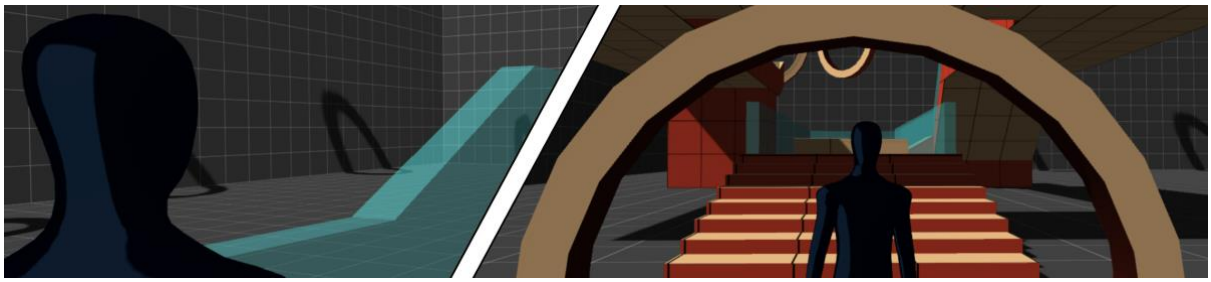
3RD PERSON CONTROLLER + FLY MODE



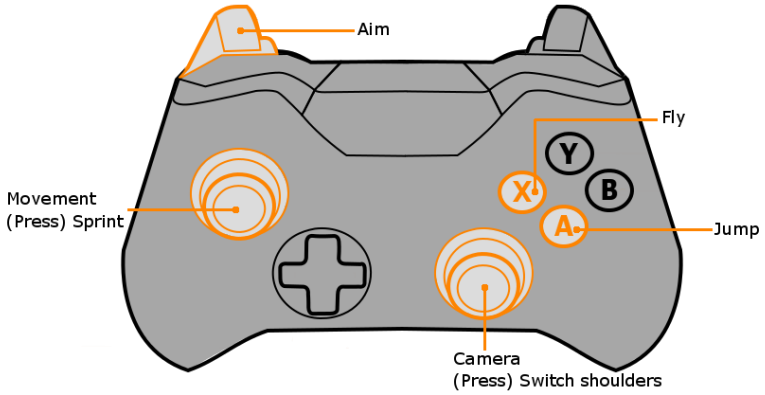
Thank you for acquiring the **3rd Person Controller + Fly Mode** asset for the *Unity 3D* engine!

This asset provides a basic setup for a **3rd person** player controller. Includes scripts for the player basic movements**, camera orbit** and a Mecanim animator controller.

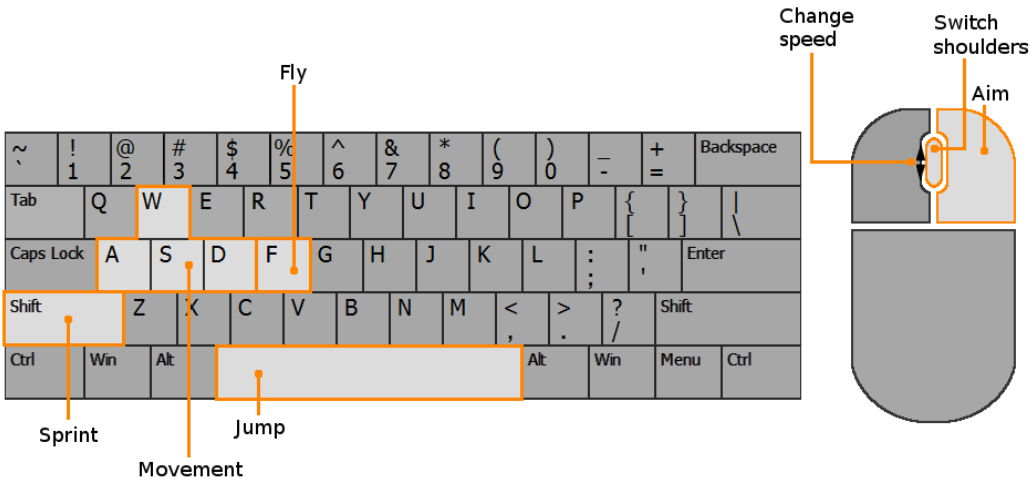
The basic locomotion modes includes **walk, run, jump, sprint, aim** & **strafe**, and also an extra **fly** mode. This package also comes with a dynamic **third person camera**, with full collision detection.

SETUP NOTES

This asset fully supports the **Xbox One** controller. The buttons and axes are configured under the *Project Settings* > *Input* section. The default controls are mapped as follows:



For keyboard and mouse, the input is mapped as follows:



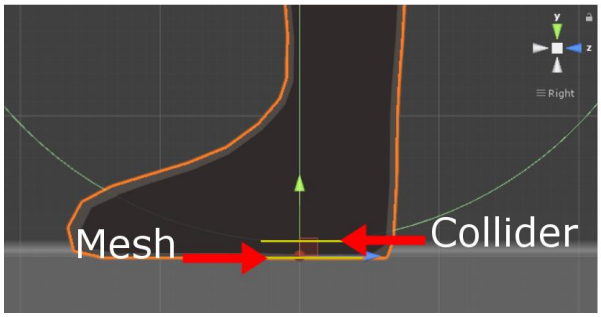
You can press **F2** on the keyboard, or the **Xbox One Controller** ☰ button to see the input commands on the example scene.

A detailed tutorial on how to setup the essential files on a custom project is provided on a video that can be accessed through the following link:

Tutorial: How to setup on a custom project

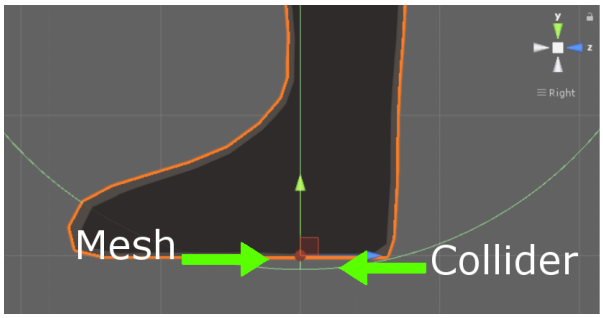
IMPORTANT NOTES

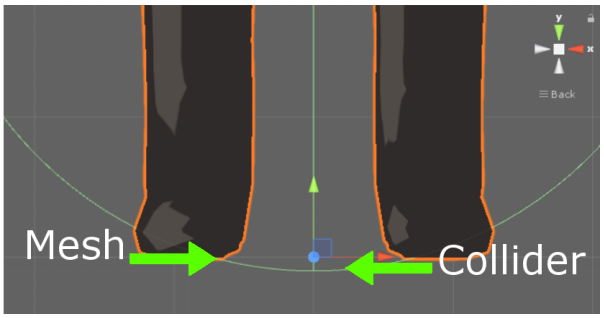
. When configuring the *Capsule Collider* on your own character avatar, you may carefully avoid the following situation:



If the collider end is positioned above the mesh end, the character will get stuck when trying to move. Do not let the avatar mesh surpass the collider!

The correct position is set when the bottom of the collider is at the same level or under the mesh end:





As a reference, to centralize the collider correctly, check the mesh size and use its height as the collider *Height* parameter. For the *Center* position, you may use half of the collider height minus 0.01, as the Y value of the parameter. After setting it, remember to check if the situation presented above does not occur.

THIRD PERSON ORBIT CAM BASIC

The **ThirdPersonOrbitCamBasic** class manages all the third person camera features, such as orbiting around the player, FOV changes, and zooming in/out to avoid collision. This script has the following external parameters:

- **Player:** a reference to the player's game object.
- **Pivot Offset:** the default offset to point the camera when orbiting the player.
- **Cam Offset:** the default offset to relocate the camera when orbiting the player.
- **Smooth:** the default camera movement responsiveness factor.
- **Horizontal Aiming Speed:** the speed of camera orbit on the horizontal plane.
- **Vertical Aiming Speed:** the speed of camera orbit on the vertical plane.
- **Max Vertical Angle:** the maximum angle to limit camera orbit on the vertical plane.
- **Min Vertical Angle:** the minimum angle to limit camera orbit on the vertical plane.
- **X Axis:** the default horizontal axis input name (for joysticks).
- **Y Axis:** the default vertical axis input name (for joysticks).

This script uses the Unity's default *Mouse X* and *Mouse Y* input axes to control the camera orbit by mouse, alongside with the **X Axis** and **Y Axis** custom inputs for joysticks. Remember to adjust the joystick axis *sensitivity* and *dead zone* to match your needs.

BASIC BEHAVIOUR

The **BasicBehaviour** class manages which player behaviour is currently active or overriding the active one, and call its local functions. This behaviour also contains a basic setup and common functions used by all the player behaviours. This script has the following external parameters:

- **Player Camera:** a reference to the player camera's game object.
- **Turn Smoothing:** the turn speed when moving to match camera facing.
- **Sprint FOV:** the camera *Field of View* to change when player is sprinting.
- **Sprint Button:** the default sprint action input name.

This behaviour uses the Unity's default *Horizontal* and *Vertical* input axes to control the player movement.

MOVE BEHAVIOUR

The **MoveBehaviour** class corresponds to walk, run and jump behaviour, it is generally the default behaviour. This script has the following external parameters:

- **Walk Speed:** the player's default walking speed (controlled by a Blend Tree).
- **Run Speed:** the player's default running speed (controlled by a Blend Tree).
- **Sprint Speed:** the player's default sprinting speed (controlled by a Blend Tree).
- **Speed Damp Time:** the delay to change between the locomotion animations when changing speed.
- **Jump Height:** the default jump height reached.

- **Jump Inertial Force:** the default inertial horizontal force applied to the player when on air, between jumping and landing. The higher it is, the longer the jump distance will be.

This behaviour uses the Unity's default *Jump* input axis to handle the jump action. The speed of locomotion can be controlled by the *Mouse ScrollWheel* axis.

AIM BEHAVIOUR BASIC

The **AimBehaviourBasic** class handles the aim and strafe movements. The following external parameters are present:

- **Aim Button:** the default aim action input name.
- **Shoulder Button:** the default switch shoulders action input name.
- **Crosshair:** the default aiming crosshair. If no parameter is passed, no crosshair is shown.
- **Aim Turn Smoothing:** the speed of turn response when aiming to match the player's orientation with camera facing.
- **Aim Pivot Offset:** the offset to repoint the camera when aiming.
- **Aim Cam Offset:** the offset to relocate the camera when aiming.

FLY BEHAVIOUR

The **FlyBehaviour** class handles fly action. The following external parameters are present:

- **Fly Button:** the default fly toggle input name.
- **Fly Speed:** the default flying speed.
- **Sprint Factor:** how much sprinting affects flying speed.
- **Fly Max Vertical Angle:** custom angle to clamp the camera vertically when flying mode is active.

ACKNOWLEGMENT

The author acknowledges some third-party assistance that facilitated the production of the demo examples within this asset. This includes the *Carnegie-Mellon University* mocap library, and the fellows that contributes to the *Unity Answers* community.

DISCLAIMER

This document is part of the **3rd Person Controller + Fly Mode** asset, that can be acquired through official channels – such as the *Unity Asset Store*.

Please support us for other great assets to come!

Author's page