Freeflow Combat **Documentation**

Thanks for purchasing Freeflow Combat and trusting Pathiral

About:

Freeflow Combat is a combat system that allows fast, fluid and smooth transition from one enemy to another generating dynamic and immersive gameplay for all the combo lovers.

Getting Started:

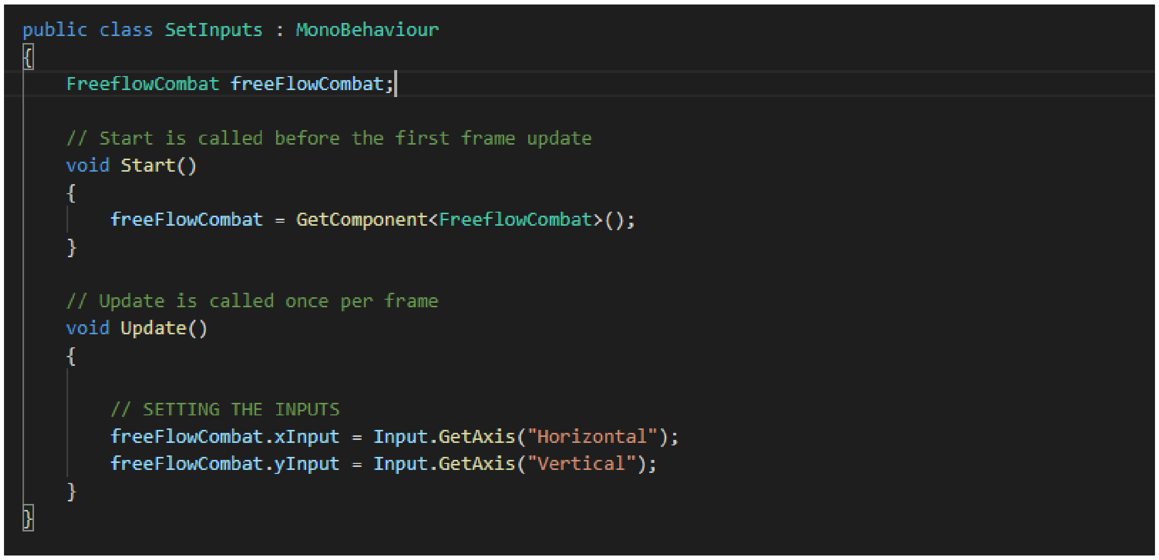
Like any other combat system that relies on camera, animations and input there are few things you will have to setup. Don't worry the steps maybe long but they're easy.

1. Add the ***Freeflow Combat*** component to your player character.

2. A **Character Controller** and **Animator** components will be automatically added. Fix the character controller properties to your liking to make it fit the player character such as *center*, *height* and *radius* just like in any other game. Also, create an empty *Animator Controller (right click in project space)* and drag it to the controller slot in the Animator component *(we'll come back to this later)*. **It's also highly recommended to turn off root motion.**

3. Now in the Freeflow Combat component, make sure the *[Auto Get Camera]* property is checked. This'll automatically get your game camera on start with no work from you. If you're dynamically spawning the camera on runtime then turn the Auto Get property to false and a [Camera] property will popup below where you can manually set the spawned camera object.

4. The next two properties *[X Input]* and *[Y Input]* are the values of the input system (horizontal and vertical). So we need to set those via another script <u>*every frame*</u>. Create a new script call it **SetInputs** or anything else really. Inside the script:



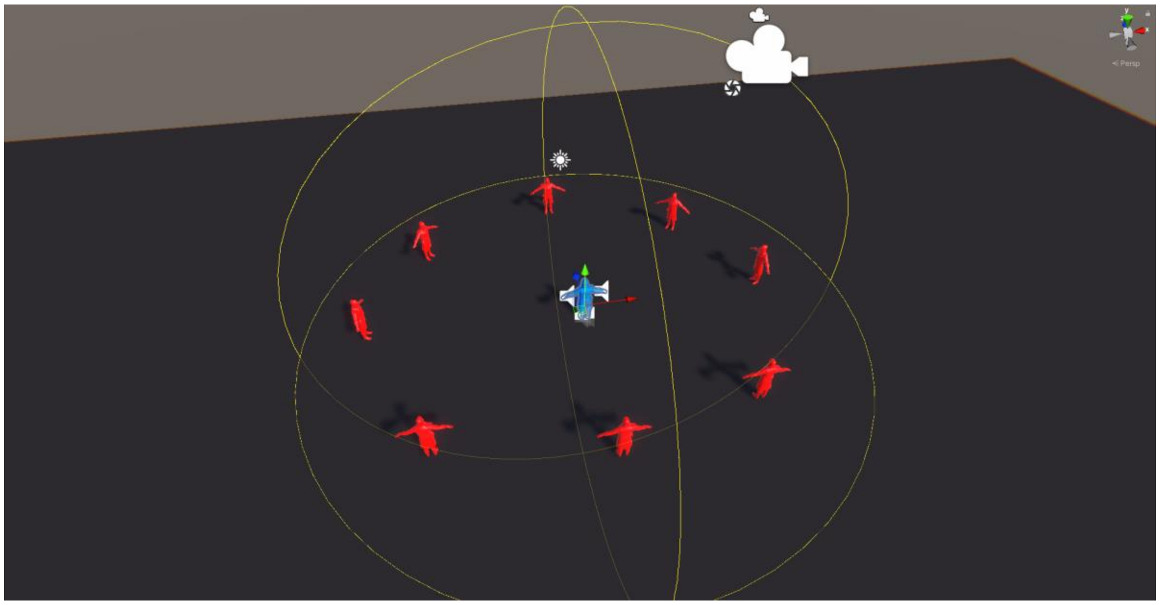
```
public class SetInputs : MonoBehaviour
{
    FreeflowCombat freeFlowCombat;

    // Start is called before the first frame update
    void Start()
    {
        freeFlowCombat = GetComponent<FreeflowCombat>();
    }

    // Update is called once per frame
    void Update()
    {

        // SETTING THE INPUTS
        freeFlowCombat.xInput = Input.GetAxis("Horizontal");
        freeFlowCombat.yInput = Input.GetAxis("Vertical");
    }
}
```

This is using the old input system if you're using the new input system. Find it's equivalent [here](). Remember, you can use both input systems together in project settings -> Player -> Input handling, to make it easier.

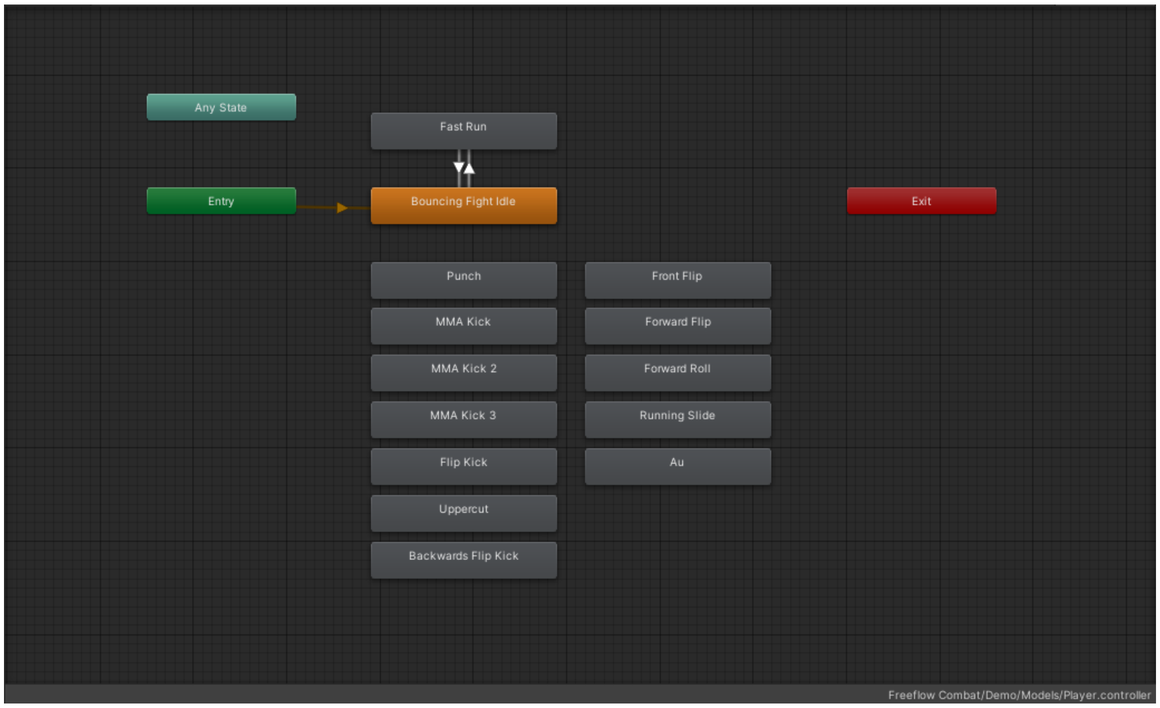
5. Now put this script on the player where the Freeflow Combat script exists.

6. Now set the *[Enemy Layers]* to the layers of enemies.

7. Set the *[Detection Radius]* to the range you would like to detect enemies. It's visualized in the scene view as a yellow wire sphere as so:



8. Now for the *[Idle Anim Name]*, **pay close attention to this**. Freeflow Combat comes with it's own animation manager to facilitate smooth animation playing and ease of use for other developers. Any property that requires you to set an animation name is actually telling you to set it's *animation state name* which is the animation name inside the animator controller. Basically this:

   These are all my animations inside the animator controller. So my idle animation name is Bouncing Fight Idle so that's what I type in inside the property and that's it! You don't even connect transitions together. You can also change the name of the animation completely from the top right corner after clicking on one of the animation blocks.

9. *[Scripts to Disable]* insert all your scripts that you want to disable when attacking and they'll automatically re-enable when attack is finished. Your movement scripts should definitely be here as moving while attacking will break things.

10. **Now go to Attack Tab.**

11. The first property is *[Attack Animations]* which is an array that contains two properties: *[Animation Name]* and *[Attack Distance]*. Just like we made it clear in the few previous steps, simply insert the names of the animations of the animator controller without connecting them. The second property *[Attack Distance]* is the distance you want to stop at from the enemy when this animation is chosen to be played.

12. **Now go to Traversal Tab.**

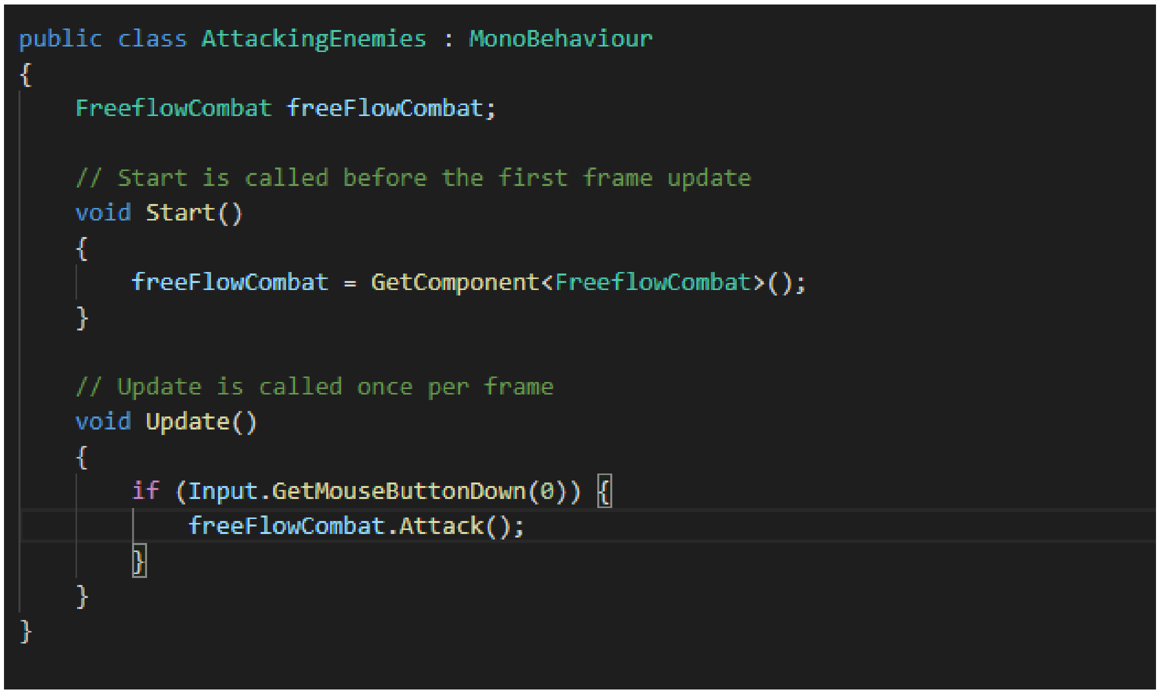
13. *[Traversal Time]* set the amount of time in seconds to traverse or move to the enemy. Set it to whatever suits you and your animations.

14. *[Use Traversal Animations]* If set to true a bunch of other properties will popup and this makes your character play animations while traversing and when reaches the attack distance then it plays the attack animation chosen. So make sure you populate this array with the animation state names of traversing that you want.

15. *[Apply Traversal Anim Dist]* Traversal animations will only play if distance between enemy and player equals or bigger than this value.

16. Create enemy cubes around the player and set their layers to the same *[Enemy Layers]* we set in the inspector. Also add the component **FreeflowCombatEnemy** which lets the system know this is an enemy. It contains one property *[Is Attackable]* from which you can control whether this enemy is eligible to be attacked or not via scripting.

17. Now we need the player to attack the cubes. Make a new script called **AttackingEnemies** and add it to the player:



```
public class AttackingEnemies : MonoBehaviour
{
    FreeflowCombat freeFlowCombat;

    // Start is called before the first frame update
    void Start()
    {
        freeFlowCombat = GetComponent<FreeflowCombat>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0)) {
            freeFlowCombat.Attack();
        }
    }
}
```

Attack() is a public method that detects surrounding enemies based on camera and input (x, y) and goes for the attack.

18. Now play the game and left click the mouse, the player should go on and attack the enemies based on direction.

Important Public Properties:

- Attack() -> the attack method that’s responsible for attacking.

- StopAttacking() -> stop method for both traversal and attacking.

- isAttacking -> returns either true or false. Returns true when the actual attack animation is playing. False if not.

- isTraversing -> returns either true or false. Returns true if traversing/moving to enemy location. False if already reached location and isn’t traversing.

- currentTarget -> returns a value of type FreeflowCombat which is the current targeted enemy traversing to or already attacking.