Upgrade guide 1.0.5 -> 1.1.0

The purpose of this “guide” is to give you a quick overview of the main changes from 1.0.5 to 1.1.0. Just like the release notes, but in a very informal way. For more information please check the release notes.

If 1.1.0 is the first version you are using of CCP, you can ignore this document.

I apologize in advance if I missed something in this guide. For more information please check the “How to…” section of the online documentation. I will be putting everything you need to know to use the asset effectively.

General

Namespaces and Demo content

A clear separation between what’s the character controller + logic system (Core and Implementation) and all the demo content built on top (e.g. NormalMovement state, Dash state, Camera2D, Camera3D, etc.)

Object Hierarchy

A big one, now the only requirement is: CharacterActor and CharacterBody belong to the root object (character). You can put the rest of the components wherever you want. This is great to keep the hierarchy clean, especially if you have many states, AI behaviours, extra components, etc.

Core

Input Velocity

The “Inputvelocity” property has been changed to “Velocity”. This is the equivalent to the Rigidbody velocity (rigidbody.velocity).

Position and Rotation

Now all the changes in position and rotation (teleportation as well) are instant changes.

Step up/down

The step up feature is now limited to the body size, which is a very logical thing to do. This brings a lot of good things, a much more fluid and stable movement, and ton of possibilities in the future.

Basic directions

“UpDirection”, “RightDirection” and “ForwardDirection” are now “Up”, “Right” and “Forward”. These fields will automatically adjust itself based on the type of character (2D or 3D).

- For a 3D character there is not much difference (think of transform.forward, transform.right and transform.up). So, **transform.forward == CharacterActor.Forward** (leaving interpolation aside).

- On the other hand, a 2D character requires a transform.forward = Vector3.forward (to prevent issues with the Collider2D area). So, in this case **transform.forward != CharacterActor.Forward**.

Character Graphics

In 1.0.x the CharacterGraphics component was responsible for controlling the rotation of the character, a very important task. Now (since this task is processed by the CharacterActor) this component is separated into small components (**CharacterGraphics2DRotator** and **CharacterGraphicsScaler**).

Now the "Graphics" object is now just a child of the character. It won't be separated from the root (like in 1.0.x), instead now it's a normal child that follows its parent position and rotation.

Kinematic Camera … Where is it?

The camera is now just a demo component, not tied to the SceneController update loop (only Characters and Kinematic Platforms).

Implementation

States … Where are they?

All the states were originally created not only to give value to the asset (and offer some useful tools) but also to demonstrate the functionalities of the Core. So, all the states have been moved to the Demo part of the package.

Transitions

In 1.0.x you choose a potential state by returning that state (CheckExitTransition method). So, what if there were more than one potential states? This was a bug!

-> In 1.1.x you are adding all the potential states to a queue.

State "Name" property

The Name property was always the class name. In 1.1.x this has been changed, now you refer to a state by getting that state by name (GetState("YourState")), or by Type (GetState<YourState>()). This works exactly as **GetComponent**, you need to specify only the class name.

Animation

Since 1.0.x didn't offer a good animation solution at all, the animation is the big new thing in this version (1.1.0). The old **CharacterAnimation** component has been removed, hopefully the integration of the Animator with the CharacterState will make your development **easier** and **possible**.

Character Actions

The concept is still the same, but the actions have been modified a bit. You can now set the value of the action, not the phase of it.

Quick example: You can choose if a button is pressed or not, but you cannot set if it is “released” (that’s something the input handler will do for you).

AI Behaviour … Where are they?

Based on some feedback (thanks David) the AI logic is now moved to separated Monobehaviour components. Now you should be able to create your own logic for you AI (basically managing the CharacterActions struct by code).