouples the controls from the character itself so that an AI substitute is relatively painless to create. If you have your own character you may substitute him or her for Mr. Platformer in the prefab's hierarchy, and drag the model to the Model Transform field in the **wsPlatformCharacter.cs** script. It is also recommended that you place any platformer characters in their own Physics layer if you have not already done so. Also, be sure to customize the input script to your liking.

Layers

It is important that certain objects be placed on particular layers, or at least on layers that properly interact with the platformer characters.

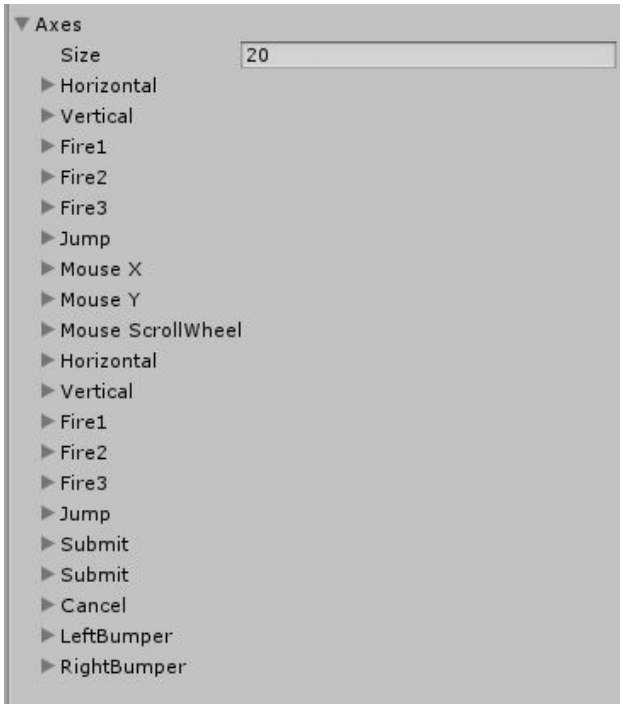


For additional support or feedback, contact *dsnettleton@whipstitchgames.com*



The layers and collision matrix for the included prefabs are shown above, but these can be altered to suit your needs. Any enemies or other entities acting as platformer characters should have similar interaction behaviors to the Player.

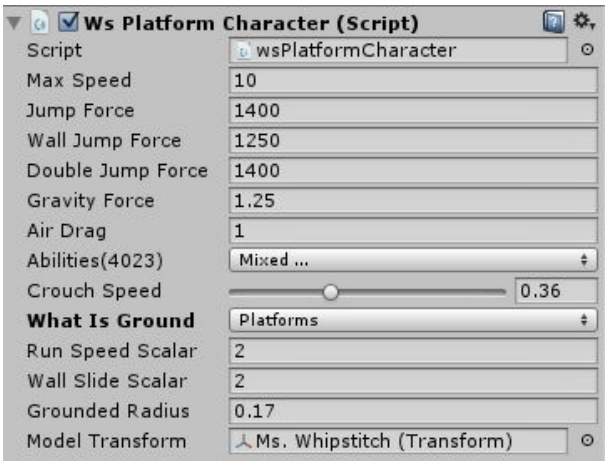
Inputs



Unity's default inputs are mostly sufficient for the setup provided. Two input options have been added, which are buttons labeled RightBumper and LeftBumper. They are the inputs used for air-dashing, if that ability is enabled, and are set to E and Q, respectively, in the webplayer demo. If your input axes have already been set up, or if you'd like to customize the scripts yourself, worry not. All of the input axes used by the platforming engine are stored and accessed in the included C# script file **wsPlatformer2DUserControl.cs**. Button presses are checked in the *Update* function, and directional axes are checked in *FixedUpdate*. Also keep in mind that any abilities not being used by the player will not require their own inputs.

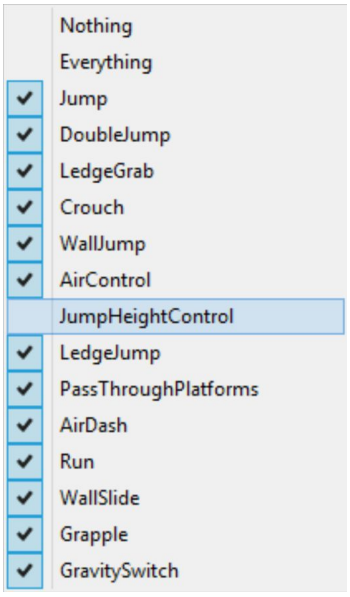
Character Values

Basic Movement



A character's basic movements are controlled in **wsPlatformCharacter.cs**. The most basic variable in the script is the character's Max Speed. This is the top walking speed of the character. Next are the jump forces and gravity force. Jump forces control how powerful a character's different types of jumps are, while gravity force is a scalar compelling the character to fall more quickly. The gravity force can be lowered to make a character feel floatier, or increased to make it feel heavier. Jump forces will have to be changed accordingly if you would like to achieve the same jump height. Finally, air drag slows the character's horizontal movement while in the air, providing slower recoveries from being knocked back, etc. Run Speed is a simple scalar causing the character to move exactly that much faster (e.g., a scalar of 2 will make the character move twice as fast) when running.

Advanced Movement



Platformer characters can have a wide variety of abilities, stored as a bitmask and accessed through the Abilities value in the editor. These abilities can be altered in order to limit your gameplay style, or to allow your character to be upgraded with new abilities throughout the course of a game. A character's ability to jump, crouch, teeter on ledges, grab onto ledges, and run can all be changed. Other abilities require a bit more explanation. Double jump allows the character to jump again while already in the air. Air

control allows the player to move the character left and right while the character is in the air. Although this doesn't make a lot of sense physically, it is generally more intuitive and precise for the player. A ledge jump allows the character to jump straight up when still hanging from a ledge. An air dash is a horizontal dash during which the character is unaffected by gravity, but gains some movement across the screen. It can be paired with attacks or used as a boost on its own.

The Pass Through Platforms ability refers to one-way platforms, described farther below. These platforms allow the character to come up from the bottom, but generally prevent it from passing through the top. If the character has the PassThroughPlatforms flag set to true, however, players can use a special input (such as Down + Jump) to allow the character to drop through to a lower level.

Wall Slide and Wall Jump are abilities that interact specifically with wall-jump panel objects. Wall slide slows the character down while on a wall-jump panel, in order to assist the player with more precise maneuvering. The speed at which the character is slowed can be controlled with the wallSlideScalar variable. Wall Jump allows the character to jump upwards and opposite of the wall by kicking off of it. The force of the wall jump is one of the jump forces included in the editor panel.

The Grapple ability is a new addition as of v1.7. It allows characters to latch onto targets with a grappling hook, and swing on a rope from that target. The rope prefab used can be set via the aptly-named ropePrefab variable. The ropeAttachPoint variable controls where on your character the rope attaches to. This can be an armature transform if you're using a 3d character, or any other attached transform will do.

Jump height control and Gravity Switching were added in version 1.8. Jump height control is a standard Mario-type jump, which allow players to limit the height of their character by releasing the jump button early. Gravity switching can be done using the platform character's setGravityDirection(*Vector3*) method, or by interacting with gravity switching prefabs. More information can be found farther down in this document.

World Building

Bounce Pads

A bounce pad can be created using the script **wsBouncePad.cs**. This script allows the user to set a force for the bounce, manipulating how high the character will bounce, as well as a flag (Bounce On Impacts Only) indicating whether the character will bounce immediately on contact, or must actually drop down from above to achieve the bouncing effect. No mesh is necessary for bounce pads, if you'd like the script to work invisibly.

The **wsBouncePad.cs** script and a trigger collider can be added to any object you might want the player to be able to bounce off of, including enemies. Bounce pads can give the player extra height if he or she jumps on landing.

Ledge Triggers

Ledge triggers are best set up using the provided prefabs, which can be placed at the ends of any ledges you might want characters to be able to grab onto. There is a right-handed and a left-handed version for your convenience. Platform characters intended to employ ledge triggers should have two transforms named *LeftLedgeGrab* and *RightLedgeGrab*. These transforms will be aligned with the *LedgePosition* transform upon a successful ledge grab.

One-Way Platforms

One-Way platforms are a common feature in platformers. They are objects that only allow characters to pass through from the bottom. Characters with the *PassThroughPlatforms* ability can drop through these platforms when given the command (often Down + Jump).

Moving Platforms

The script **wsMovingPlatform.cs** is a simple one which is given a begin and end position in the forms of Unity Transforms, as well as a movement interval indicating

how long it takes the platform to move from one position to another. Intervals are used instead of speeds in order to make it easier to synchronize the movements of multiple platforms. In other words, two platforms with the same interval value, yet different travel distances, will still change directions at the same time.

Sloped Terrain

Slopes have been accounted for by the platforming engine, so there's no need to perform any special coding or apply a script to sloped objects. The sloped prefab included in the package is intended only as a sample. Any 2D edge or polygon collider will do the trick.

Wall-Jump Platforms

As with bounce-pads, this script can work invisibly without a model or sprite. These platforms are used as triggers to indicate walls which a character can slide down or jump off of. It is important to set the trigger indicating whether the wall faces left or right, since this decides which direction the character will jump.

Falling Platforms

The script **wsFallingPlatform.cs** allows you to create a platform that will fall down after a platformer character stands on it. The Fall Delay variable allows you to alter set how long it takes, in seconds, before the platform begins to fall, and Gravity Force affects the speed at which the platform falls. Falling platforms can be automatically removed from the game-world by setting the Kill Time to a positive value representing the number of seconds after the platforms begins to fall before we should remove it. If Respawn Time is greater than or equal to zero, the platform will reappear that many seconds after it is removed. Finally, the object can be tinted as the clock ticks before it falls, allowing for a visual effect letting the player know something is about to happen.

Buoy Platforms

Buoy Platforms are simple platforms that begin falling at the designated speed (indicated by the variable Gravity Force) once the platformer character is on top of

them. When there is no longer anyone standing on the buoy platform, it raises back up to its original height. The maximum height to travel downwards can be set with the variable Max Drop Length.

Libra Platforms

Libra platforms are a special case of buoy platforms, named after the scales in the zodiacal symbol. Libra platforms come in pairs; one platform moves up whenever the other moves down and vice versa. Each platform moves down when the platformer character is on top of it, causing the other to move up.

Grapple Points

Grapple points are points in the world which a grappling hook can attach to. A grapple point must contain a trigger collider, the radius of which indicates how far away a character can attach its rope. If a character swings outside of that collider, the rope will detach. It's best to think of the collider radius as your character's rope length. You will probably want to set all of your grapple points to have the same radii, but this is not a requirement.

Gravity Zones

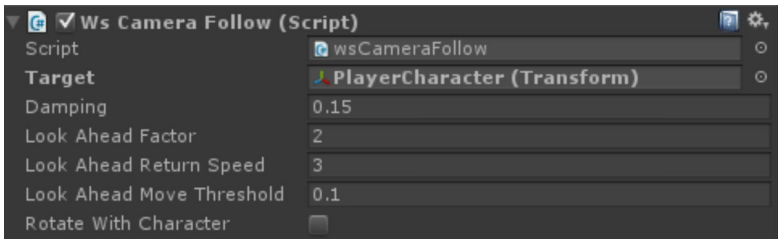
Gravity Zones are simple trigger prefabs that interact with characters who have the GravitySwitch ability. They set the character's gravity according to the rotation of their transform (gravity is set to the negative direction of the transform's "up" or Y axis).

Gravity Halfpipes

Gravity halfpipes are curved platforms that set the gravity of the character on them according to the normal plane of their surface. They allow characters to walk directly from floors to walls without ever having to leave the ground.

Additional Tools

As of version **1.8**, new tools have been added to the scripts wsCameraFollow.cs and wsPlatformer2DUserControl.cs. The camera script now has a toggle labeled "Rotate with Character." This toggle instructs the camera to change its angle according to the target character's transform. In other words, if the character rotates so that the negative X axis now down, the camera will rotate so that "down" is in the same direction.



Two new unrelated tools have been added to the user control script. The first is a crouch toggle, which allows players to simply press the crouch button in order to get their character crouching, and press it again to stop their character from crouching. The second tool is related to the new gravity feature, and is called camera relative movement. This adjusts the player's movement input so that the character moves in the direction pressed. For example, if the character is walking on the left wall due to a gravity switch, pressing the left arrow key will make the character crouch rather than walk to its left.