Mec Foot Placer

User Guide

Reference Pose

Contents

1- Introduction

Mec foot placer provides an automatic workflow for the character feet to be placed on grounds and uneven terrains. This document provides the details of this system and depicts how it can be setup. Mec foot placer acts as a post process on animations so while it places the foot automatically on grounds, it will save the overall shape of the feet determined by the active animation(s).

2- Workflow

Mec foot placer can find the appropriate foot position on ground by using raycasts. The system uses three raycasts to find foot, toe and the corner of heel position. Toe position is used for foot pitch rotation and heel corner position is used for foot roll. The foot yaw rotation will be obtained from the animation itself to make sure the original animation pose is not wrongly affected. The system always check the ground availability based on foot position from the current active animation(s). If system detects any ground, the system will set the foot in an appropriate position and rotation on it.

When the system is active, it will automatically place the foot on ground in these steps respectively:

1. First it gets the foot position from current active animation(s).

2. It casts a ray from an origin on top of the foot position in direction of the up vector with a custom offset distance.

3. The ray is pointing down from the origin in the direction of the opposite up vector with a distance equal to the same offset distance from step 1 plus foot height and a custom extra ray distance value. Figure 1 shows how the ray is casted for step 1 to 3.

   Figure 2 shows the final foot position after detecting a contact point. White sphere in the figure 2 is showing the detected contact point.
   The detected contact point is not suitable for placing the foot because it is ignoring the foot height and it causes the leg to be stretched and penetrate through ground. So a vector equal to UpVector * FootHeight will be added to detected contact point (white sphere) to set the final foot position. The Up Vector will be normalized automatically within the system. The blue sphere in Figure 2 shows the final foot position.

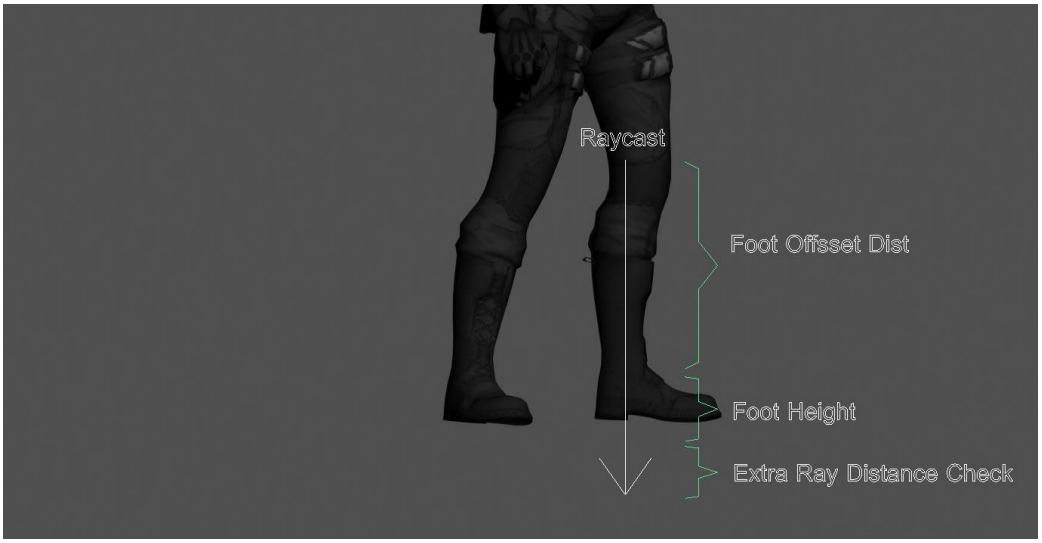

Figure 1- Ray casting for finding foot contact point

Figure 2- Final foot position after detecting a contact point

4. From the detected foot position another ray is casted based on the forward vector and current Foot rotation from the FK pose (foot animation pose). This ray is used to find toe position. The toe position is going to be used to find foot pitch rotation. Figure 3 shows how toe position is going to be found by using raycasts.

Figure 3- Raycast for finding Toe position

The Toe Vector in figure3 is equal to foot yaw rotation from animation multiplied by normalized forward vector multiplied by foot length:

$$ToeVector = FootAnimationYawRotation * ForwardVector * FootLength$$

where

$*$ *for* FootAnimationYawRotation *is the overloaded quaternion vector mutiplication in Unity3D and forward vector is normalized*

Applying foot yaw rotation from animation is causing the system to save the original foot direction determined by artist/animator. Figure 4 shows the detected toe position leading the foot to be placed on the surface correctly. Blue sphere shows the detected toe position.

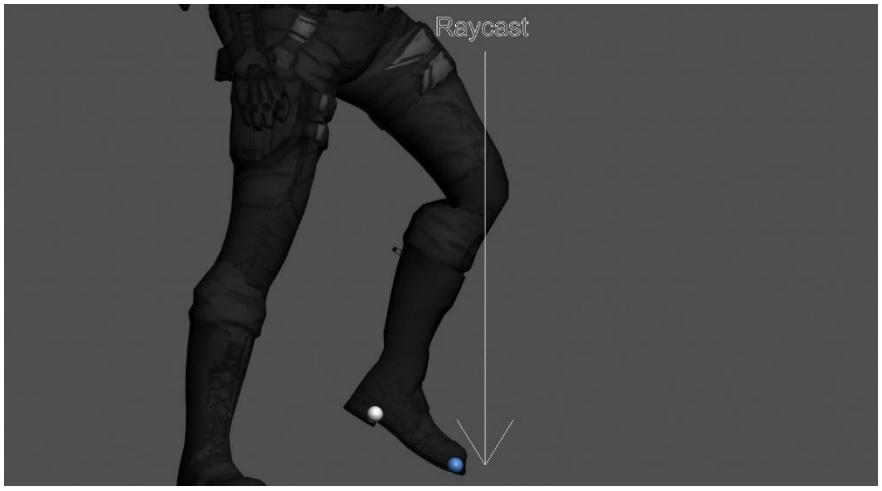

Figure 4- Detected toe position and the according foot rotation

5. From the detected foot position in step 1 to 3 another ray is casted to find the heel corner position. This ray is used to find the foot roll rotation. Figure 5 shows how this step is working.

   The Heel Vector in figure5 is equal to foot yaw rotation from animation multiplied by normalized right vector multiplied by foot half width. Where right vector is forward vector rotated 90 degrees around up vector.

$$HeelVector = RightVector * FootHalfWidth$$

Figure 5- Raycast for finding Heel corner position

The blue sphere in the Figure 6 shows the detected heel corner position and the roll rotation of Foot IK based on that.

Figure 6- Detected heel corner position and the according foot roll rotation

6. The system also adjusts the IKHints of legs automatically to achieve a natural knee shape. IKHints are known as swivel angles as well. The IKHints or swivel angles are determining the surface in which The IK chain is being solved. Figure 7 shows how the detected IKHint position is set. The blue sphere is showing the final IKHint position and the white sphere is showing the detected toe position. Calf vector is a vector in direction of calf bone (lower leg) and with magnitude equal to calf bone length.

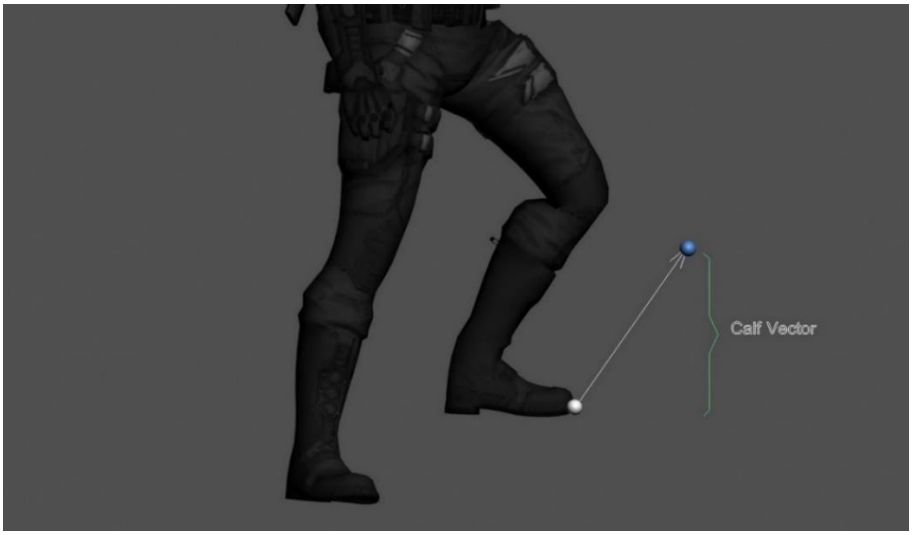

Figure 7- Final IKHint position (swivel angle)

7. The system can also adjust the character's pelvis to avoid unrealistic foot stretching. If the pelvis adjustment feature is checked, the system compares the distance of the detected ground contact position and the upper leg position. If the distance is higher than the specified leg length, it means the leg is stretched and it comes up with an unrealistic look. So the pelvis is adjusted based on the calculated error between the detected ground contact position and the leg length. This error is multiplied by the current foot up vector and will be added to the pelvis through time to move the pelvis along the up vector. Figure 8 shows the situation.

Note that when the system is in IK mode, if the ray for foot position detection fails to contact to any ground the foot placement system will transition to FK mode smoothly through time. This is also true if the system wants to switch back from FK to IK as well.

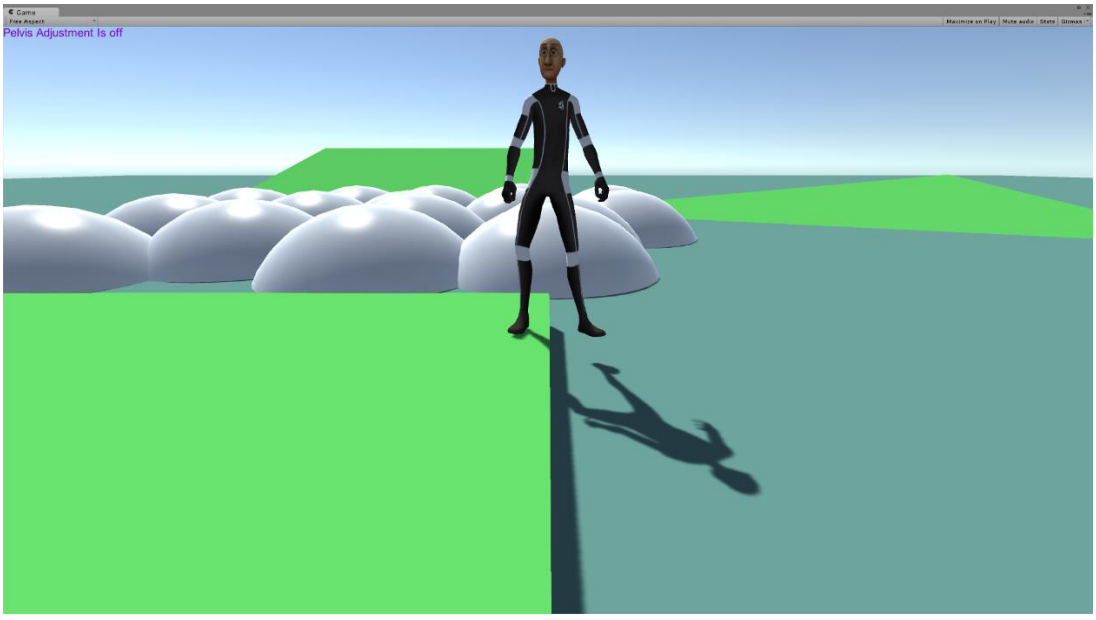

Figure 8-1- Character without pelvis adjustment. Leg stretched and located on air

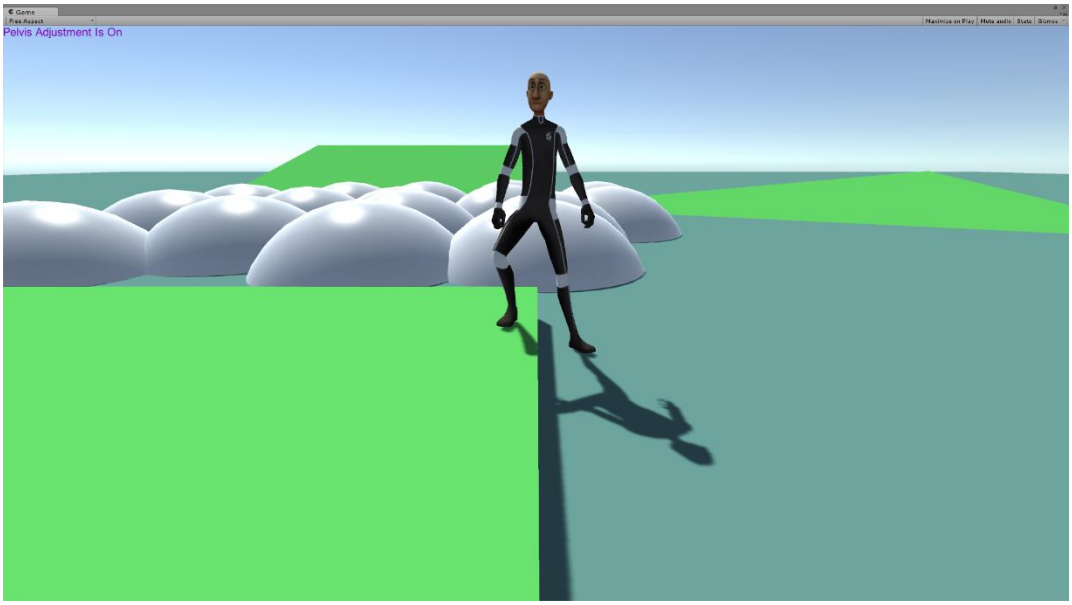

Figure 8-2- Character with pelvis adjustment. Leg is not stretched anymore and it's placed exactly on ground

3- Foot Placement Input Data

Each foot needs an input data to work correctly. For this reason a FootPlacementData component is provided to manipulate each foot needed data.

The input variables coming with the component should be filled by user. Each input variable is described here:

- **Foot ID:** Selects the foot or hand ID. This shows which foot or hand this component information belongs to.

- **Plant Foot:** If this check box is checked, the system will check for foot planting feature. Character's foot will be planted after detecting a contact point and it will remain on the detected position and rotation until it automatically returns to FK mode. It also checks for the ground height changes so feet can be placed on ground while being planted. This feature provides good solution to avoid foot skating.

  If the plant foot check box is not checked the foot always gets its position and rotation from animation and after that the foot placement uses raycasts to place it on the ground. If the plant foot is checked the foot will be placed on the first detected contact point and it will not follow the animation while it is in the IK mode. While foot plant feature is active the system can blend between the planted foot position and rotation with the position and

rotation of the foot without plant foot feature. However feet are always placed on the uneven terrains and grounds.

Foot plant feature has some functions to manipulate its blend weights. Check out section 4 to find out more about its functions. It is recommended to enable this feature in some states which might have foot skating like locomotion states and disable it in the other states which is not capable of having foot skating like standing Idle. Please check "IdleUpdate.cs" and "LocomotionUpdate.cs" to find out how to disable and enable this feature safely.

- **Disable Plant From Distance:** If the distance between planted foot position and the current Foot position from animation goes beyond this value the foot stops panting and it will blend back to non-planted foot.

- **Disable Plant From Angle:** If the rotation difference between planted foot position and the current Foot position from animation goes beyond this value in degrees the foot stops panting and it will blend back to non-planted foot.

- **Forward Vector:** This vector should be set to show the character foot initial direction. What you see in the Character foot reference pose or Mecanim T-Pose is what you should use here. At many times character initial forward vector is equal to foot forward vector.

- **IK Hint Offset:** The IK hint position will be calculated automatically as stated in section 1. The IK Hint Offset is added to this the final calculated IK Hint Position to fine tune the final IK hint position.

- **Up Vector:** Up vector shows the character up vector. This should be equal to world up vector (Vector3(0, 1, 0)) if the character is moving on ground. Otherwise for some rare situations like running on walls this should be changed accordingly.

- **Foot Offset Dist:** The distance used for raycasting. Figure 1, 3 and 5 show the parameter in action.

- **Foot Length:** Foot length is used to find the toe position to set the foot pitch rotation. It should be equal to character's foot length. Figure 3 shows the details.

- **Foot Half Width:** This parameter should be equal to half width of the foot. It is used to find foot roll rotation. Figure 5 shows the details.

- **Foot Height:** Foot height is used to set the correct heel position on ground. It should be equal to distance of heel center to lower leg joint. Figure 1 and 2 show the details.

- **Foot Rotation Limit:** The IK foot rotation will not exceed from this value and this will be its limit for rotation. The value is considered in degrees.

- **Transition Time:** When no contact point is detected by the system, it will switch to FK mode smoothly through time. Also when the system is in FK mode and it finds a contact point it will switch to IK mode smoothly through time. The "Transition Time" parameter, is the time length of the smooth transition between FK to IK or IK to FK.

- **Extra Ray Distance Check:** This parameter is used to find correct foot, toe or heel corner position on ground. Figure 1, 3 and 5 show the parameter in action. This parameter can be changed dynamically to achieve better visual results. Check out the IdleUpdate.cs and LocomotionUpdate.cs scripts in the project to find out the details. These scripts are changing the "Extra Ray Distance Check" value based on the foot position in the animations and the current animation state. Both scripts use Unity 5 animator behavior callbacks.

- **Set Extra Ray Distance Check Automatically:** As mentioned above it's better to change the extra ray distance check parameter manually based on the animation frames which the foot is stable or unstable on the ground. So at the frames in which the foot is planted on the ground like standing idle or support phases in run or walk, the Extra Ray Distance Check should be increased to make sure that foot hits the ground. Surely this needs some extra scripting like what the IdleUpdate or LocomotionUpdate scripts are doing or you can do them with animation curves. However if you don't want to do extra scripting or adding animation curves, you can turn this feature on and the FootPlacementData component will set the ExtraRayDistanceCheck value automatically. By turning this feature on, the component will find the frames that in which the foot is stable on the ground and at these frames it will set the ExtraRayDistanceCheck to a user specified maximum value and if the foot is not stable it will set the ExtraRayDistanceCheck to a minimum value specified by user. Check out "UnityCharacter_No_Animator_Extra_Scirpts" prefab and "DefaultAvatar_AutoRayDistanceSet" scene

- **Foot Distance from Ground Threshold:** This value only works if the "Set Extra Ray Distance Check Automatically" parameter is checked. The system checks out foot distance from character position alongside up vector. If the distance is lower than this threshold, it means foot is close enough to its planting animation frames and it should be adjusted by casting a longer raycast. Here the ExtraRayDistanceCheck will be set to "Extra Ray Distance Check Max" and if this distance is higher than this threshold, the ExtraRayDistanceCheck will be set to "Extra Ray Distance Check Min".

- **Extra Ray Distance Check Min:** The minimum value used to set "Extra Ray Distance Check". Only works if the "Set Extra Ray Distance Check Automatically" is checked.

Please check out “Foot Distance from Ground Threshold” and “Set Extra Ray Distance Check Automatically” sections for more info.

- **Extra Ray Distance Check Max:** The maximum value used to set “Extra Ray Distance Check”. Only works if the “Set Extra Ray Distance Check Automatically” is checked. Please check out “Foot Distance from Ground Threshold” and “Set Extra Ray Distance Check Automatically” sections for more info.

- **Draw Debug Ray:** If checked, three debug rays will be drawn only in the scene viewport. The red ray shows the ray casted for the foot, green for heel and blue for toe. Please note that the rays are drawn if they hit a physical object otherwise no ray will be drawn.

4- Mec Foot Placer Component

The Foot Placement Component is responsible for setting the correct rotation and position of feet on the ground and switching between feet IK and FK automatically. It also can adjust character’s pelvis to avoid unrealistic foot stretching often seen while using foot IK.

Mec Foot Placer provides some functions which can be used by user. These functions are stated here:

- **void SetActive(AvatarIKGoal foot_id, bool active):** This function will set the system on or off safely for each foot. At some states you don’t need the system to be active. For example when character is falling, there is no need to check for foot placement. However the system will work correctly on this state too but the user can disable it to ignore the calculations of the component. Each feet can be activated or deactivated separately.

- **bool IsActive(AvatarIKGoal foot_id):** If the current foot is active the function returns true otherwise false.

- **void SetLayerMask(LayerMask layer_mask):** Sets the layer mask which is going to be used in raycasts within system. The default value is Everything (LayerMask.NameToLayer(“Everything”); which means all of the objects in the world can be collided by the raycasts.

  Previously there was a check box in foot placement data components named “Ignore Character Controller”. The option is removed and instead the ability of setting layer masks is added. To avoid probable collisions between the character controller of the owner game object and the raycasts within system, the layer mask should be set correctly. For example

if you set the owner game object layer to 8 and you want to have collision with the whole world and avoid collisions with the owner game object, you can use this sample code:

SetLayerMask(~0 & ~(1 << 8));

or

SetLayerMask (LayerMask.NameToLayer ("Everything") & ~Mathf.RoundToInt(Mathf.Pow(2, LayerMask.NameToLayer(“LayerName”)))

- **LayerMask GetLayerMask():** Returns the layer mask set for the raycast.

- **void EnablePlant(AvatarIKGoal foot_id, float blend_speed):** This function will set the plant foot weight from its current value to 1 through time based on blend speed parameter. The function is useful to be used for some states which need foot planting like locomotion. By using it plant foot feature will be activated smoothly through time. To have the plant foot feature effective the plant foot weight value should be higher than 0.

- **void DisablePlant(AvatarIKGoal foot_id, float blend_speed):** This function will set the plant foot weight from its current value to 0 through time based on blend speed parameter. The function is useful to be used for some states which doesn’t need foot planting like standing idles. By using it plant foot feature will be deactivated smoothly through time.

- **void SetPlantBlendWeight(AvatarIKGoal foot_id, float weight):** Sometimes you need to manually change the plant blend values rather than using DisablePlant or EnablePlant functions. This function sets the blend weight between planted foot position and rotation with the non-planted foot position and rotation.

- **float GetPlantBlendWeight(AvatarIKGoal foot_id):** This function returns current blend weight for foot planting feature.

Mec foot placer have some public member variables which are related to the pelvis adjustment feature. They are listed here:

- **Adjust Pelvis Vertically:** It is highly recommended to turn this feature on just in the stationary situations like idle animations. Turning this feature on, in situations like fighting and locomotion can affect the original motion since having a stretched leg in running or in many combat animations is common and the pelvis should not be adjusted because of the stretched leg.

Turning the pelvis adjustment on or off can be done easily via behavior scripts in Unity 5. For example in “SimpleLocomotionCointroller” the pelvis adjustment is just turned on while entering IdleLeft and IdleRight states and it turns on when it goes to Locomotion State. Check out PelvisSet.cs and PelvisUnset.cs scripts in the “Extra Scripts“ folder to find out more.

- **Damp Pelvis:** If this variable is checked, pelvis will have a slight damping while being adjusted.

- **Max Leg Length:** Indicates the leg length. It should be equal to the distance of character’s upper leg and foot. Pelvis adjusts itself based on the error between max leg length and the current detected position on ground to avoid leg stretching.

- **Min Leg Length:** This shows the minimum distance between upper leg and the foot of the character. Pelvis adjusts itself so that difference between upper legs with their corresponding foots never goes less than the min leg length.

- **Pelvis Adjustment Speed:** Shows the speed of pelvis adjustment translation.

- **Layers To Ignore:** The layers that needed to be ignored for ray-cast collisions can be added to this list. The layers also can be added using SetLayerMask function. It’s always suggested to add character’s layer to this list so the system would not face self-collisions.

5- Quick Setup Guide

To setup the system you have to add a MecFootPlacer component. The foot placement component needs at least one FootPlacementData component otherwise it will not work. If two feet are needed to be considered, two foot placement data components should be set. One for right foot and one for left foot so the system can manipulate both feet.

After setting the components the Mec Foot Placer system should work correctly. Check out the example scenes for more info.

5-1- Important notes on setting up the system

Some important notes should be considered before setting up the system:

- **Exposing Necessary Bones**: If you checked "Optimize Game Objects" in the avatar rig, then some bones have to be exposed since the Mec Foot Placer needs them to work correctly. The bones are listed here:
  1- The corresponding bones for left and right feet.
  2- The corresponding bones for left and right lower legs.

  Check out the "Robot Kyle" avatar in the project to find how it can be done.

- **Setting up the data correctly**: Don't forget that setting up the data for foot placement needs precision and at some states the data needs to be changed dynamically to achieve the best effects. For example the "Extra Ray Distance Check" parameter should be increased or decreased in different states or animation times to achieve better visual results.

  Fortunately this can be done easily in Unity 5 by using animator behavior scripts. Check out the "IdleUpdate.cs" and "LocomotionUpdate.cs" in project to find out the details. Both scripts are being called within "SimpleLocomotion" animation controller. As Scripts show, the "Extra Ray Distance Check" parameter is increased while character enters in idle state. Also in the locomotion state, the "Extra Ray Distance Check" parameter increases in the times that character is putting his feet on the ground. This is to make sure that foot touches the ground. It will be decreased while the foot is not on the ground to help the foot move freely while looking for ground contacts as well.

- **Setting Extra Ray Distance Check Automatically:** From version 1.5 it is possible to set the Extra ray distance check automatically based on the frames in which the feet are stable. The component can detect the stable foot frames and change the extra ray distance check to a maximum value if the foot is stable and to a minimum value if the foot is unstable. Stable means that the foot is not moving much and unstable means that the foot is moving frequently. Please check out "Set Extra Ray Distance Check Automatically**"** for more info.

- **Checking IK Pass Check Box:** On any layer in which you need IK you have to check IK pass check box in the animator controller so the IK callbacks can be called by Mecanim.

- **Mecanim Specific Features:** Mecanim humanoid rigs provide a leg stretching feature which can help foot IK to look better. The leg stretching feature can avoid Knee popping while setting foot IK. However the value should be set accurately. High values of this feature makes the character cartoony and low values can increase the chance of knee popping in the character.

  To setup Leg Stretch feature you have to select your character asset. It should be a humanoid rig. In the rig tab, select configure then select muscles tab. In the "Additional

Settings" group, you can find a parameter named "Leg Stretch". Check out "Robot Kyle" avatar in the project to find out more.

- **Using Pelvis Adjustment Feature:** It is highly recommended to turn this feature on just in the stationary situations like idle animations. Turning this feature on, in situations like fighting and locomotion can affect the original motion since having a stretched leg in running or in many combat animations is a must and the pelvis should not be adjusted because of the stretched leg.

  If you want to use the pelvis adjustment feature, you have to turn off the "Optimize Game Objects" in the mecanim rig. When having the game objects optimized, the skeleton is just updated within the mecanim avatar update loop and it can't be manually adjusted. So it's recommended to use this feature if you don't need the "Optimize Game Objects" feature.

- **Using Mec Foot Placer for Quadrupeds:** From version 1.4 the hands can also be placed on ground using the same mechanism for foots. From now, Mec Foot Placer can be used for quadrupeds as well but only quadrupeds which have legs with 3 joints since the Mecanim humanoid rig just supports two bones IK solver and gives control for the third bone rotation so no leg with more than 3 joints can be used correctly. Using Mec Foot Placer for legs with more than 3 joints might cause unexpected behavior.

- **Using Mec Foot Placer for climbing and cliff hanging:** From version 1.4 Mec Foot Placer can be used for climbing situations like ladders and cliff hanging. Just note that in cliffhanging or ladder climbing animations, you need to change the up vector in the foot placement data components to point perpendicular to the surface which the character is climbing. Changing the values of the up vector can be done easily at run-time.

- **Avoiding self-collisions:** It's always good to add character's layer to the layer ignore list of the Mec Foot Placer component to be sure that there would be no self-collision between the physical objects of the character and the rays casted by the Mec Foot Placer component.