Platformer Character Controller Guide

Platformer Character Controller is a free to use Character Controller Script for platformer games in Unity.

Actions :

- **Run**
- **Jump :**
    - Fixed Height : Jump Height remains fixed.
    - Dynamic Height : Jump Height depends on the duration of the keypress.
- **Dash (8 Directional)**
- **Grab :**
    - Hold : Can hold the walls
    - Climb : Can climb the walls
    - Climb Jump : Can jump while climbing the wall
   - Climb Jump Away : Can jump away from the wall

Additional Stuff:

- **Coyote Time**
- **Microscopic Dash Pause**
- **Stamina System :**
    - Grab Ability depends on the stamina points managed by the stamina system. ○ Each Grab Ability i.e. Hold, Climb, Climb Jump consume different amounts of stamina points.
- **Action Events :**
    - Each Ability triggers respective events that can be subscribed to, by different scripts to initialize various actions like animations.
- **Interaction with moving platforms:**

Setup :

- **Use provided Prefab :**
  - Drag the Player prefab from *Assets/DGM/CharacterController/Prefabs/* into the scene.
  - Add sprites to the gameObject and reset both the box collider and the capsule collider. Note : Capsule Collider should be a little smaller than the box collider from every side.
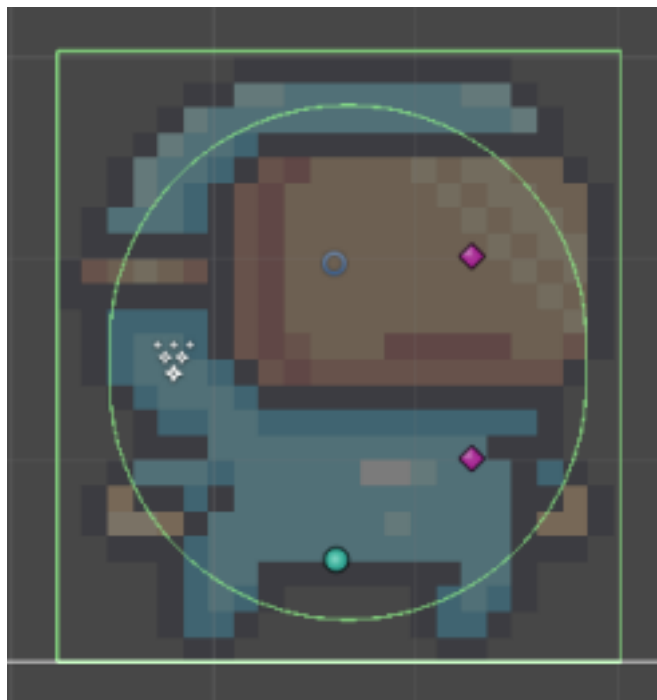
  - Adjust the position of foot, hand and shoulder gameobjects according to the collider. Adjust their length from the inspector (Enable *Hand and Foot Visualization* from the Inspector) and set the appropriate ground Layer and Grab Layer to foot and hand respectively. Note : these gameobject are for raycasting, so make sure foot, hand, shoulder raycast touch the ground, grab surface once they are close enough. Also make sure that shoulder is always above the hand.
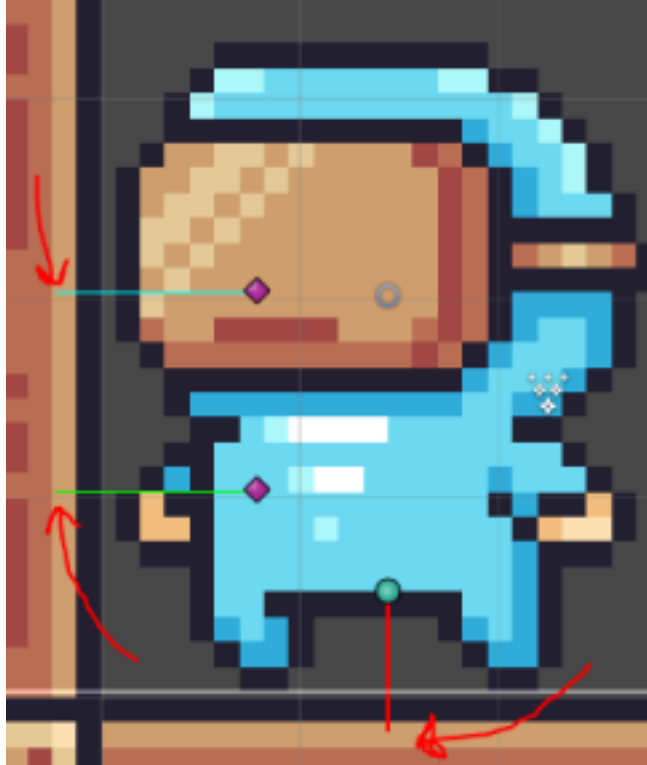
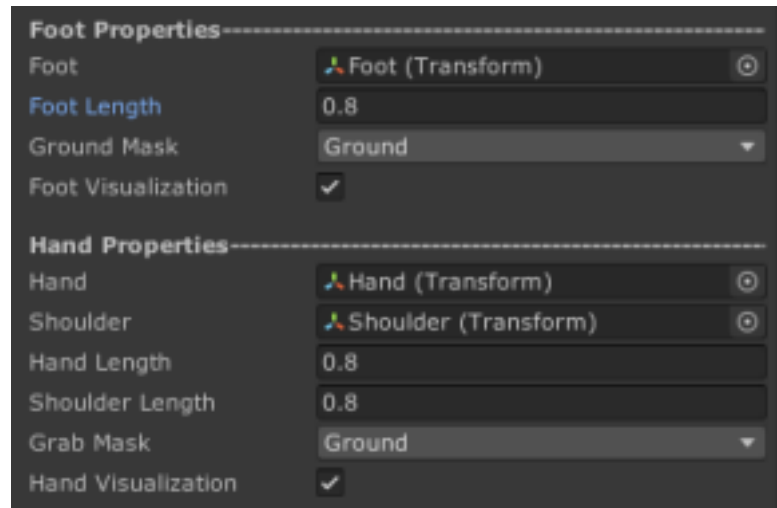○ Determine the actions from the inspector that the character should be able to perform.
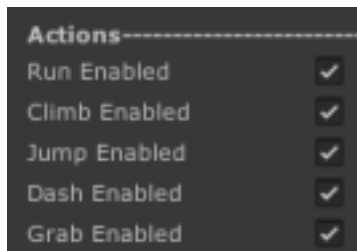
○ Adjust the values for different actions.

● **Add Script to existing GameObject :**

○ Apply the *Assets/DGM/CharacterController/Script/PlayerController.cs* to the parent of the player gameobject. (Rigidbody2D, Box Collider, Capsule Collider will be added automatically).

○ Adjust the colliders as described above.

○ Add three empty gameobjects as the child of the parent gameobject. These will work for the foot, hand and shoulder raycast position. Adjust their position as described above.

Make sure that they touch their respective surfaces when they should and add their reference to the Player Controller Script. Also make sure to add their reference in the inspector.

```
Foot Properties-------------------------------------------------------------
Foot                    ⚘ Foot (Transform)                        ⊙
Foot Length             0.8
Ground Mask             Ground                                    ▼
Foot Visualization      ✔

Hand Properties-------------------------------------------------------------
Hand                    ⚘ Hand (Transform)                        ⊙
Shoulder                ⚘ Shoulder (Transform)                    ⊙
Hand Length             0.8
Shoulder Length         0.8
Grab Mask               Ground                                    ▼
Hand Visualization      ✔
```

○ Adjust the values for the *Move Speed*, *Jump Force, Dash Force* etc from the inspector according to the actions.

➜ **Changing the controls of the character controller :**

◆ Open the PlayerMovementAction action map from the Scripts folder and modify the bindings.

Properties :

● **States** are the different states for different actions of the character.

CurrentMovementState is the one from SIMPLE, JUMP, DASH, GRAB according to the current action performed by the player.

CurrentGrabState is one from NONE, HOLD, CLIMB, CLIMBJUMP according to the current grab action performed by the character

CurrentAnimationState is one from IDLE, RUN, JUMP_GOINDUP, JUMP_GOINGDOWN, DASH, GRAB

States

Current Movement State	SIMPLE
Current Grab State	NONE
Current Animation State	IDLE

- **Actions** are the actions allowed to be performed by the character. Enable all those actions which can be performed and disable the rest.

Actions

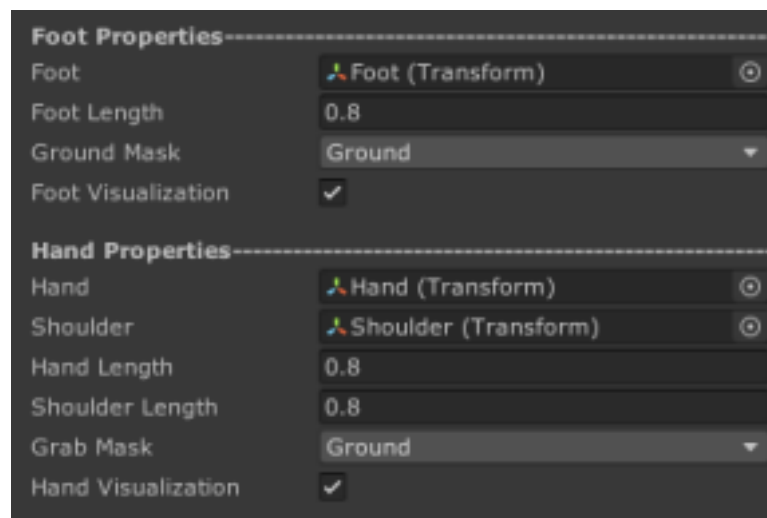|	
Run Enabled	✔
Climb Enabled	☐
Jump Enabled	✔
Dash Enabled	✔
Grab Enabled	✔

- **Checkers** are for checking various conditions. They are just for debugging and cannot be edited as they are updated each frame.

Checkers (Only for reference, Can't be edited)

Ground Check	☐
Ground Check Realtime	☐
Hand Check Realtime	☐
Old Hand Check Realtime	☐
Shoulder Check Realtime	☐
Ledge Nearby	☐

● **Foot Properties and Hand Properties**
   ○ *Foot, Hand* and *Shoulder* store reference to the foot, hand and shoulder transforms.
   ○ *FootLength, HandLength* and *ShoulderLength* are the raycast length at the particular location.
   ○ *GroundMask* and *GrabMask* are the layer masks of the ground and walls.
   ○ *FootVisualization* and *HandVisualization* turn on a visual ray at the respective location for the visualization of length and position of the foot, hand and shoulder.

Foot Properties---		
Foot	⚛ Foot (Transform)	⊙
Foot Length	0.8	
Ground Mask	Ground	▼
Foot Visualization	✔	
Hand Properties---		
Hand	⚛ Hand (Transform)	⊙
Shoulder	⚛ Shoulder (Transform)	⊙
Hand Length	0.8	
Shoulder Length	0.8	
Grab Mask	Ground	▼
Hand Visualization	✔	

● **Gravity Properties** are for the adjustment of the gravity.
   ○ *FallGravityModifier* is the value by which gravity is multiplied during falling.
   ○ *JumpGravityModifier* is the value by which gravity is multiplied during going up.
   ○ *AirDrag*, *LandDrag*, *DashDrag*, *GrabDrag* are the drags in the respective movement state.

```
Gravity Properties------------------------------------------------------
Fall Gravity Modifier      10
Jump Gravity Modifier      4
Air Drag                   1
Land Drag                  0
Dash Drag                  5
Grab Drag                  8
```

- **Coyote Values**
  - *CoyoteTime* is the time in which the character can still jump even when not grounded.

```
Coyote Values------------------------------------------------------------
Coyote Time                0.2
```

- **Jump Properties**
  - *FixedJumpHeight* enables jump of being of fixed height, else the jump height is determined by the duration of key press.
  - *NumberOfJumps* is the number of jumps that are allowed without touching the ground.
  - *JumpForce* is the force applied while jumping upwards.
  - *JumpForceAway* is the force applied while jumping away from a wall.
  - *JumpCancelModifier* is the value by which the velocity is multiplied once the jump key is up. 0.5 means that once the jump key is up, the velocity is reduced to half for quick descent.

```
Jump Properties----------------------------------------------------------
Fixed Jump Height          ☐
Number Of Jumps            2
Jump Force                 20
Jump Force Away            40
Jump Cancel Modifier       0.5
```

● **Dash Properties**
  ○ *NumberOfDashes* is the number of dashes allowed without touching the ground. ○ *DashForce* is the force applied for the dash.
  ○ *DashRecoverTime* is the time by which the dash is expected to end and gravity is enabled again. Note : Dash Force and DashRecoverTime are independent of each other, so it is possible for the character to still move fast because of the dash force and due to small dashRecoverTime, it regains the control. To keep things good, make sure dashRecoverTime runs out nearly as soon as the force of the dash is reduced to a considerable amount.
  ○ *PreDashRecoverTIme* is the time less than the dashRecoverTime, by which control is given back so that the character can line up the landing. Ideal value is somewhere close to dashRecoverTime - 0.15.
  ○ *MicroscopicPauseTime* is the time for which the game pauses when the dash is started. (Just for game feel).

Dash Properties---	
Number Of Dashes	1
Dash Force	3000
Dash Recover Time	0.3
Pre Dash Recover Time	0.2
Microscopic Pause Time	0

● **Grab Properties**
  ○ *IsGrabbing* is true all the time when the character is grabbing something. (Cannot be edited, for debugging purposes).
  ○ *ClimbSpeedModifier* is the value by which Move Speed is multiplied and set as Climb Speed. A value of 1 means climb with the same speed as the move speed. ○ *ClimbJumpForceModifier* is similar to ClimbSpeedModifier but it is with climbJumpForce. A value of 1 means climb jump with the same force as the jump force.
  ○ *LedgeJumpForce* is the same as ClimbJumpForce but while grabbing a ledge. A value of 1 means jump with the same force as the jump force while grabbing a

ledge.

- *MaxStaminaPoints* are the stamina points that are available for consumption. The current stamina points are reseted to maxStaminaPoints once character touches the ground.
- *HoldStaminaConsumption* is the stamina consumed per frame while holding any object.
- *ClimbStaminaPoints* is the stamina consumed per frame while climbing on any object.
- *ClimbJumpStaminaPoints* is the stamina consumed at the time of jumping when the climb jump is started.
- *ClimbJumpTime* is the time by which the climb jump is expected to end. It is an experimental value (determined by trial and error) and should be close to the time taken for the climb jump to end.
- *CanGrab* is true when character canGrab any object else it's false.
- *TempGrabJumpDirection* is the direction in which the climbJumpAway force is applied. (x, y) means if the character is holding something in left, the force will be applied in (x, y) direction. If the character is holding something in the right, force will be applied in (-x, y) direction.

Grab Properties--
Is Grabbing	☐
Climb Speed Modifier	0
Climb Jump Force Modifie	0
Ledge Jump Force Modifi	0
Max Stamina Points	0
Hold Stamina Consumpti	0
Climb Stamina Consump	0
Climb Jump Stamina Cor	0
Climb Jump Time	0
Can Grab	✔
Temp Grab Jump Directic	X 1 Y 1

Animation Events:

PlayerController.cs provides several events like :

```csharp
public delegate void Dashed(bool started);
public static event Dashed EDashed;

public delegate void Grabbed(bool started);
public static event Grabbed EGrabbed;

public delegate void Jumped(bool goingUp);
public static event Jumped EJumped;

public delegate void Movement(bool isMoving);
public static event Movement EMovement;
```

By subscribing to these events, various functionalities can be started as soon as the action is started/ended. For example :

```csharp
void Awake()
{
    PlayerController.EDashed += ToggleDash;
}

void ToggleDash(bool started)
{
    if(started)
    {
        Debug.Log("OMG, Dash started");
    }
    else
    {
        Debug.Log("Dash Ended");
    }
}
```

This will print "OMG, Dash started" as soon as the dash key is pressed and "DashEnded" when dash ends.

Similar approach can be taken for animations.

If the ToggleDash function is called with started = true, play the dash animation and when it is called with started = false, end the dash animation.

There is an example Animation script in the Additional folder for getting started.