➔ Contents

Pro 3D Camera

System Reference Guide

Thank you for your purchase.

This small investment will make your life so much easier.

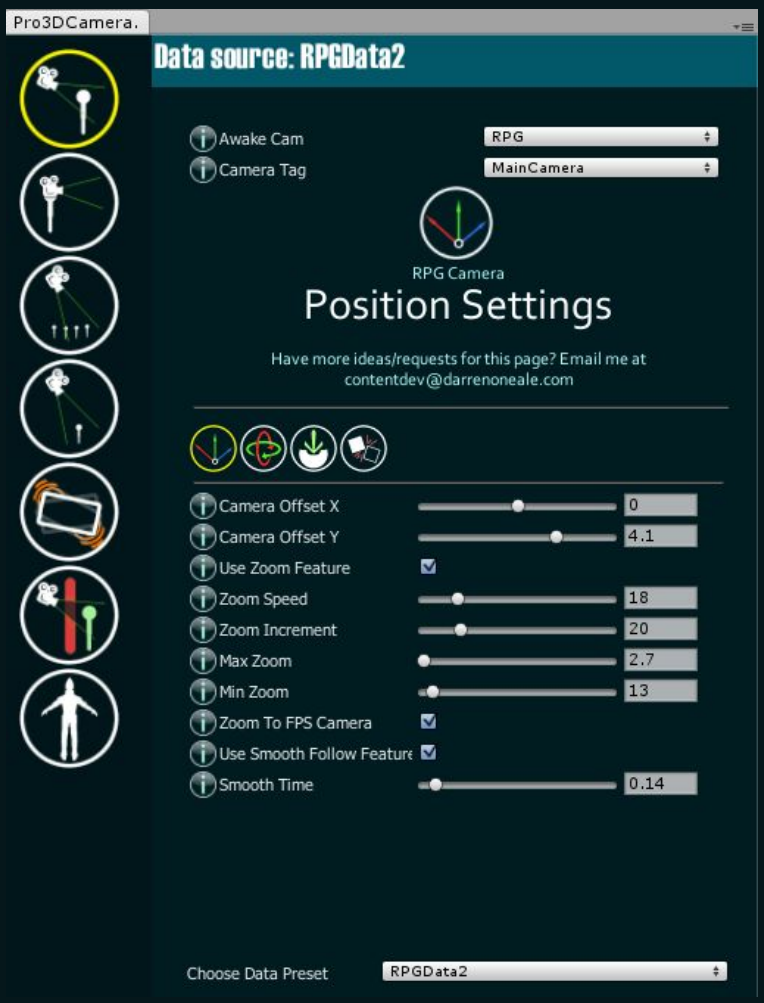
---

4 Cameras
1 Player Controller
Over 70 Settings
Easy Editor
Save Custom Settings Presets
API Documentation

Contents

- Overview
- The RPG Camera
- The FPS Camera
- The Top Down Camera
- The RTS Camera
- The Player Controller
- Shake Settings
- Camera Data Presets
- The Obstruction Handler
- Settings Detail
- API Docs and Examples
- Version Notes

➔ Contents



Pro 3D Camera

Pro Camera 3D was designed with the user in mind. You will be able to work in a clean, organized editor that offers you only the settings you care about - reducing redundancy and clutter as much as possible. You will have four popular camera styles to choose from with over 50 settings to play around with.

Use the RPG camera to follow closely behind the target. Complete with smoothing and orbiting options.

Use the FPS camera to gain a Point-of-View perspective - great for shooters and simulations.

Use the Top Down camera to follow the target from a distance while getting a breathtaking view of the environment.

Use the RTS camera if you want the player to pan environments without any particular target to focus on.

One of the unique features with this camera package is the ability to produce infinite amounts of camera data. So if you have multiple RPG preferences, you can switch between the RPG data sets effortlessly until you decide the data you prefer the most.

Don't assume you need to use all of these cameras in each project. While you are only modifying a single camera script - it was designed to give you options across all of your projects. Some cameras share functionality, but you can mostly stick to one camera based on the style you are shooting for. Below is a legend of the editor that comes with the package.


Position


Orbit


Input


Pan

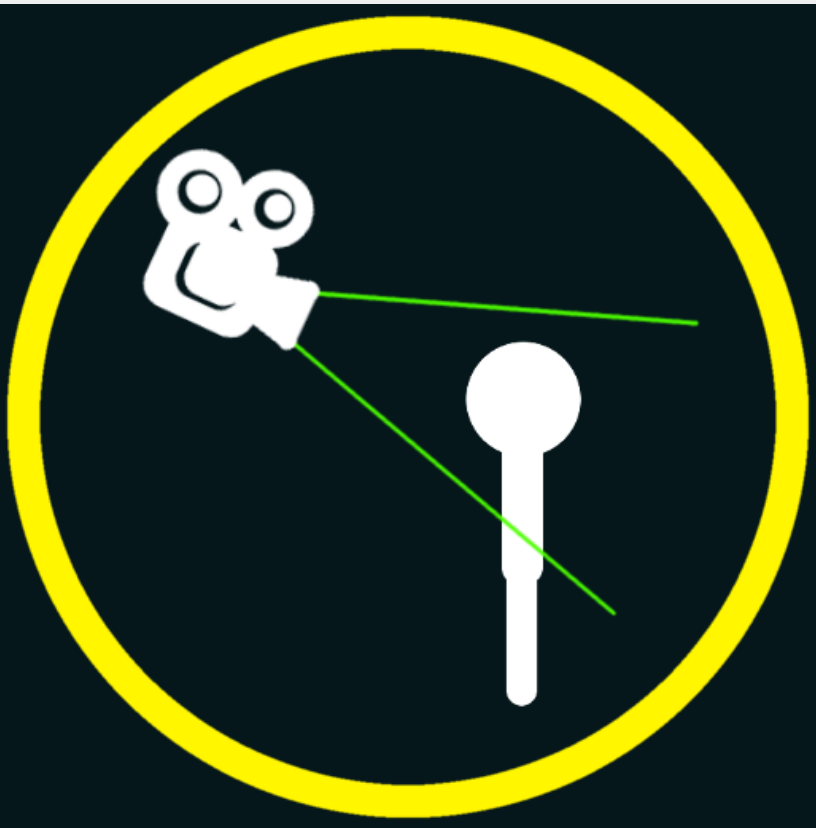

Collision


Shake Creator


Obstruction Handler


Player Controller

➔ Contents



The RPG Camera

The RPG camera is great for close quarters third-person-view perspective situations.

Settings offered for this camera:

Adjust camera position offset, zooming, smoothing, and optionally zoom directly into the FPS camera.

Adjust angle thresholds, orbit speeds, and utilize the auto-return feature.

Modify input settings for orbiting and zooming.

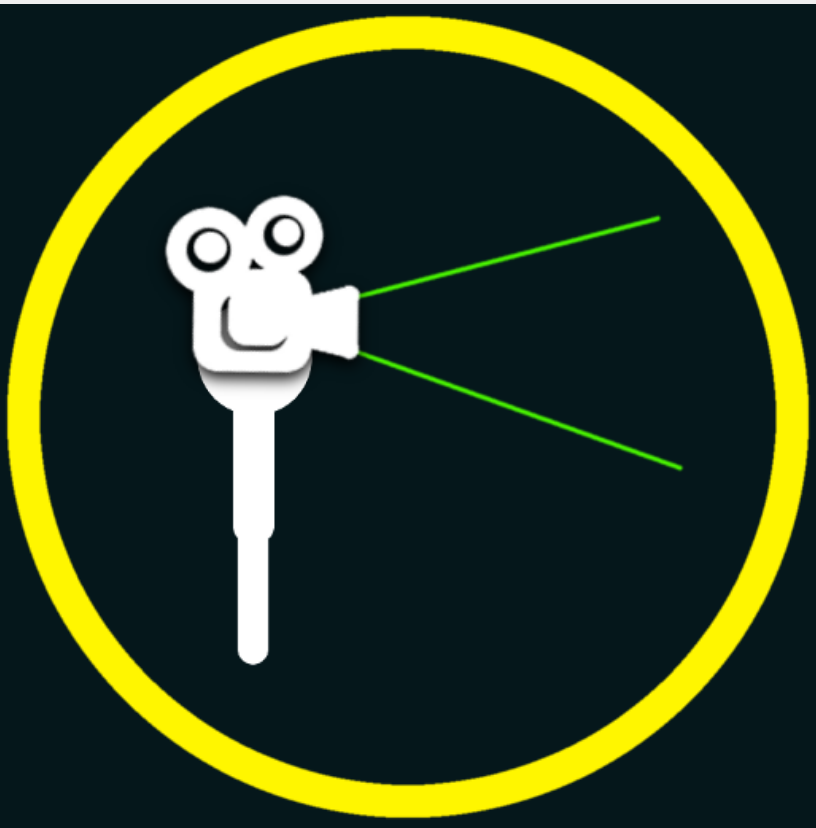
Toggle wall collision and occlusion and modify the collision layer.

What kind of game are you making? You may want to use the RPG camera if your genre is:

- ➔ Stealth
- ➔ RPG/MMORPG
- ➔ Racing
- ➔ Third Person Shooter

These are just some suggestions, of course, and you could use any camera for any genre!

➔ Contents



The FPS Camera

The FPS camera is useful for point-of-view projects.

Settings offered for this camera:

Adjust camera position offset, bounce options, smoothing, and optionally zoom directly out into the RPG camera.

Adjust X/Y axis sensitivities and smoothing options for responsive or groggy look speeds.

What kind of game are you making? You may want to use the FPS camera if your genre is:

- ➔ First Person Shooter
- ➔ Survival
- ➔ Racing
- ➔ Simulation

These are just some suggestions, of course, and you could use any camera for any genre!

➔ Contents

Top Down Camera

The Top Down camera is the way to go when you want the player to catch a bird's-eye glimpse of your environment while following the target.

Settings offered for this camera:

Adjust zooming and smoothing options.

Adjust angle thresholds, orbit speeds, and utilize the auto-return feature.

Modify input settings for orbiting and zooming.

What kind of game are you making? You may want to use the Top Down camera if your genre is:

- ➔ Platformer
- ➔ Third Person Shooter
- ➔ RPG

These are just some suggestions, of course, and you could use any camera for any genre!

➔ Contents

The RTS Camera

The RTS camera is great if you want the player to pan environments without any particular target to focus on.

Settings offered for this camera:

Adjust camera zooming, movement boundaries, and elasticity settings.

Adjust angle thresholds and orbit speeds.

Modify pan speed and pan drag after input is released. Also determine if panning should be inverted or not.

Modify input settings for orbiting, panning and zooming.

What kind of game are you making? You may want to use the RTS camera if your genre is:

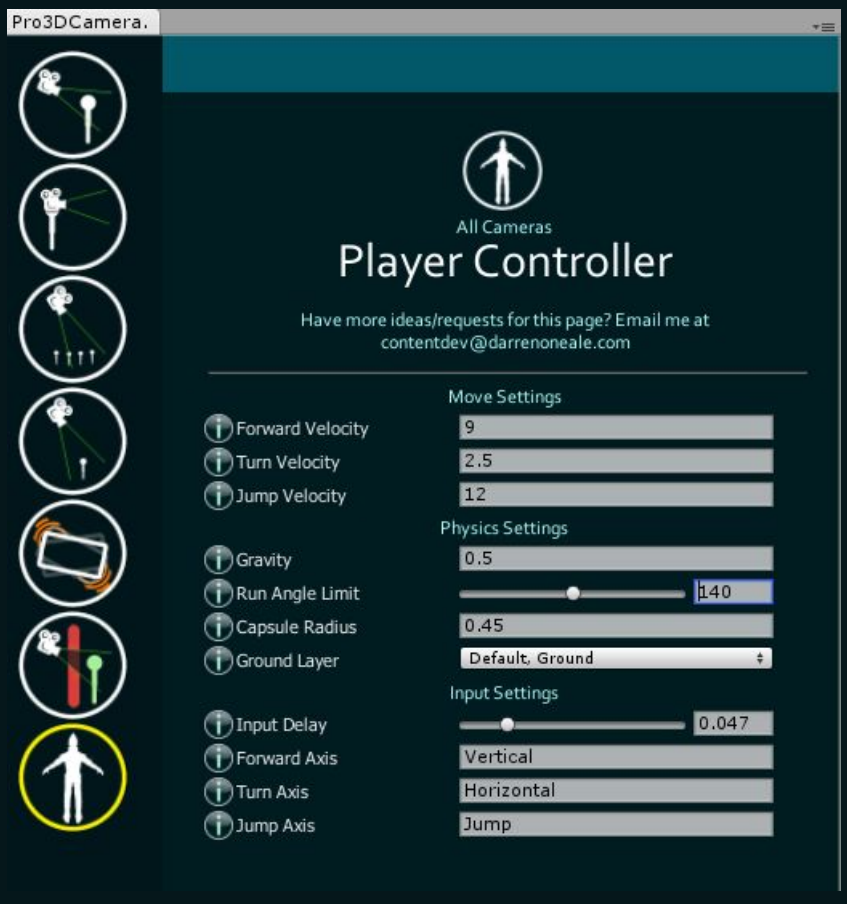
- ➔ Tower Defense
- ➔ Real-time Strategy
- ➔ Simulation

These are just some suggestions, of course, and you could use any camera for any genre!

➔ Contents

The Player Controller

The player controller is intended to work in conjunction with the camera system. It is a custom player controller, which is open source and fully extendable. It is not necessary that you choose to use this player controller, though, if you already have your own. The camera system will work on any object - player controller or not. It is worth mentioning, however, that the FPS bounce (head bob) feature relies on variables in the player controller, such as whether the player is moving or grounded. So this feature alone will lose functionality if using a different player controller. It is still possible to get the bounce feature to function either way as long as information is provided for movement and grounding.



Significant upgrades have been made to the player controller. It can now handle any terrain consistently without becoming airborne and causing buggy-looking behavior with animations and so forth.

For the most part, the settings in the player controller are self explanatory. It is divided into three sections:

- ➔ Movement
- ➔ Physics
- ➔ Input

The only noteworthy section of the settings is the capsule radius field. The controller calculates the *grounded* state by Physics.CheckSphere(). This method requires a location for the capsule's top, the capsule's bottom and the capsule's radius. Lucky for you, the capsule top and capsule bottom is pre-calculated for you - so you only need worry about the radius.

It can be difficult getting the *grounded* state to be as accurate as possible. For best results, set the capsule radius field to be slightly lower than the actual capsule collider radius attached to your player object.

➔ Contents

Shake It Up!

You can create your own shake sequences by opening the shake settings page.

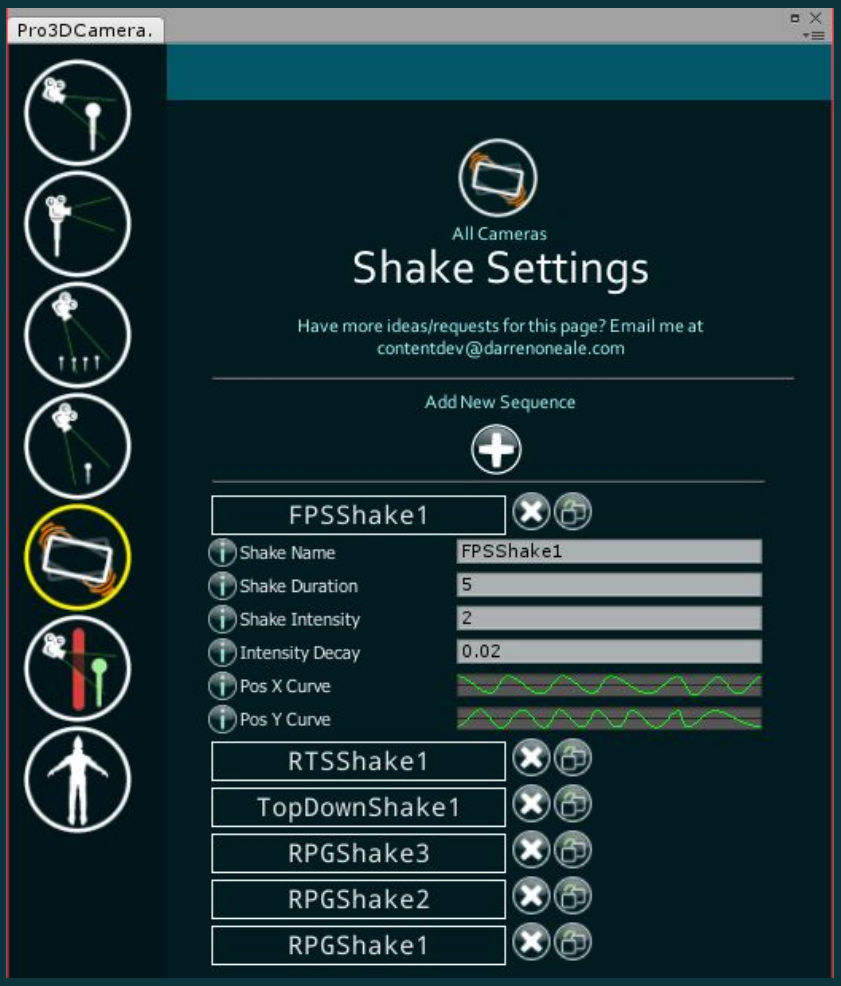
You will need to:

- ➔ Name the new sequence
- ➔ Modify the duration
- ➔ Set the intensity (multiplier for the curves)
- ➔ Set the intensity decay (how fast intensity approaches zero)
- ➔ Setup the X and Y position curves

The curves will determine how the camera moves over time. The amplitude of your curves will be different based on the camera you are shaking. Curves with an amplitude of 1 may work great for RPG and FPS, but the RTS and Top Down cameras - due to their distance - will need an amplitude of 5 more more. Of course, you can always just play with the shake intensity and shake decay settings to unleash the power you are looking for.

After creating the sequence and setting the curves, you can hit the play button and test the shake on each of your cameras. A button will appear at run-time for you to test effortlessly.

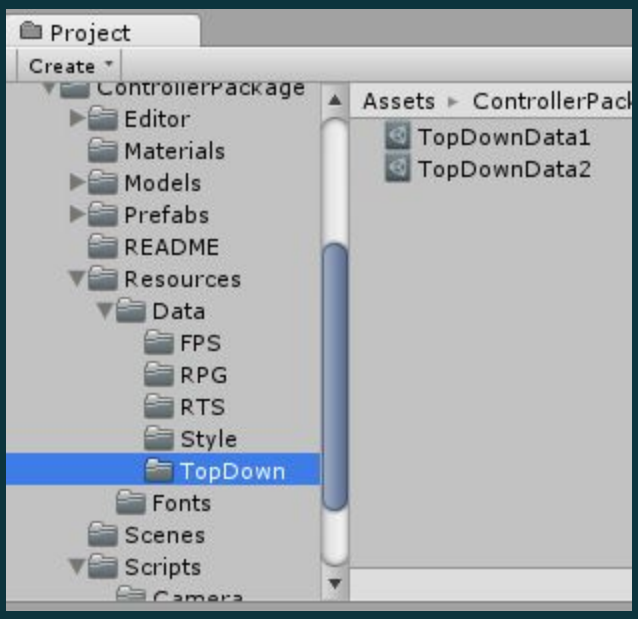
To use your shake sequence during an event specific to your game, you will need to call the shake method.



➔ Contents

Camera Data

The way your new camera gathers data is through accessing scriptable objects. The package will provide you with a few options to get started, but when you get comfortable, you may end up wanting to save your settings - but experiment with new styles. To do that, you can just get into the Data folder and duplicate one of the data objects to get a starting point. This is useful when you want to quickly iterate different camera settings until you find something perfect for your project. Switching between camera data does not require any extra effort on your part.



You will see an option at the bottom of your camera editor to change data presets. By accessing the dropdown, you will see all available data options based on the camera page you are under. In other words, if you are on the Top Down settings page, you will see all of your top down data in the dropdown. Simply choose one, and voila! Your camera will make necessary adjustments based on the data object that is selected here.

➔ Contents

Obstruction Handler

The obstruction handler is another optional feature that fades out obstructions between the camera and the player, while optionally changing the target's color. To work properly, the obstruction handler needs to be on the same object as the Mesh/SkinnedMeshRenderer component of the target. It will only fade objects that have materials with color properties. Similarly, the target's materials need to have color properties.

After placing the obstruction handler component on your object, you will need to link it to the system data manager - which controls functions for how the camera operates.

➔ Contents

Settings Detail

	**Camera Offset**	Dictates the horizontal and vertical offset of the camera's look position, relative to the target.	RPG, FPS
	**Zoom**	Enabling this provides options for zooming in and out on the target.	RPG, FPS, RTS, Top Down
	**Zoom Speed**	How fast does the camera zoom through each zoom increment?	RPG, FPS, RTS, Top Down
	**Zoom Increment**	For each zoom input, how far should the camera zoom? Think of this like a zoom sensitivity, but not speed.	RPG, FPS, RTS, Top Down
	**Max Zoom**	The closest position allowed, relative to the target.	RPG, FPS, RTS, Top Down
	**Min Zoom**	The furthest position allowed, relative to the target.	RPG, FPS, RTS, Top Down
	**Zoom to FPS/RPG**	Allows the camera to transition back and forth between FPS and RPG camera types via zooming.	RPG, FPS
	**Smooth Follow**	Gives the camera some drag when following the target.	RPG, Top Down
	**Smooth Time**	The time taken for the camera to reach it's destination position.	RPG, Top Down
	**Initial Position**	Gives an initial position for the RTS camera.	RTS
	**Boundaries**	Provides means for restricting the RTS camera to a box region.	RTS
	**Min Bounds**	The minimum threshold the camera is permitted to travel on the X-Z axis.	RTS
	**Max Bounds**	The maximum threshold the camera is permitted to travel on the X-Z axis.	RTS
	**Elastic Boundaries**	Gives the camera a bounce threshold when traveling beyond the permissible boundary region.	RTS
	**Boundary Elasticity**	The threshold the camera is permitted to travel beyond the boundary region.	RTS

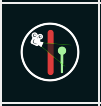
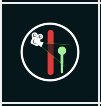
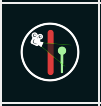
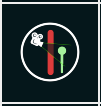
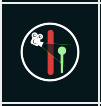
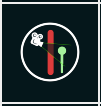
➔ Contents

	**Ground Layer**	The layer the RTS camera looks for when determining a distance, or height, to maintain.	RTS
	**Bounce Feature**	Enable this if you want to use head bobbing when running.	FPS
	**Bounce Frequency**	The speed of the head bob.	FPS
	**Bounce Amplitude**	The intensity of the head bob.	FPS
	**Max X Angle**	The highest angle the camera can rotate 'up and down' around the player.	RPG
	**Min X Angle**	The lowest angle the camera can rotate 'up and down' around the player.	RPG
	**X Angle**	On the top down and RTS cameras, the angle at which the camera looks down on the environment.	RTS, Top Down
	**Orbit Feature**	Toggles the orbit feature, which allows the player to orbit around the target.	RPG, RTS, Top Down
	**Orbit Speed**	The speed at which the camera rotates around the target on the X and Y axes.	RPG, RTS, Top Down
	**Orbit With Target Feature**	Enabling this keeps the camera aligned with default angles.	RPG, Top Down
	**Auto Return Angle**	Enabling this will automatically align the camera's angle according to 'Default Angle'.	RPG, Top Down
	**Default Angle**	The angle the camera will automatically move to.	RPG, Top Down
	**Angle Return Time**	The time it takes for the camera to move from it's current angle to 'Default Angle'.	RPG, Top Down
	**FPS Sensitivity**	The speed the camera will look around.	FPS
	**FPS Smooth Look**	Enable this if you do not want the look speed to be instantaneous.	FPS
	**FPS Smooth Look Speed**	The speed the camera looks around in response to input.	FPS
	**Orbit Input**	The KeyCode that provides input to orbit the camera around the target.	RPG, Top Down, RTS

➔ Contents

	**Pan Input**	The KeyCode that provides input to pan the camera around the environment.	RTS
	**Zoom Input**	The string value Input Axis name that provides input to zoom in and out on the target.	RPG, Top Down, RTS
	**Collision Feature**	Enable if you want the camera to respond to collision and occlusion events.	RPG
	**Collision Layer**	The camera will collide with objects on these layers. Objects on these layers will not occlude the target.	RPG
	**Collision Smooth Time**	The time taken for the camera to reach it's destination position while colliding with an object.	RPG
	**Draw Collision Lines**	Enable to view the collision lines from the editor view. Collision lines represent the location of the camera with no collision, and with collision.	RPG
	**Collision Padding**	When colliding, the camera will move forward an additional amount according to this value.	RPG
	**Close Quarters Feature**	If enabled, the camera will move up and look down on the target when the target's back is against the wall.	RPG
	**Close Quarters Distance**	The distance the camera is away from the target before the camera moves upward.	RPG
	**Close Quarters Height**	The height the camera moves to when close quarters is activated.	RPG
	**Close Quarters Fade Feature**	If enabled, the camera's target will fade to your desired alpha when the camera is within a certain distance. This feature only works on targets whose materials use the Standard Shader - or a variation of it.	RPG
	**Distance to Fade Target**	The distance the camera must be to the target before the target fades.	RPG
	**Target Fade Alpha**	The opacity level of the target when the camera comes within the specified range.	RPG
	**Pan Speed**	The distance the camera travels when responding to pan input.	RTS
	**Pan Drag**	The time taken for the camera to reach the desired pan destination.	RTS
	**Invert Pan Direction**	Enable this to invert the pan direction. When enabled, the camera will pan in the opposite direction of the input. i.e. If dragging right, the camera will move left.	RTS
	**Shake Name**	The name of this shake sequence. This name is what will be called via the API when you want to shake the camera with this sequence.	RPG, FPS, RTS, Top Down

➔ Contents

	**Shake Duration**	The length of time this shake sequence will execute for.	RPG, FPS, RTS, Top Down
	**Shake Intensity**	The multiplier of the X and Y curves when shaking the camera.	RPG, FPS, RTS, Top Down
	**Intensity Decay**	The rate at which the intensity decreases. Higher values will make it appear as though the sequence ends early.	RPG, FPS, RTS, Top Down
	**Shake Curves**	The curves that the camera's X-Y positions will loop through for the duration of the sequence.	RPG, FPS, RTS, Top Down
	**Obstruction Handler Feature**	Enable if you want objects to fade out when they occlude the camera's target.	RPG, RTS, Top Down
	**Obstruction Layer**	Objects on this layer will fade out when the player is occluded by them. To work, these objects will need to use the Standard Shader (or an extension of it).	RPG, RTS, Top Down
	**Obstruction Fade Speed**	The speed at which objects fade when they occlude the player from the camera view.	RPG, RTS, Top Down
	**Min Obstruction Alpha**	The alpha of the object's shader color property when fading out. Lower values will make the obstructions easier to see through.	RPG, RTS, Top Down
	**Change Target Color**	Enable if you want the camera's target to change color when objects occlude him.	RPG, RTS, Top Down
	**Obstructed Color**	The color of the target when objects occlude him.	RPG, RTS, Top Down
	**Target Color Intensity**	The intensity of the color property of the target's occluded color.	RPG, RTS, Top Down
	**Target Fade Speed**	The speed that the target's color changes when occluded.	RPG, RTS, Top Down
	**Forward Velocity**	The speed that the player moves on it's Z axis.	RPG, FPS, RTS, Top Down
	**Turn Velocity**	The speed that the player turns left and right.	RPG, FPS, RTS, Top Down
	**Jump Velocity**	The speed with which the player jumps.	RPG, FPS, RTS, Top Down
	**Gravity**	The acceleration rate at which the player is pulled toward the ground when airborne.	RPG, FPS, RTS, Top Down
	**Run Angle Limit**	The maximum angle the player can run against. A good value for this is 140.	RPG, FPS, RTS, Top Down

➔ Contents

	**Capsule Radius**	Used to determine if the player is grounded. This value should be slightly lower than the capsule collider radius that is on your player. This value will be different based on your player model.	RPG, FPS, RTS, Top Down
	**Ground Layer**	Objects on this layer will set the player to grounded if the player is on them.	RPG, FPS, RTS, Top Down
	**Input Delay**	This is the delay from when a button is pressed to when a response is triggered.	RPG, FPS, RTS, Top Down
	**Forward Axis**	The string value from input settings that will move the player forward.	RPG, FPS, RTS, Top Down
	**Turn Axis**	The string value from input settings that will turn the player left and right.	RPG, FPS, RTS, Top Down
	**Jump Axis**	The string value from input settings that will allow the player to jump.	RPG, FPS, RTS, Top Down

API Docs and Examples

The following functions and properties are created for your convenience. It is expected that some of these settings will be modified at runtime as a result of a game feature or via settings menu. If there is something else you would like to modify at runtime - and think it would be runtime safe - email me at contentdev@darrenoneale.com and we can discuss any additional API features for future updates.

Functions	Properties
OffsetCamera SetCameraTarget SetCameraType SetSensitivity SetDistance SetObstructionHandlerActive SetOrbitInput SetOrbitSpeedX SetOrbitSpeedY ShakeCamera	ActiveCamera BounceFrequency CameraOffsetX CameraOffsetY IsColliding InvertPan LookSpeed MaxBounds MinBounds PanInput PanSpeed SensitivityX SensitivityY

➔ Contents

Functions

OffsetCamera

void CameraControl.OffsetCamera(float x, float y)

- ➔ Modifies the RPG camera offset vector
- ➔ X and Y values locked [-10, 10]
- ➔ Permanently modifies data to X and Y
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.OffsetCamera(2, 1); //Shifts the camera two units to the right relative to the target
}
```

SetCameraTarget

void CameraControl.SetCameraTarget(Transform t)

- ➔ Assigns the camera's focus to a new target 't'
- ➔ Attempts to assign a new PlayerController variable
- ➔ If the new target does not have a PlayerController component, some functionality may be lost

```
using Pro3DCamera;
CameraControl camControl;
  void Start()
  {
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.SetCameraTarget(transform); //assigns the camera to this game object
  }
```

➔ Contents

SetCameraType

void CameraControl.SetCameraType(CameraType cam)

- ➔ Changes the type of camera being used
- ➔ Initializes the camera settings for ‘cam’
- ➔ Hides the target if FPS is chosen

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
}
void Update()
{
    if (Input.GetKeyDown(KeyCode.T))
    {
        camNum++;
        if (camNum > 3)
            camNum = 0;
        switch (camNum)
        {
            case 0: camControl.SetCameraType(CameraControl.CameraType.RPG); break;
            case 1: camControl.SetCameraType(CameraControl.CameraType.FPS); break;
            case 2: camControl.SetCameraType(CameraControl.CameraType.RTS); break;
            case 3: camControl.SetCameraType(CameraControl.CameraType.TOP_DOWN); break;
        }
    }
}
```

SetSensitivity

void CameraControl.SetSensitivity(float x, float y)

- ➔ Modifies FPS sensitivity
- ➔ Values locked [10, 500]
- ➔ Permanently modifies data to ‘x’ and ‘y’
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.SetSensitivity(200, 200);
}
```

➔ Contents

SetDistance

void CameraControl.SetDistance(float _dist)

- ➔ Sets a target distance for the current camera to move to.
- ➔ _dist is locked between the current camera's max and min zoom values [maxZoom, minZoom]

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.SetDistance(20); //moves camera 20 units away at the camera's smooth speed
}
```

SetObstructionHandlerActive

void CameraControl.SetObstructionHandlerActive(bool _active)

- ➔ Sets the obstruction handler to be active based on the value '_active' that was passed.
- ➔ All components with the obstruction handler will be set to _active.
- ➔ If _active is set to false, all existing obstructions will fade in and be removed.
- ➔ Does nothing if obstruction data cannot be found.

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.SetObstructionHandlerActive(true); //activates all Obstruction Handlers in the project
}
```

➔ Contents

SetOrbitInput

void CameraControl.SetOrbitInput(CameraType forCam, CameraData.InputOption input)

- ➔ Modifies orbit input based on the current active camera
- ➔ Permanently modifies data 'orbitInput'
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
camControl.SetOrbitInput(CameraControl.CameraType.RPG,CameraData.InputOption.RIGHT_MOUSE);
}
```

SetOrbitSpeedX

void CameraControl.SetOrbitSpeedX(CameraType forCam, float speed)

- ➔ Modifies X orbit speed based on the current active camera
- ➔ Value locked [0.1, 50]
- ➔ Permanently modifies data 'xOrbitSpeed'
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.SetOrbitSpeedX(CameraControl.CameraType.RPG, 24);
}
```

ShakeCamera

void CameraControl.ShakeCamera(string _shakeName)

- ➔ Shakes the camera based on shake data specified by '_shakeName'
- ➔ Will exit upon failing to locate shake information

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.ShakeCamera("RPGShake1"); //RPGShake1 must exist in settings
}
```

➔ Contents

Properties

activeCamera

CameraControl.CameraType CameraControl.activeCamera

➔ Get
➔ Returns the camera currently in use by the camera control system.

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    Debug.Log("The active camera is "+camControl.activeCamera); //The active camera is RPG
}
```

bounceFrequency

float CameraControl.bounceFrequency

➔ Get/Set
➔ Modifies FPS Bounce Frequency
➔ Value locked [0.01f, 10]
➔ Permanently modifies 'bounce frequency' data
➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
}
void Update()
{
    if (Input.GetKeyDown(KeyCode.UpArrow))
        camControl.bounceFrequency += 0.1f;
    if (Input.GetKeyDown(KeyCode.DownArrow))
        camControl.bounceFrequency -= 0.1f;
}
```

➔ Contents

cameraOffsetX

float CameraControl.cameraOffsetX

- ➔ Get/Set
- ➔ Modifies RPG Camera Offset X
- ➔ Value locked [-10, 10]
- ➔ Permanently modifies 'offset x' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.cameraOffsetX = -3;
}
```

isColliding

bool CameraControl.isColliding

- ➔ Get
- ➔ Returns true if the camera is in a collision state.

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    if (camControl.isColliding)
        Debug.Log("The camera is colliding.");
}
```

➔ Contents

invertPan

bool CameraControl.invertPan

- ➔ Get/Set
- ➔ Toggles RTS camera invertPan
- ➔ Permanently modifies 'invertPan' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.invertPan = true;
}
```

lookSpeed

float CameraControl.lookSpeed

- ➔ Get/Set
- ➔ Modifies time taken for FPS camera to look
- ➔ Value locked [-0.01, 30]
- ➔ Permanently modifies 'lookSpeed' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
    camControl = Camera.main.GetComponent<CameraControl>();
    camControl.lookSpeed = 10;
}
```

➔ Contents

maxBounds

Vector2 CameraControl.maxBounds

- ➔ Get/Set
- ➔ Modifies the minimum X-Z position of the RTS camera
- ➔ Permanently modifies 'minBoundary' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
   camControl = Camera.main.GetComponent<CameraControl>();
   camControl.maxBounds = new Vector2(100, 100);
}
```

panInput

CameraData.InputOption CameraControl.panInput

- ➔ Get/Set
- ➔ Modifies pan input for the RTS camera
- ➔ Permanently modifies 'panInput' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
   camControl = Camera.main.GetComponent<CameraControl>();
   camControl.panInput = CameraData.InputOption.LEFT_MOUSE;
}
```

➔ Contents

panSpeed

float CameraControl.panSpeed

- ➔ Get/Set
- ➔ Modifies pan speed for the RTS camera
- ➔ Value locked [0.1, 500]
- ➔ Permanently modifies 'panSpeed' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
   camControl = Camera.main.GetComponent<CameraControl>();
   camControl.panSpeed = 340;
}
```

sensitivityX

float CameraControl.sensitivityX

- ➔ Get/Set
- ➔ Modifies responsiveness to FPS mouse input
- ➔ Value locked [10, 500]
- ➔ Permanently modifies 'XSensitivity' data
- ➔ Data can be readjusted from the editor

```
using Pro3DCamera;
CameraControl camControl;
void Start()
{
   camControl = Camera.main.GetComponent<CameraControl>();
   camControl.sensitivityX = 200;
}
```

➔ Contents

Version Notes

V3.0

New

General

- ➔ Mobile support/development discontinued. (I am unable to test on multiple devices)
- ➔ Editor window to modify over 70 settings across 4 different cameras
- ➔ The editor window speaks for itself - each setting has an info button next to it. When clicked, information about that setting can be viewed.
- ➔ A new test environment has been created to present extreme collision scenarios and obstacles for the player controller.
- ➔ API access for accomplishing various tasks on the camera at run-time
    - ❏ Adjust camera offset
    - ❏ Set new targets for the camera
    - ❏ Change the camera type
    - ❏ Set FPS sensitivity
    - ❏ Modify orbiting and panning input
    - ❏ Modify orbiting and panning speeds
    - ❏ Shake the camera
    - ❏ Set min and max bounds for the RTS camera
    - ❏ & Much More!

Cameras

- ➔ You can now choose to have the camera auto-revert to certain angles when not orbiting. For instance, the RPG camera angle can always rotate back to being behind the target - or any X-Y angle combo that you choose.
- ➔ You can now seamlessly transition between the RPG and FPS cameras when zooming in and out
- ➔ RTS camera now has settings for restricting camera location to boundaries
- ➔ RTS camera now has elasticity settings. You can choose to have the RTS camera “bounce” back to the boundary when the user releases pan input
- ➔ You can now set the initial position for the RTS camera - since it does not have a target to rely on.

Player Controller

- ➔ The test environment now uses Unity Technologies’s character asset, Ethan.

➔ Contents

Special Features

➔ New shake feature. Create your own camera shake sequences from the editor by modifying position curves, intensity, and decay values. Give them names and call them from the API method.

Collision

➔ You can easily add padding to your camera when it collides, pushing it forward based on the padding value you choose. This can reduce the amount of clipping in close quarters situations. Using high padding will never push the camera through the target.

➔ A toggleable feature has been added to the collision settings where when the camera is backed against a wall, it will get pushed upward and look down on the target. Distance to determine when this is activated, and the height the camera moves up to, are both values that can be tweaked.

➔ The camera's smoothing rate when colliding and not colliding are now independent of each other and can be modified from either the position settings or the collision settings.

➔ A new toggleable feature has been added where the camera's target will fade when the camera is too close. The user defines how close is "too close" and how much fade will be applied to the target. For now, fading can only be done on targets whose materials use the Standard Shader or some variation of it.

Improved

General

➔ Improved workflow. You no longer need to worry about saving camera data.

➔ All features and settings can be modified from an easy editor - accessible through the camera control script.

➔ Camera data is now handled more safely on the lower level (c++) side, through the use of scriptable objects

➔ Consolidated cameras into one script

Cameras

➔ Transitions between different cameras at run-time has been smoothed out

Collision

➔ Collision detection has been greatly improved - with more unique cases being cared for in volatile, clip-prone scenarios.

➔ Contents

Player Controller

➔ PlayerController handles bumpy terrain and hills more properly. The forward vector is based on the normal of the ground below, meaning the speed of the player will not be limited by slope of terrain.

➔ A maximum slope is set (so the player cannot run up walls). This value can be tweaked from the Player Controller section in the editor.

➔ To check for grounded, the player controller now uses Physics.CheckSphere(). This method has been tested against other methods and proves to be reliable.

Special Features

➔ The obstruction handler settings can be modified from the editor. There is now an API method which will allow you to set this feature to be active at runtime - since this feature operates independently of the camera control system. See the method here.