These are some rough guidelines for what to consider when putting together a physics-based Character Controller in Unity.

## PlayerCharacter script

Make a script to go next to the CharacterController component that’s provided by Unity

* Remove any Colliders. The CharacterController has a built-in capsule
* This should talk to the sibling CharacterController component, and call CharacterController.Move() once per frame. Note that the movement you pass in is a position delta, so multiply by Time.fixedDeltaTime or Time.deltaTime as appropriate.
* Your class should have a member variable for velocity.
* Apply gravity every frame whether grounded or not. This makes the CharacterController.isGrounded method reliable. Use Physics.gravity and a possible multiplier
* Horizontal velocity should get overridden by the input every frame, with special care when not grounded that you maintain any horizontal velocity in the absence of control input.
* Add an OnControllerColliderHit function to get more detailed information about where the CharacterController is being hit. This gets called automatically like OnTriggerEnter does.
* Jumping should just set the velocity.y to a positive value when the Jump button is pressed and the character is grounded. You can use the kinematic equations to make the jump velocity match a desired jump height.

## Orbit Camera script

Make a simple orbit camera to follow the player and orbit with the mouse when the right button is held

* Moving the mouse left and right should make the camera’s eulerAngles.Y change. Use Input.GetAxis(“Mouse X”) to get horizontal mouse deltas
* Moving the mouse left and right should make the camera’s eulerAngles.X change. Use Input.GetAxis(“Mouse Y”) to get vertical mouse deltas. Maybe clamp this between 0 and 70 to stop the camera getting gimbal lock at the poles, or going under the floor.
* Position the camera at the target’s transform (player’s feet on an animated character), plus some distance up, maybe a metre, and then a distance back along the camera’s forward direction so that it always looks at the player.
* Use GetAxis(“Mouse ScrollWheel” to zoom in and out if you like by changing the distance back from the player.
* You can raycast from (player’s feet plus distance up) back along the forward vector to see if you hit anything. If you do, use RaycastHit.distance to pull the camera in. Let it relax out again if there’s no collision using Mathf. MoveTowards

## Character Animation

Add the Beta android model as a child of your character. This will automatically create the transform hierarchy. Replace the Animator component on Beta with an Animator component at the top level.

Make a simple AnimationController that does the following

* Defaults to a BlendTree (call it “Grounded”) which blends between an idle and a walking/running animation, based on a Speed variable. Set this variable to the forwards input in your code using Animator.SetFloat()
* Have a Boolean value for jumping. Transition to a Jump node with the jump animation and the Grounded state based on this variable. Untick “Has Exit Time” for these transitions.
* Do the same for a Crouch node, ie make another Boolean and hook up the transitions.
* When you’re crouching, make sure that the CharacterController’s capsule is shrunk, so they can fit under low doors easily. Just change CharacterController.height in your code

## Cloth Animation

How to give your character a cape

* Make a Unity Plane and scale it to 0.05, 0.1, 0.15. Move it on to the character’s shoulders and make it a child of the Spine2 node. Remove the MeshCollider.
* Give the plane a Cloth component. This will automatically remove the MeshRenderer and add a SkinnedMeshRenderer, while leaving the MeshFilter behind.
* In the Cloth tool for editing vertices, paint the first line of vertices on the shoulders to have a MaxDistance of 0.2, so they stay level with the character’s shoulders.

At this point you’ll be able to see the cloth flap right through the character’s body.

* Add a CapsuleCollider to Spine02, set it along the x-Axis and align it to fit the character’s chest. Err on the side of making it too big. Add to the Cloth’s list of Capsules. Test this.
* Add a SphereCollider to the hips and Spine02. Add these as a pair to the Cloth and you’ll see the pear-shaped capsule it makes. Test again.
* If you’ve added ragdoll to your character, you can add the leg and arm CapsuleColliders to the Cloth’s list of capsules, so the cape doesn’t clip through your legs.