AFPC Manual

AFPC - Advanced First-Person Controller allows you to create a common controller for first-person games with movement, health-damage-death mechanics and few regular effects.

Quick setup

1. Go to AFPC - Resources - Prefabs
2. Place “Hero” prefab on the scene.
3. Assign your Camera in the “Overview” section.
4. Optional assign HUD component.
5. Press Play.

Script Reference

public class Hero : MonoBehaviour

Example class with AFPC implementation.

public class HUD : MonoBehaviour

Optional example UI interface for Hero class.

public class Overview

Mouse looking, aiming, shaking class.

public virtual void AllowLooking ();

Allow the controller to read looking input values and rotate the camera.
By default for common FPS games.

public virtual void BanLooking ();

Ban controller to read looking input values and rotate camera.
Use this if you want to block the user's ability to look around.

public virtual void AllowAiming ();

Allow the user to change camera FOV to view far objects.

public virtual void BanAiming ();

Ban the user to change camera FOV to view far objects.
The camera FOV value moves forward to the "default FOV" value.

public virtual void AllowShaking ();

Allow camera shaking by lens shifting. Required "Physical camera" mode on.

public virtual void BanShaking ();
Ban camera shaking by lens shifting.

public virtual void Follow (Vector3 target);
Follow the camera to the controller with offset.

public virtual void Looking ();
Rotate the camera with looking input values.
Using it as a "Mouse look" in common cases.

public virtual void Aiming ();
Changing the camera FOV value or return to the default FOV value;

public GameObject Search ();
Raycast in the forward direction to search some objects.
Good practice to use it for shooting or interaction.

public virtual void Shaking ();
Control the camera lens shift values.

public virtual void Shake (float value);
Shake the camera lens with value.

public void RotateRigigbodyToLookDirection (Rigidbody rb);
Rotate rigidbody to looking direction.

public class Movement
Move, Jump, Run class.

public virtual void Initialize ();
Initialize the movement. Generate physic material if needed. Prepare the rigidbody.

public virtual void AllowMovement ();
Allow the user to move.

public virtual void BanMovement (bool isStopImmediately = false);
Ban the user to move. Optional, immediately stop the rigidbody.

public virtual void AllowRunning ();
Allow the user to move faster.

public virtual void BanRunning ();
Ban the user from moving faster.

public virtual void AllowJumping ();
Allow the user to jump up.

public virtual void BanJumping ();
Ban the user from jumping up.

public void AssignLandingAction (UnityAction action);
Perform an action when the character was landed.

public virtual void AllowAirControl ();
Allow the user to change movement direction in the air.

public virtual void BanAirControl ();
Ban the user to change movement direction in the air.

public float GetEnduranceValue ();
Current endurance value.

public bool IsGrounded ();
Is this controller on the ground?

public virtual void Accelerate ();
Physical movement. Better use it in FixedUpdate.

public virtual void Jumping ();
Jumping state. Better use it in Update

public virtual void Running ();
Running state. Better use it in Update.

public class Lifecycle
Health-damage-death mechanic class.

public virtual void Initialize ();
Set maximum health and shield in the start.

public bool Availability();
Check the availability of this character.

public virtual void Activate ();
Activate the character.

public virtual void Deactivate ();
Deactivate the character.

public virtual void SetMaximumHealthAndShield ();
Restore the health and shield to the maximum.

public virtual void SetMinimumHealthAndShield ();
Drive the health and shield values to the 1.

public float GetHealthValue ();
Current health of the character.

public void SetHealthRecoveryRate (int value);
The health of the character will increase in 1 every "value" frames.

public virtual void AllowHealthRecovery ();
Allow this character to recover health.

public virtual void BanHealthRecovery ();
Ban this character to recover health.

public float GetShieldValue ();
Current shield of the character.

public void SetShieldRecoveryRate (int value);
The shield of the character will increase in 1 every "value" frames.

public virtual void AllowShieldRecovery ();
Allow this character to recover health.

public virtual void BanShieldRecovery ();
Ban this character to recover health.

public bool IsFrenzy ();
Check the Frenzy state.
The Frenzy state is used to give your users a special state when his health level is low.

public void SetFrenzyThreshold (float value);
Set a minimum health threshold for the frenzy state.

public virtual void Runtime ();
Recovering health and shield.

public virtual void Damage (float value);
Damage the character. The shield will be damaged first.

public void AssignDamageAction (UnityAction action);
Perform an action when the character was damaged.

public virtual void Heal (float value);
Heal the character.

public void AssignHealAction (UnityAction action);
Perform an action when the character was healed.

public virtual void Respawn();
Activate the character and restore health and shield.

public virtual void Death ();
Deactivate the character and set health and shield to the minimum.

public void AssignDeathAction();
Perform an action when the character dies.

Support
Check the Publisher Page for contact information.