Smart Colliders Manual

Creative Spore

Support: creativespore@gmail.com

Web: http://www.creativespore.com

Unity Forums: user CreativeSpore

Index

1 Introduction

This is the user manual for the Unity asset Smart Colliders.

Smart colliders are an enhanced version of physic colliders included with Unity, improved to solve some problems found when using the default colliders like trespassing colliders when moving fast, sticking when moving over a surface due imperfections or collider joints, like moving a rectangle or box over a floor made by several colliders, etc.

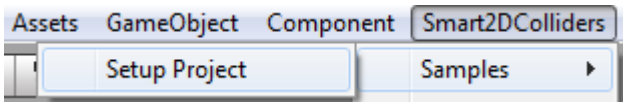
This manual will teach you how to use and configure the smart colliders to use them properly in your projects.

2 Project Setup

This asset have some scene demos in the folder "\CreativeSpore\SmartColliders2D\Extra".

This directory can be completely removed once you don't need it any more.

Also, if you want to configure the project to see the layer names and physic configuration used by these samples, you can go to the menu bar option "Smart2DColliders -> Samples -> Setup Project".

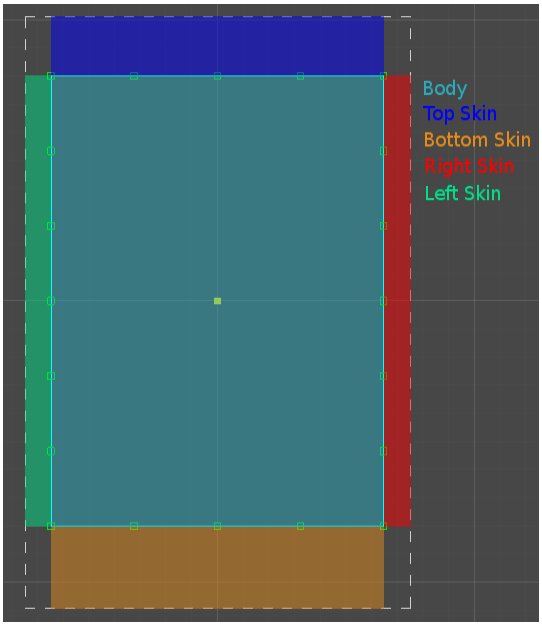


This will add the layer names to the project and also change the physics layer collision configuration.

3 Smart Colliders

3.1 SmartRectCollider2D

The SmartRectCollider2D is similar to the Box Collider 2D. It has a body with a size and center (offset) but it also has an skin around it.

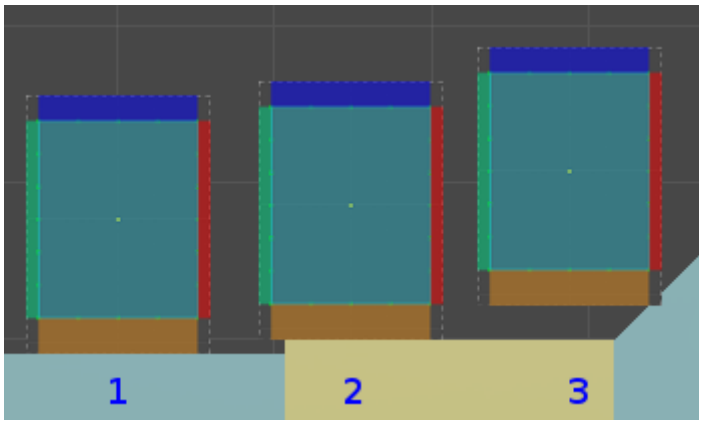


The body is used to check collisions against 2D and 3D colliders when moving from one position to another. The body is moved until a collision is found, even if the body is translated a long distance it will collide with colliders in the middle.

Then, the skin is used to separate the body from the colliders so, at the end, body and skin won't be colliding against any collider.

The skin is used to add a threshold were obstacles or collider joints can be avoided and it is divided in 4 different areas for each side.

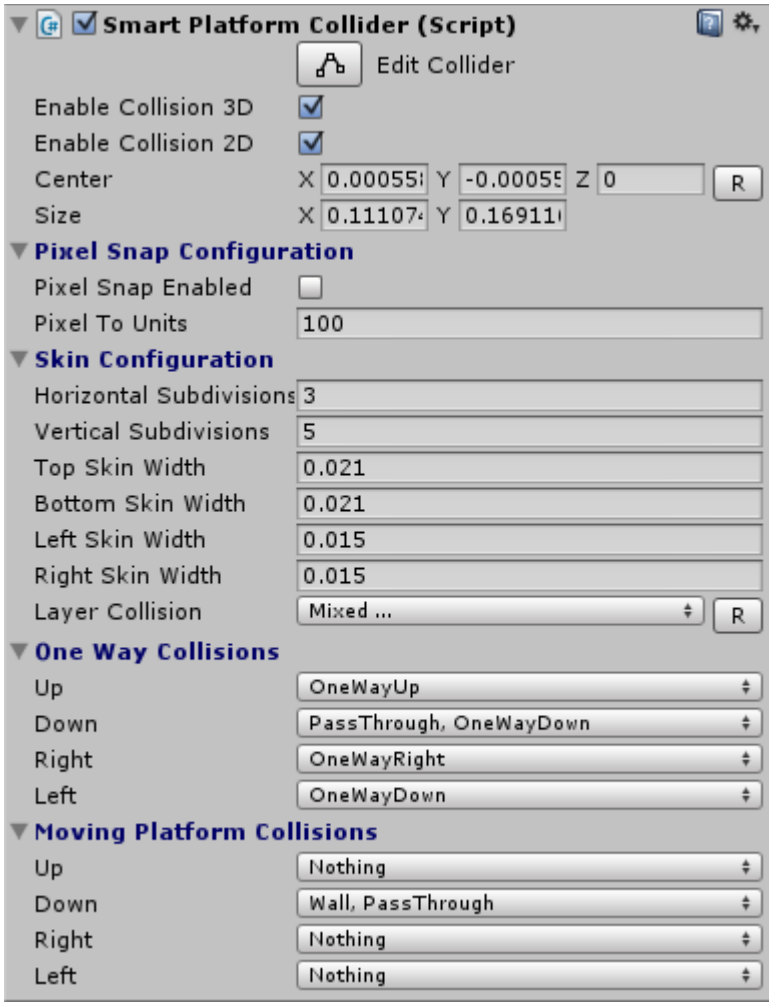
So when it is moved right, for example, the bottom skin allow the rectangle to climb slopes or stairs and pass through small obstacles.



- 1: First Frame
- 2: Right skin is not colliding with yellow collider and neither the body, but bottom skin is colliding so it is moved up to solve the collision
- 3: Body stops moving when colliding with the slope, then bottom skin is moving the rectangle up to solve the collision

3.1.1 SmartRectCollider2D (Inspector)

The inspector view of the SmartRectCollider2D script allow you to change some attributes and configure the collider for your needs.

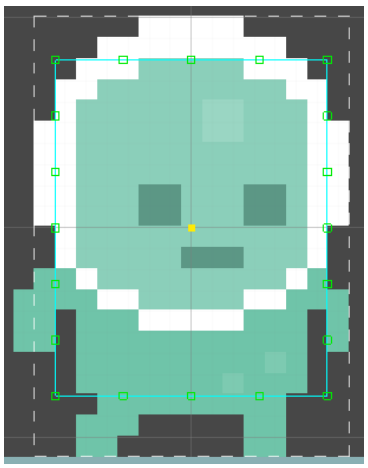


- **Enable Collision 3D:** enable collisions with 3D colliders.
- **Enable Collision 2D:** enable collisions with 2D colliders.
- **Center:** the position of the smart collider in the object's local space.
- **Size:** the size of the smart collider in the X, Y, Z directions.
- **Center/Size reset button ('R'):** these button will reset these values to default values. These are, if an sprite renderer is found, the center and the size of this sprite renderer.
- **Pixel Snap**: Enables the pixel snap performed before rendering the object using the value of PixelToUnits

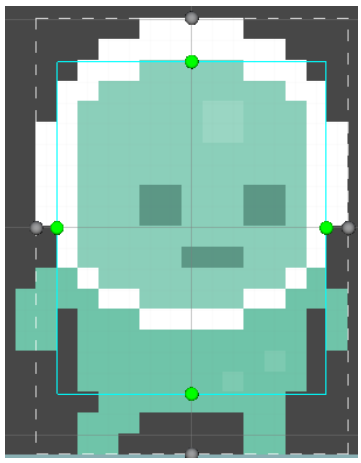
- **H(orizontal) Skin Subdivisions:** the number of check points between the two points for each extreme of each horizontal side (top and bottom). The final number of TopCheckPoints and BottomCheckPoints will be HSkinSubdivisions + 2.
- **V(ertical) Skin Subdivisions:** the number of check points between the two points for each extreme of each vertical side (left and right). The final number of LeftCheckPoints and RightCheckPoints will be VSkinSubdivisions + 2.
- **Skin Top Width:** the width of the skin on the top side.
- **Skin Bottom Width:** the width of the skin on the bottom side.
- **Skin Left Width:** the width of the skin on the left side.
- **Skin Right Width:** the width of the skin on the right side.
- **Layer Collision:** the Layer Mask of this smart collider with all layers that will be checked when solving collisions. Reset Button ('R') will reset this to the default values taken from Physics3D or Physics2D layer mask configuration according to the rigid body found in the owner gameobject.
- **One Way Collisions:** the collisions set here for each side will be taking into account only by the right side of the collider. So "Down: Passthrough|OneWayDown" will make the colliders with layer Passthrough or OneWayDown to displace the collider up until collision is solved when bottom skin is entering that collider.
- **Moving Platform Collisions:** the collisions set here for each side will make the owner game object move with the collided collider when collision is done in the specified side.

3.1.2 SmartRectCollider2D (Scene View)

The SmartRectCollider2D center, size and skin width can be modified from the scene view.



The yellow rectangle can be dragged to change the center.



Holding shift key, four gray circles will appear that could be dragged to change the skin width. Also, four green circles appear that could be dragged to change the body size.

3.1.3 SmartRectCollider2D - OneWay collisions

One important thing about these collisders is, the width of the skin matters.

When the right side of the smart collider inside an edge of the right collider, collisions will be solved so the smart collider is not touching the other collider. It means, for a long width, for example, for the bottom side, if the player is jumping over a pass through platform, when bottom side touch the top of the platform, the player will be moved up intermediately, so a long width in the skin will look like the player is getting impluse.

This is fixed in the Smart Platform Controller by detecting this kind of collisions and skipping them when jumping speed > 0.

If you are thinking about why are these collisions not checked when the moving direction is opposition the the one way direction, it is because of the moving platforms detection.

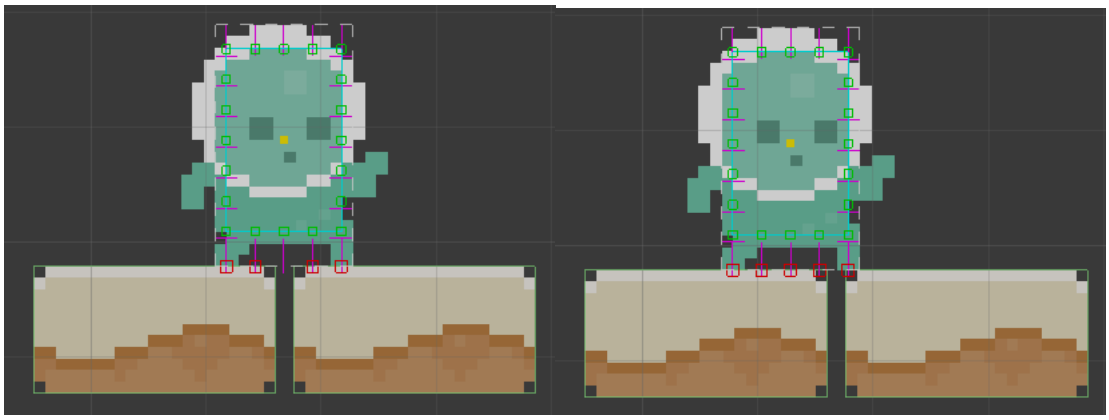
3.1.4 SmartRectCollider2D - Moving Platform Collisions

These collision masks allow some kind of layers to move the smart collider when collider is over it. Mostly it will be over the top of the platform (down side of the smart collider) but you can use the other sides as well, for example when the smart collider is supposed to be attached to moving walls.

Here, the width of the skin also matters, because the smart colliders only have dynamic collision detection when moving their self, but moving platforms doesn't have that detection, so if a collider of a moving platform is moving so fast, the collision could be miss.

Good thing is, once the smart collider is attached to the moving platform, it doesn't matter, only the first collision matters.

Other important thing is how much a platform move an smart collider when it is over more than one. It takes the average of movement based on how may rays are colliding each platform.

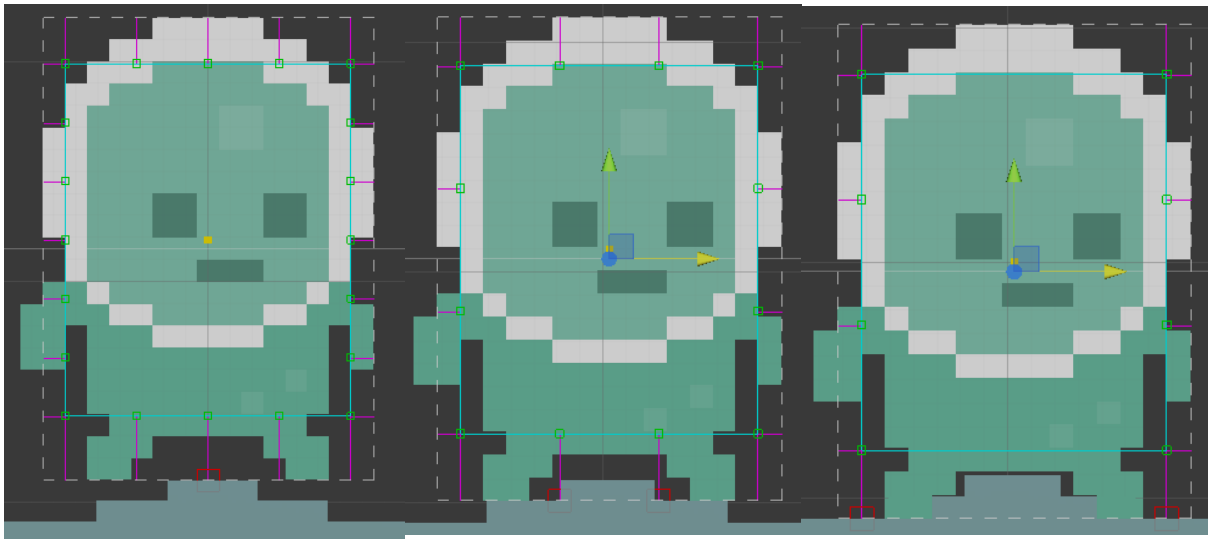


In this example, if left platform goes left and right goes right, in the first case, the player will be moved the average of the speed of each platform, so if speed if the same, he won't move.

But in the second case, the average will be 4 times the left plus 1 time the right divided by 5.

3.1.5 SmartRectCollider2D - Tips

- Disable the collisions you don't need to avoid unnecessary calculations.
- Reduce skin subdivisions as possible to improve the performance. Each subdivision will be another raycast thrown to check side collisions. Check this example to see when you could need more subdivisions.



(example with H Skin Subdivisions set to 3, 2 and 0)

- Even if the body is never trespassing colliders while moving fast, it's not checking when other colliders, like moving platforms, are being

moved fast over the rectangle. In this case, the width should be long enough to avoid missing the collision.

- The collider work even if there is a rotation in the object. Top, bottom, left and right sides will rotated as well, so if you rotate it 180 degrees around Z axis, bottom side will be on top, always facing -tranform.up direction, and the same with the other sides. Scaling the object will modify the width of the body and skin proportionally, and set a negative scale will flip the sides, so right side will be the left side and vice-versa.

3.2 SmartPlatformCollider

This collider is derived from **SmartRectCollider2D**. It works almost the same but it's specialized form a platform controller.

For example, this colliders give priority to bottom and top collisions before resolving the side collision, allowing a better movement through slopes.

4 Platform Character Controller

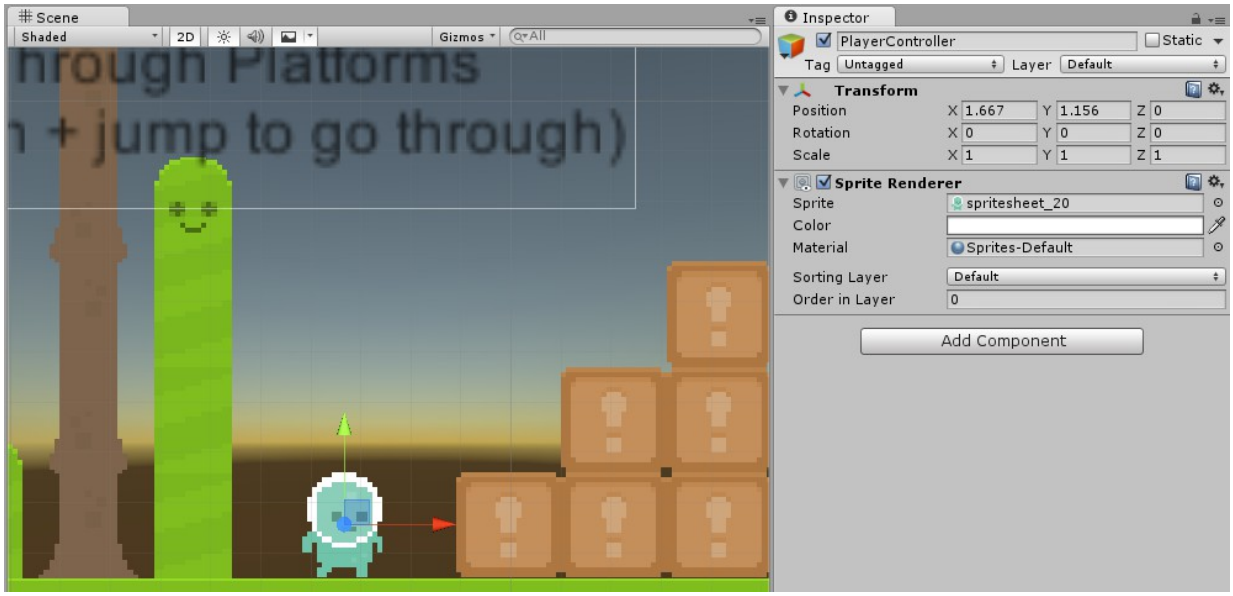
The **Platform Character Controller** is a component created to manage the platformer physics of a player character using a **smart rect collider** to manage the collisions with the world.

There is a prefab already created and configured in "**\CreativeSpore\SmartColliders2D\Extra\Prefabs\PlatformCharCtrl.prefab**".

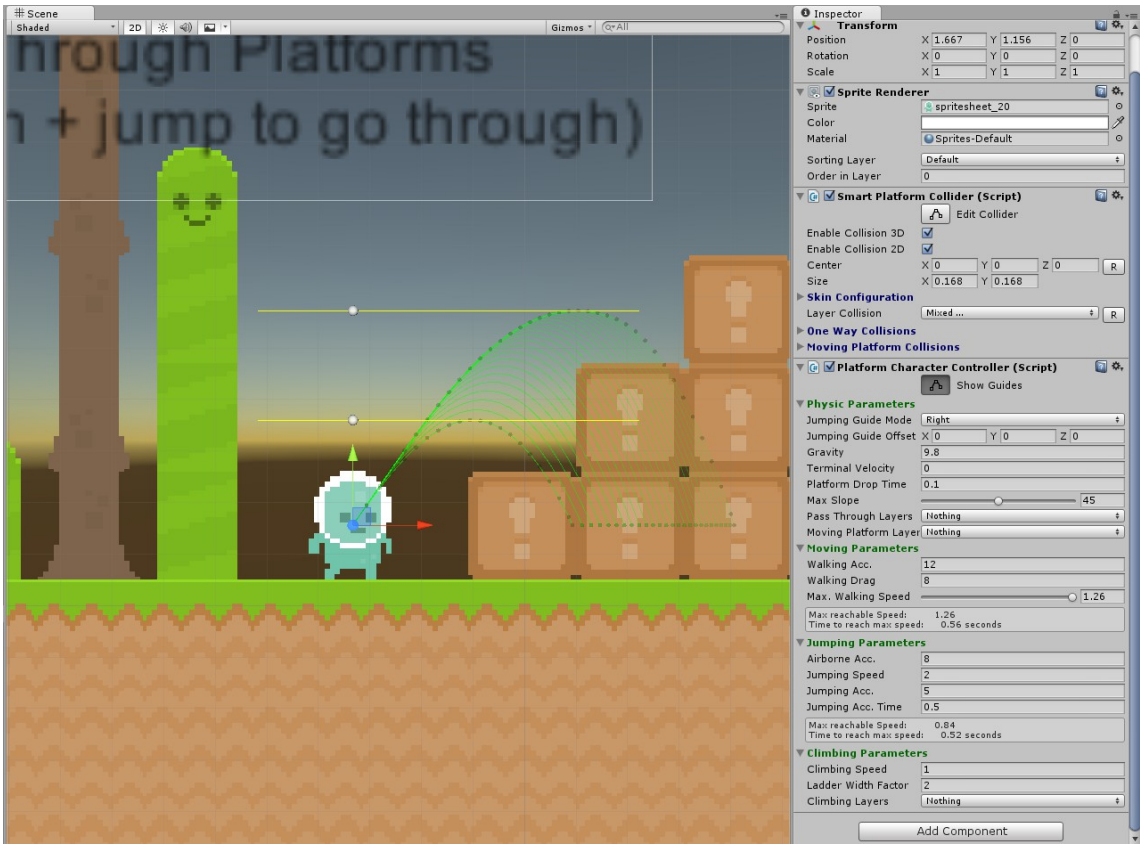
The next sections will teach you how to create and configure your own character controller.

4.1 Creating a 2D Platform Character Controller

1. Add an empty game object to the scene first and add a **SpriteRenderer** to this object and select a sprite for the player.



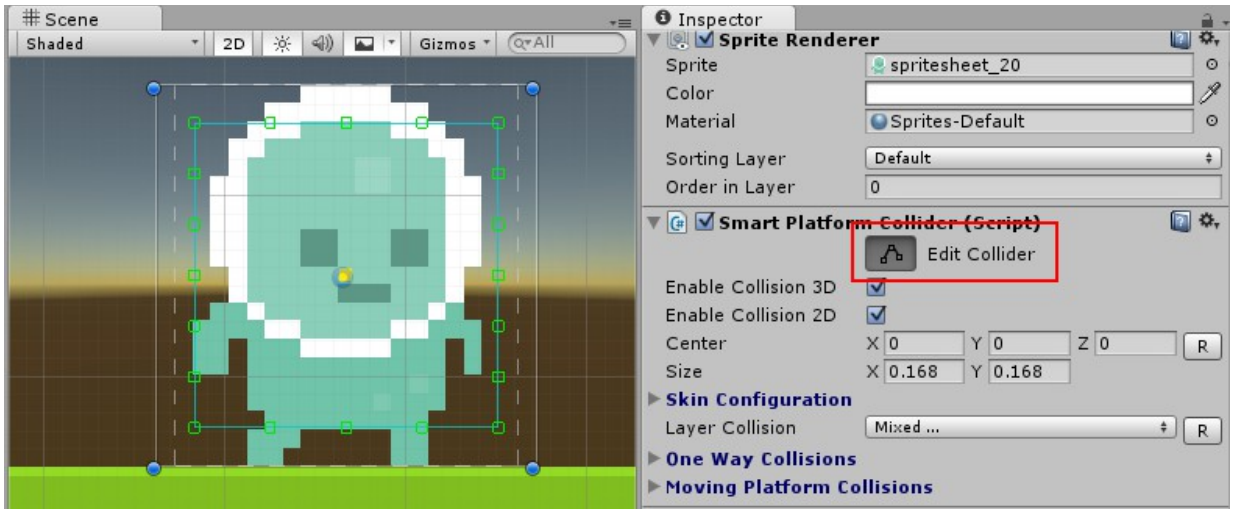
2. Now add the **Platform Character Controller** component.



The **Platform Character Controller** will also add the **SmartPlatformCollider**. This component has similar functionalities to the **SmartRectCollider2D** but it has been specialized in platform games.

3. Add the **Platform Character Input** component to allow moving the player with keys of gamepad.
4. Optionally add the **Platform Character Animator** component to add animations for each player state.

4.2 SmartPlatformCollider Setup



Press the Edit Collider button to enable the collider edition.

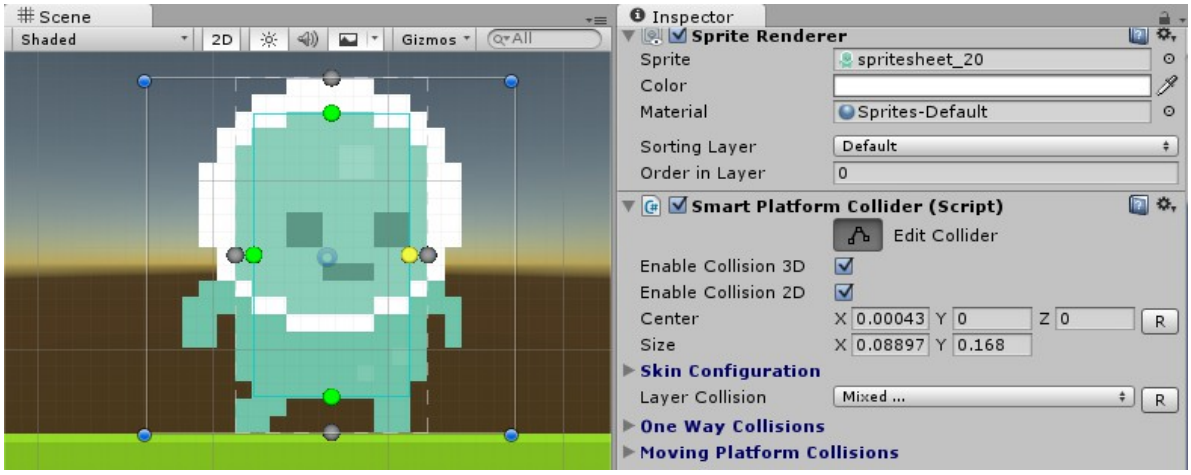
The inner square with small squares on their sides is the body of the collider. This rectangle will never go inside any world collider when moving the player.

The dotted square around the body is the skin. The skin will resolve collisions after the body does, and only in the skin side direction. There four sides: top, bottom, left and right.

For example, the bottom skin side will resolve collision going up, so the width of the bottom skin will allow the player to go up stairs with steps sorter than the width.

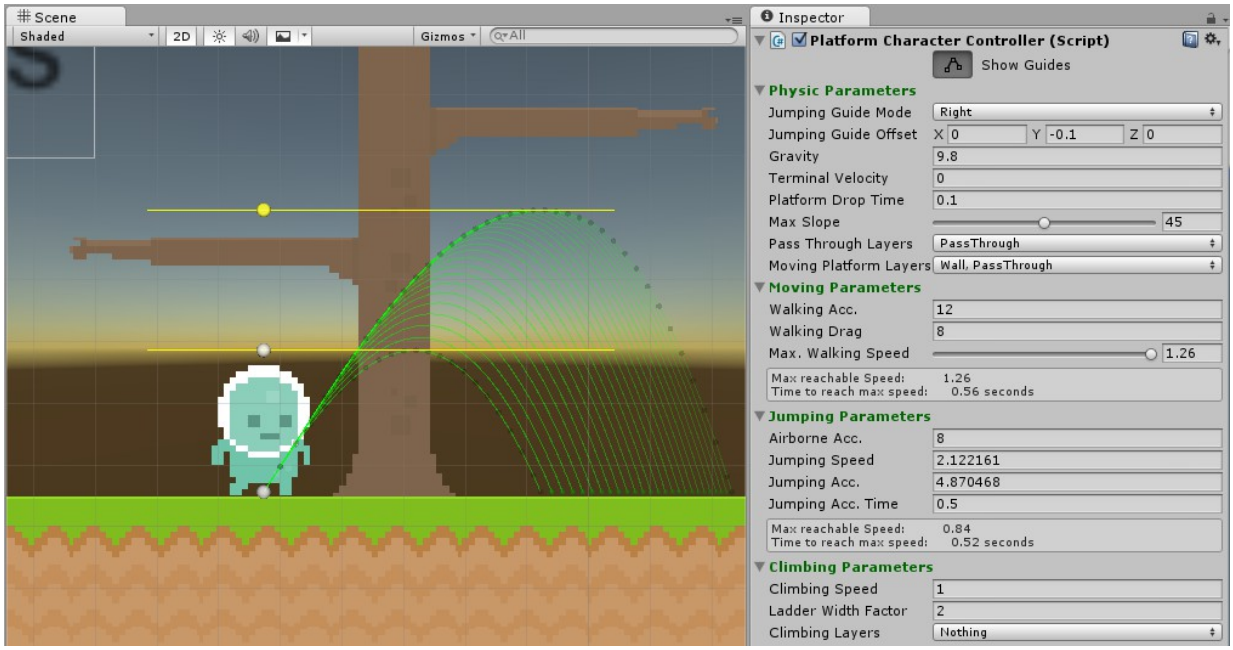
To change the **body** and the **skin**, hold **shift key**.

Two handles will appear per each side, one for the skin and other for body. Drag the handles to change the body or skin sizes.



4.3 Platform Character Controller Setup

Press the "**Show Guides**" button to display the jumping guides and handles.



These guides show the different trajectories the player would do between a sort and a long jump.

The two horizontal lines represent the minimum and maximum height the player will reach for a sort and long jump.

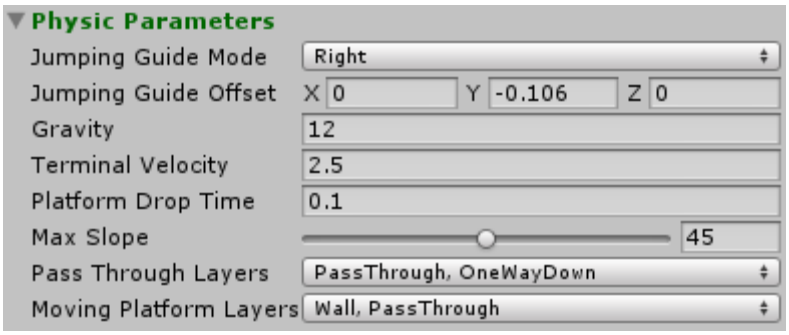
They both have a handle so you can drag them at any position, allowing the player to reach a different hight.

The bottom handle, where the guide lines start, is the offset used as jumping reference and can be dragged to any other position.

The rest of properties of the platform controller can be modified in the inspector and will be explained in next sections.

4.3.1 Physic Parameters

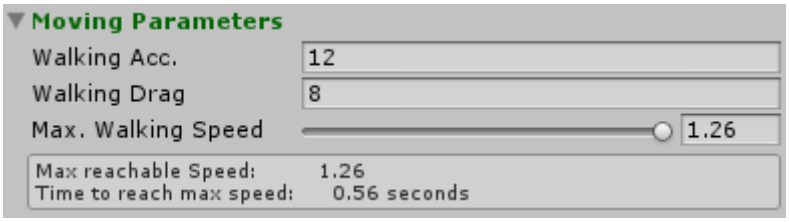
All these properties are related with physic behaviour:



- **Jumping Guide Mode**: can be Right, Left and Moving Direction. Display the jumping guides in the specified direction.
- **Jumping Guide Offset**: the offset where the jumping guides start.
- **Gravity**: the gravity force applied to the player.
- **Terminal Velocity**: the maximum falling speed. If 0, there will be no limitation.
- **Platform Drop Time**: when there is an action of dropping down of a platform, the bottom colliders will be disabled during this time to allow trespass pass through platforms.
- **Max Slope**: this is the maximum slope angle the player is allowed to climbing up and down being attached always to the ground and keeping the same walking speed.
- **Pass Through Layers**: these are the layers whose colliders the player will be able to trespass in any direction but down direction.
- **Moving Platform Layers**: the colliders with this layer will move the player when it is on top of them to any place they are moved.

4.3.2 Moving Parameters

These parameters affect the maximum walking speed and ground slipperiness.

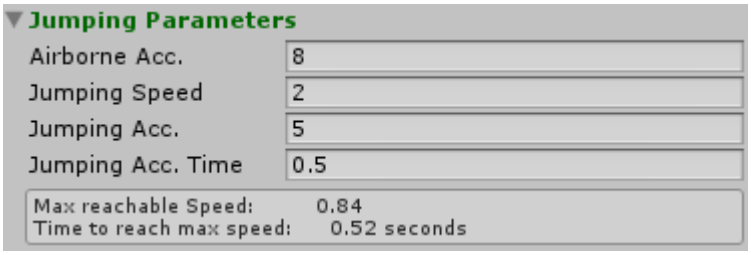


- **Walking Acceleration**: the acceleration applied to the player when moving right or left on the ground.
- **Walking Drag**: the resistance applied to the player for any lateral displacement. A higher value will reduce the maximum walking speed and a small value will make the player slide on ground.
- **Max. Walking Speed**: the maximum speed the player can walk at previous walking acceleration and applying the walking drag. You can set this speed between 0 and the maximum possible speed.

The bottom info box will display the max reachable speed and the time to reach this speed.

4.3.3 Jumping Parameters

These parameters define how much the player can jump and the moving speed on air.



- **Airborne Acceleration**: this is the force applied when moving the player right or left while not in ground.
- **Jumping Speed**: this is the initial vertical speed applied when jumping.

- **Jumping Acceleration**: this is a vertical speed applied when jumping if jumping action is hold during the Jumping Acc. Time.
- **Jumping Acc. Time**: the maximum time the Jumping Acc. is applied while jumping.

The **jumping speed** value is directly affecting the minimum jump marker in the jumping guide.

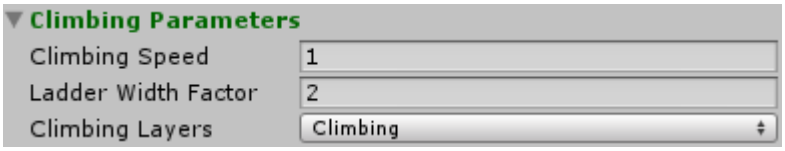
The first jumping guide shows the trajectory if only this jumping speed is applied.

The other guide lines until the top one are the different trajectories if **Jumping Acc.** is applied from 0 to **Jumping Acc. Time** seconds.

The info box shows the maximum reachable horizontal speed and the time to reach this speed.

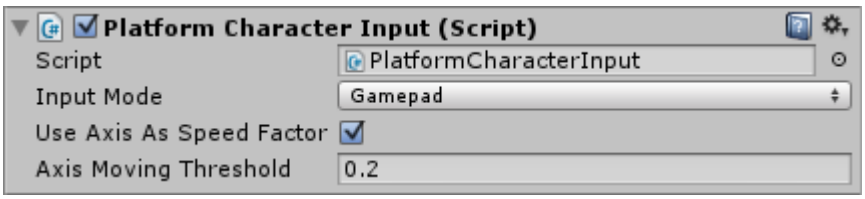
4.3.4 Climbing Parameters

These parameters are used when player press up over a climbing collider.



- **Climbing Speed**: the speed of the player when moving over a climbing area.
- **Ladder Width Factor**: if the climbing collider width is less than the player smart collider width multiplied by this number, the player will be snapped to the center of the collider. This is used for ladder where you want the player to stay in the middle.

4.4 Platform Character Input



This component will behave as a bridge between the input controller and the **Platform Character Controller**.

The code inside is really simple. It basically have a gamepad detection based on the press of button 0 of the controller and keyboard input detection if left or right arrow is pressed.

- **Input Mode**: select the initial input mode.
- **Use Axis As Speed Factor**: apply the axis value [-1..1] to the horizontal and vertical speed if true, or apply always the maximum speed if false.
- **Axis Moving Threshold**: the minimum value of the axis to consider a movement.

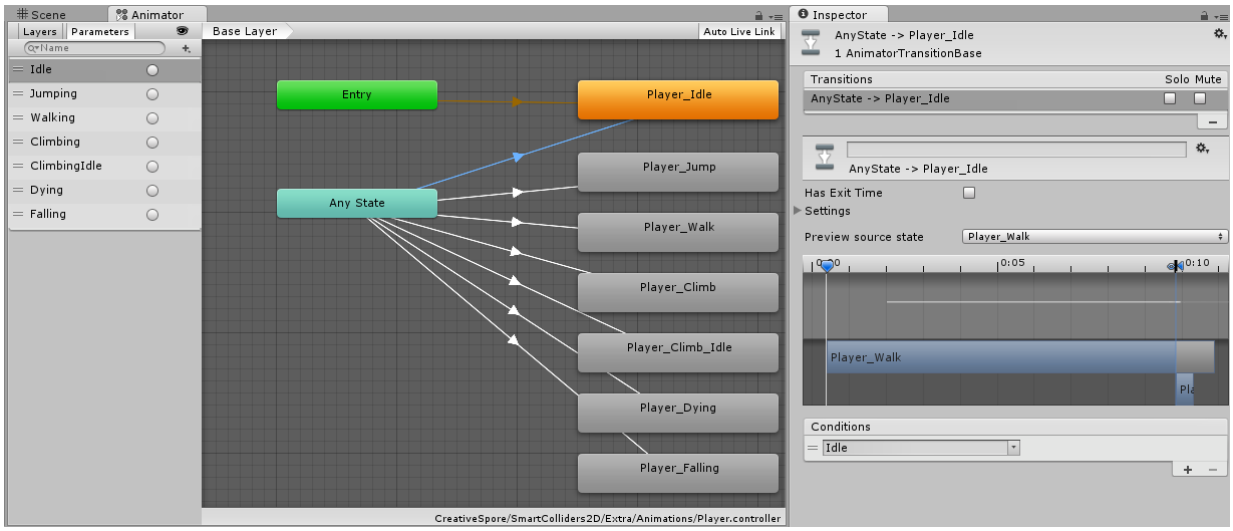
4.5 Platform Character Animator

This component adds an animator to the game object and connect with the **Platform Character Controller** to determine the current state of the player: Idle, Walking, Jumping, Falling, Climbing, ClimbingIdle.

Each time the player state changes, it activates a trigger in the animator with the same name of the state.

If the animator has a condition to trigger the right animation when the trigger is launched, the player will be animated according to the state.

The asset includes two animators for two different characters: Player and Monk.



This is the Player animator.

There are as many triggers as state names in the parameters list.

From Any State there is a condition to go the the right animation with the only condition of the trigger.

5 Additional Information

If you want to know more about using **Smart Platform Colliders**, you can find some useful tutorials in youtube:

- Getting Started https://youtu.be/yM3sGyvD6lE
- Create a simple platform game https://youtu.be/ZwIl62at6qA
- How to create a Platform Controller https://youtu.be/pyFnkjhfyoE

Creating Moving Platforms: https://creativespore.wordpress.com/2015/11/27/moving-platforms-patrol/