Ultimate FPS Camera

Home
http://www.visionpunk.com

Twitter
https://twitter.com/#!/VisionPunk

Asset Store
http://u3d.as/content/vision-punk/ultimate-fps-camera

Webplayer demo
http://www.visionpunk.com/products/ufpsc/webplayer/demo.html

Youtube preview
http://www.visionpunk.com/products/ufpsc/video/preview.html

Official F.A.Q.
http://www.visionpunk.com/products/ufpsc/faq/FAQ_Temp.html

Forum thread
http://forum.unity3d.com/threads/126886-Ultimate-FPS-Camera-RELEASED?goto=newpost

----

v1.4.0 RELEASE NOTES

>> IMPORTANT: BACKUP ALL EXISTING PRESETS AND SCRIPTS BEFORE INSTALL <<

Installation

Do NOT install this version over any previous version of Ultimate FPS Camera! This is a complete rewrite of the system and it is not backwards compatible. In other words: it's 100% certain to break your project. It's a good idea to install v1.4.0 into a clean project, read the manual and study new core concepts (especially the new event system) before getting started.

Recommended update steps:
1. Backup your existing project folder
2. Delete the "Ultimate FPS Camera" folder from your project
3. Open Asset Store, browse to Ultimate FPS Camera and click the Upgrade button
4. Read up on the new important concepts below and in the manual
5. Integrate the new system with your existing game scripts

General info

Ultimate FPS Camera v1.4, which has been a long time in the making, is a more or less complete rewrite of the system, taking into account all the user feedback, many of the feature requests and many, many reported bugs. The system works the same way to a large extent, and very different from before to some extent. v1.4, being a reboot, is pretty much a v1.0, so expect to re-learn many things (editor workflow is hopefully easier, scripting difficulty probably harder). And expect many new and exotic bugs.

Finally, I want to direct a huge **Thank You** to all the awesome users that have provided fantastic feedback, testing and ideas. Ultimate FPS Camera has come a long, long way since the first prototype release (which didn't even have support for firing the weapon) and is now a quite extensive system with tons of features. I couldn't have done it without you and I hope you'll have tons of fun with it!

Special notes

Decoupling
There has been a big effort to make the system more decoupled and component based. Ultimate FPS Camera previously had heavy cross linking between classes. This made it hard to opt out of specific feature sets (among other bad things). Looser coupling makes the system easier to maintain and extend long term. A casualty of this process has been the vp_FPSPlayer script, which is no more (its functionality was split into separate scripts to the point where there wasn't anything left in it). You may need to create your own dedicated player script, if needed, to store game specific functionality or data.

Event System
A new advanced event system has been introduced with four types of events for varying fields of application: **Messages**, **Attempts**, **Values** and **Activities** (the last one being quite powerful actually). The event system is delegate based for lightning-fast execution. It uses a central event handler class where all player events are declared. The event handler uses reflection (on startup) to automatically register user methods, making new event creation a fairly slim process (it's basically like how Unity detects an "Awake" method). In v1.4, an effort has begun to make most things happen via this new event system. Note that this process is not complete yet and will continue over the following builds.

IMPORTANT: The event system changes the scripting workflow completely. If you will be doing a lot of scripting I strongly recommend reading the event system documentation first and it will all come to you a lot more naturally.

Motions moved to FixedUpdate
Nearly all physics / motion logic of the camera and controller is now executed in FixedUpdate, mainly for framerate & timescale independence, but also to make controller movement and camera & weapon motions more predictable.

Big "spring cleaning"
All the old classes have been restructured for easier overriding of functionality. For example, most methods are now "protected virtual". A method like FixedUpdate will now typically have - instead of just a huge blob of code - a small number of calls to easily overridable and safely mutable methods. This makes it a lot easier to extend, strip down or optimize a class.

XML method headers
Old school method summary delimiters replaced by XML summary tags. In Visual Studio and MonoDevelop you should now be able to mouseover on all method names to get their summary description in a popup. Try it. It's great.

Weapon model instantiated at runtime
The weapon model is now spawned from a prefab instead of located in the scene from the beginning. This is an important difference from before. You need to set up your weapon on a gameobject, make it into a prefab, and assign the prefab to the FPWeapon's weapon prefab slot. The reason for this change was limitations in the Unity workflow with prefabs and animations that didn't become apparent until you got further into working with regular animations.

Core class naming convention change
Classes previously prefixed "vp_FPS" are prefixed "vp_FP" in v1.4. This is to demarcate that they are:

1. intended for all types of First Person games (not just Shooters)
2. and that they are not backwards compatible.

Big directory structure overhaul

There has been slight confusion over what is considered core features and what is just example scripts. With this release I'm starting an effort to make this crystal clear. The first measure will be separating the intended focus feature set from the "feature creep" bonus scripts. This is to provide cleaner folder structure, but also to clarify which scripts are considered **CORE** *(actively developed, optimized and supported)* and which ones are measly **EXTRAS** *(educational bonus and placeholder scripts for prototyping that will require additional work for your final game)*. These extras are typically hastily conceived for simple testing or demonstration. For that reason, experienced programmers may want to disregard the "Extras" folder right away and write more well-structured gameplay scripts in their own style and taste (especially if targeting mobile).

This is not a mobile release

Ultimate FPS Camera v1.4 does not specifically target mobile platforms. Mobile support is a high priority (and the next big focus) but will be an addon asset. The most pressing limitation with this release would be the lack of gameobject pooling. That said, this version has many other new optimizations that should be very beneficial to a mobile build.

New core features

- vp_TimeUtility: Helper methods to set timescale for proper rigidbody physics, fading and clamping timescale, and pausing. This allows for cool slomo (a.k.a. Bullet Time) sequences.
- vp_PlayerEventHandler: This is now the main hub of the system and a good starting point for overview. It declares all actions that can be performed by - or happen to - the player. Note that the class can't be inherited in this version, a known issue that will be amended down the line. See "Known Issues" below for more in-depth comments on the event system.
- vp_FPInput, a new class to store all first person gameplay input in one place. Axis based input should make it easier to target alternative controllers (touch / gamepad). Note that mouse input still resides in the camera class. This is temporary and it will be moved to the input class in a future release. Properties have been prepared in the input class.
- vp_WeaponHandler - This new class offloads weapon switching logic and various other weapon handling tasks from the camera and weapon classes. No more "weapon logic in the camera class".
- vp_Spring: Added smooth forces, the ability to apply force to a spring over several frames. This is crucial for slow motion, and increases the range of possible motions, for example it's a cornerstone in the new weapon footstep system and proved very useful for swinging the mace about smoothly in the prototype melee implementation.
- vp_FPInput: Mouse LockCursor overhaul: New, much better logic to switch the mouse cursor on and off, and the ability to define rectangles on screen where you can click without going into mouselook mode (for GUI buttons).
- vp_State: State blocking feature introduced, allowing states to block all or some of the other states on the same component while active. This is a rather powerful feature for motion logic, removing the need for programming intricate state disabling and re-enabling logics, or to create a full state with all values zeroed just to block an underlying one. Instead, state blocking is supported via the editor and in a standardized way. Just click on the new small "B" button on a state and the rest should be self explanatory.
- vp_FPWeaponReloader: This component handles firearm reload logic, sound, animation and reload duration for an FPWeapon.
- vp_FPController:
    - Has been largely rewritten for better physics. Among other things: proper deflection when the character is sent by forces into obstacles, and configurable force absorption.
    - An advanced slope sliding feature, allowing the player to slide at fixed speed, or

with an accumulating slide speed for out-of-control sliding. Sliding motion is propagated to the camera and weapons which looks really cool.
    - Crouching logic improved. Among other things, the new event system is used to prevent a character from getting up again if blocked by an above object (i.e. hiding under a table, crawling through a ventilation shaft).
    - Far more tweakable jumping, allowing more and better control over jump height and air speed.
    - Fixes to prevent bunny-hopping up steep surfaces.
- vp_FPWeapon:
    - A weapon footstep motion system for increased realism of walking and running motions (inspired by Battlefield 3).
    - New weapon retraction logic that pulls back the weapon on contact with external objects to avoid intersection. This makes it possible to use the system without the weapon camera, allowing e.g. external objects to cast shadows on the weapon.
    - A new pivot rotation spring for the weapon, allowing rotations that were previously not possible.
    - Integrated support for traditional animation on weapons, e.g. fire, reload, wield, along with a system for scheduling ambience / idle animations at random intervals.
    - Angular recoil can now be added around the Z vector, with a dead zone and randomness. In short, this allows for cooler and more aggressive recoil motion (inspired by Far Cry 3).

New extras

- Melee system (prototype): a test class devised to see if melee combat would be feasible using springs and completely without 'real' animations. Procedurally moves any arm & weapon model. No rigging or traditional animation involved / required. *NOTE: This is a brute force prototype script and it's not designed for ease-of-use. Use at your own peril!*
- Pain HUD: a real simple (and perhaps expensive) full screen fade that makes the screen edges fade to red. Intensity of the effect is determined by the amount of damage.
- vp_SimpleAITurret: a quick demo of how to use vp_Shooter independently of the fist person player. Will scan for the local player within a certain range and, upon line-of-sight, start firing its shooter at the player.
- A primitive inventory system for storing items and weapons. Items could be anything and are used for ammo clips in the demo. Note that the demo uses this very crudely. The system actually supports creating different ammo types and having the revolver not accept machinegun ammo etc. Weapons are a special item type that can be used, and depletes other items (ammo). This inventory class is not really intended for use in any complex game, but it illustrates how to write a class that fills up player event handler delegates with activity conditions and attempt callbacks (CanStart_Reload, OnAttempt_DepleteAmmo, OnAttempt_AddAmmo, OnAttempt_AddItem, OnAttempt_RemoveItem, OnAttempt_RemoveClip etc.). Studying this will allow you to write your own more advanced inventory handler that interfaces with the player event handler without the need to change any code in the other UFPS systems.
- Example scenes in v1.4.0:
    - CleanScene - an empty scene for prototyping and playing around with player prefabs.
    - DemoScene1 - The Half Life "orange map" style level used in the webplayer demo.
    - DemoScene2 - A new example scene for physics testing. It has an extra "slidy" player and several ramps tilted at various degrees for different test cases.
    - DemoScene3 - Extended gameplay example scene with scripting examples for creating different kinds of item pickups and powerups. Includes a simple HUD, an example health and death implementation with a "death camera", some explosive cubes and a simple but very angry AI turret.
- 3 new example player prefabs

    - Camera&Controller, which is exactly that - a simple FPS controller with a simple camera.
    - SimplePlayer, which is a very basic player setup with a pistol (similar to what has been previously available for Ultimate FPS Camera). The lack of an inventory component will allow this player to fire an unlimited amount of bullets.
    - AdvancedPlayer, a complex setup that makes use of many states per component, many state blocking rules, most of the new player events, and the new weapon handler, weapon reloader and inventory extra scripts.
- Arm source content now included in the "Content\Art\Weapons\Source\" folder. These are the original arms, rigged for easier adaptation to your own weapon models. See the readme text file for more info.

Bugfixes and Misc changes

- Pausing now works properly. For Ultimate FPS Camera, I recommend using the vp_TimeUtility.Pause property, which properly remembers the current timescale (in case you were in slow motion). Future pausing logic may be added here aswell.
- Unity4: No more annoying warnings regarding gameobject.active having been made obsolete.
- vp_Utility: Several methods to filter floating point value types for non-numerical-errors (NaN) before use with object translation methods. These errors are sneaky bastards, since they don't always communicate a clear error message. Things like gravity or mouselook would sometimes just stop working properly with no clue as to what went wrong. Places where there's a risk of this happening now use the static "NanNSafe" utility methods in vp_Utility.
- vp_FPSShooter split into two scripts: "vp_Shooter" which can be used with any gameobject, and "vp_FPWeaponShooter" that inherits it but is tailored for FPWeapons.
- vp_DamageHandler also split into two scripts: "vp_DamageHandler" which can be used with any gameobject, and "vp_FPDamageHandler" that inherits it but is tailored for the first person player.
- Camera & weapon angle convention overhaul. XZ vectors swapped for some properties. More info to come.
- Camera springs now updated properly in local space.
- All core UFPS classes have been made reasonably timescale and framerate independent.
- vp_Shell: Shells now inherit controller speed, which looks way better. Especially in slow motion!
- Optimization: Many if not most calls to gameobject properties are now cached throughout the system.
- Optimization: Heavy use of dictionaries for lookup throughout, especially where strings are involved, for example in the State Manager.
- Optimization: In case you hadn't noticed, Ultimate FPS Camera uses timers like crazy. The timer class has been heavily optimized and should now have a tiny impact on the garbage collector.
- vp_Timer: The timer handle class has a new property: "CancelOnLoad" which determines whether a timer event should be canceled when a level is loaded. Default is true (preventing some ugly potential nullreferences).
- Audio source pitch adapted to current time scale throughout the project (this is actually something that might be a good candidate for a vp_TimeUtility feature).
- vp_Layer: "Player" renamed "LocalPlayer", and "Props" renamed "MovableObject" (better reflecting how these layers are really used).
- vp_Layer: New layers: "Trigger", "Pickup", "Enemy".
- vp_Layer: "Mask" constants added, for various raycasting use cases.
- vp_Bullet has been renamed to vp_HitscanBullet, a more fitting name.
- vp_Component extended with automatic event handler hookup, caching of gameobject properties and startup gameobject hierarchy.
- Countless other small bugfixes ...

Known issues

- The manual is only partly up-to-date for v1.4. The parts about getting started, about project hierarchy and setup, and about the new event system are new, but other sections reflect the state of the system at v1.33. I chose to prioritize getting the build out quicker. The documentation will be made more complete in the coming days.
- Regular animations haven't been tested extensively. There is a fair chance some of the animation hooks are broken. Please get in touch if you find anything suspicious (note that the package does not contain any regular animations so this would be with your own animated weapon meshes).
- Event system
    - Decoupling and transition to purely event driven system is not complete yet. Core classes have some hardcoded references to each other which should preferably be removed.
    - vp_PlayerEventHandler knows too much about specific gameplay features (reloading etc.) and should be split up further by inheritance. However its reflection logic won't tolerate inheritance (yet). This basically means that you must add new events directly to the vp_FPPlayerEventHandler class, keep a backup and re-integrate on update (for now).
    - Also, I have done some prototyping on an *optional* string-based way of sending messages (like Unity's SendMessage), which would make the system "silent and forgiving" - that is, allowing removal of declared events with zero compilation errors from listening classes. Dictionaries already exist to make this fairly fast, but didn't have time to wrap it up for this release.
    - Event system may generate garbage on registration and unregistration of events. This will be optimized in following builds.
- Earthquake is a player activity for now, until I have decided how to deal with global events.
- For lack of a better place, the inventory has some responsibilites it perhaps shouldn't have. The whole class is a placeholder "extra" though so I didn't prioritize this very high for this release. The main objective has been to clean as much as possible out of the core classes.
- I need sleep ;)

----

v1.33 RELEASE NOTES

>> IMPORTANT: BACKUP ALL EXISTING PRESETS AND SCRIPTS BEFORE INSTALL <<

Installation

If you haven't previously installed v1.3, please read the v1.3 release notes below carefully before installing.

General info

This is a compatibility update to get Ultimate FPS Camera ready for Unity 4.

Special notes

- Unity 4 component hierarchy cannot be altered in "Awake" (this is good to start preparing for, since it'll be a breaking change for many projects). For UFPS it means that initialization code gets pushed to "Start". Subsequently, all code in "Start" that depends on finished initialization needs a new place to live (see changes to "vp_Component" below).
- In Unity 4, "GameObject.active" and "GameObject.SetActiveRecursively" have been made obsolete. In order to stay backwards compatible, Ultimate FPS Camera will keep using these for the time being. This will work fine in Unity 4, but will unfortunately generate

warnings there until I drop support for Unity3.x.

Changelist

- vp_Component: Extended with "Init", a method complementing "Awake" and "Start" that will run once in the first call to "Update". "Init" can be overridden in any class derived from vp_Component to provide this option in a clean manner. Also, added descriptions outlining this class in a bit more detail.
- vp_FPSCamera: Added the "Init" override to allow the default "SetWeapon" call to happen after all "Start" calls but before everything else. Changed default setting from weapon 1 to weapon 0 (none) which often makes more sense. NOTE: vp_FPSPlayer, however, overrides this and in turn sets weapon 1 as its default, which is more beginner friendly.
- vp_FPSShooter: Muzzleflash initialization moved from "Start" to "Awake"
- vp_FPSWeapon: Weapon group, pivot and springs initialization moved from "Start" to "Awake". Removed a bug that would execute "base.Update" twice. Added a few more null checks for stability.
- vp_Component derived classes: All standard Unity methods made overridable. "m_Delta" is now a property named "Delta"
- Presets "WeaponMachinegun" and "WeaponMachinegunReload": Fixed an issue where the machinegun would accept too high input velocities - for example during very long falls - resulting in the arm sometimes flying around detached from your body. To fix this, changed "PositionMaxInputVelocity" from 1000 to 20 (this is a great example of the purpose of that parameter BTW. It's basically there to keep the springs in check).

----

v1.32 RELEASE NOTES

>> IMPORTANT: BACKUP ALL EXISTING PRESETS AND SCRIPTS BEFORE INSTALL <<

Installation

If you haven't previously installed v1.3, please read the v1.3 release notes below carefully before installing.

General info

This is a small bugfix update. Also, some simple code comments have been added for demonstrating how to play conventional animations on the weapon in vp_FPSPlayer.cs.

Bugfixes and misc. changes

- vp_FPSWeapon: fixed an issue with layer assignment on weapons not being recursive, resulting in weapon sub objects with the wrong FOV, clipping etc.
- vp_FPSWeapon: removed some obsolete code which modified the names of weapon models, breaking animation clip metadata
- vp_FPSPlayer: added commented-out code snippets showing how to play animations on the weapon object upon firing
- FPS editor classes: fixed an issue with inspector values being reverted upon pressing play
- CameraDefault.txt: updated a reference to an old parameter that was resulting in a preset error

----

v1.31 RELEASE NOTES

>> IMPORTANT: BACKUP ALL EXISTING PRESETS AND SCRIPTS BEFORE INSTALL <<

Installation
If you haven't previously installed v1.3, please carefully read the v1.3 release notes below before installing.

General info
This is a very small hotfix update to address issues with trigger detection.

Bugfixes and misc. changes
- Broken trigger detection caused by crouching logic fixed
- An issue with the camera collision code jittering upon trigger collision fixed
- OnGUI method added to vp_FPSPlayer with snippets to display a simple health and current ammo HUD

----

v1.3 RELEASE NOTES

>> UPDATING? BACKUP AND DELETE YOUR OLD FOLDER FIRST! <<

Installation

Do NOT install this version over the previous version of the Ultimate FPS Camera folder! This is a very extensive overhaul of the system. It's more than likely to break your project. If you're close to a deadline you will want to wait. It might be a good idea to first install v1.3 into a clean project and have a look before integrating it into your project.

Recommended update steps:
1. Backup your existing project folder
2. Delete the "Ultimate FPS Camera" folder from your project
3. Go to the Asset Store and upgrade Ultimate FPS Camera to version 1.3
4. Read up on the new important concepts below
5. Integrate the new system with your existing game scripts

General info
Ultimate FPS Camera v.1.3 is a major update with many new features, optimizations, bugfixes and system overhauls. This release aims to make it easier to get started with Ultimate FPS Camera, to reduce scripting complexity and add many requested features, helper scripts and special effects for getting up and running quickly with a new FPS. The manual has also been expanded with 14 more pages, explaining new and old concepts in great detail.

Special notes

State Manager
This version introduces a new powerful State Manager to handle things like running, crouching and zooming. Every FPS component is now able to blend between multiple simultaneous Presets smoothly, depending on the current combination of States. This greatly reduces the need for scripting state changes.

"Resources" loading replaced by Text Assets
Loading from the "Resources" folder is being phased out in favor of Text Assets. This makes for a better editor workflow, reduces executable / webplayer file sizes and is better suited for asynchronous asset streaming. Furthermore, it removes tons of hardcoded paths from the scripts and is just more of the 'Unity way'. That said, it's still possible to load Presets from the Resources folder even though it's no longer used by the demo scripts

SimpleScript replaced by vp_FPSPlayer
v1.3 has a new well-featured player script that may be used as a starting point for a game. This script makes extensive use of the new State Manager. It is suited for heavy customization to meet your game's various needs. NOTE: this script completely replaces the old 'SimpleScript'. vp_FPSPlayer is now the recommended starting point for studying how this system works (the other demo scripts are highly specialized for the demo walkthrough and are not meant to be used as the starting point for a game, nor examples of best workflow practices).

Directory structure overhaul
The directory structure has been cleaned up and redesigned to separate the core systems from demo content and functionality. NOTE: the root "Presets" folder now contains "clean" presets for use with a new FPS game. The Demo folder has its own "Presets" folder with tons of less useful, demo specific presets.

Image Effects & weapon camera changes
The weapon camera functionality has been redesigned in order to allow for Unity Pro image effects - and to make the system more tweakable + less error prone in general. There is no longer one auto-generated weapon camera per weapon. Instead there is a global weapon camera that exists alongside the weapons in the hierarchy (also when the game is not running). Note that image effects should preferably be added to this new WeaponCamera, not the FPSCamera (see the Image Effects chapter in the manual for detailed information about this).

New base class for FPS components
Base class vp_Component introduced to minimize code duplication. All FPS classes except FPSPlayer now inherit from it (FPSPlayer is a regular monobehaviour to allow for heavy customization).

New features

- New, more "game-ready" FPS player class designed as a starting point for a game or game prototype (replacing "SimpleScript").
- Shooter: New ammo & reload functionality
- New damage handler script to inflict damage on gameobjects and make them respawn with a delay. Bullets now execute a method on the hit object with a floating point damage parameter. Name of the damage method can be set per bullet type/prefab.
- Smooth weapon switching code integrated into the camera system
- Explosion class with area damage, a shockwave affecting camera shakes + player velocity along with example particle effects
- New advanced state system to blend between multiple presets in a controlled manner (reducing the need for scripting in many cases)
- Image effects: Big weapon camera overhaul to allow for Unity Pro image effects. Weapon cameras are no longer one per weapon & auto generated. There is one single weapon camera directly under the FPSCamera. System tested with all Unity Pro Standard Assets image effects.
- New feature to prevent the camera from going through walls. A ray is cast every frame from the camera spring's rest position to its current position. If it crosses a wall, the camera is moved back with a margin. A debug ray can be drawn to visualize the intersection.
- A new set of standard presets for the pistol, revolver and machinegun with crouch, run and reload presets
- Improved crouching logic, shrinking the character controller to allow for climbing through small spaces. NOTE: In order to pull this off without having the player fall through the ground, the center of the controller has been moved to its bottom. This may introduce strange bugs in existing projects but should be easy to fix.
- Presets can now be loaded into memory so you don't have to load them from the resources

folder all the time (loading from resources as a methodology is being phased out in favor of loading from text assets, which is better practice)
- Parameters to limit springs and bob so they don't freak out under extreme velocities
- Feature to save 'tweaks only' to presets, greatly simplifying the process of creating additive presets
- A bunch of new sounds and particle effects
- New demo framework created to better separate the system from the walkthrough demo. Simply disable or delete the "Demo" gameobject in the example levels to remove the demo entirely and get a fresh FPS level.
- Demo walkthrough enhanced with some of the new features
- Earthquake and bomb functionality expanded to allow for much more customizable earthquakes and explosions
- FPSCamera: standard weapon switching functionality introduced. Weapons can be moved smoothly in and out of view with wield and unwield sounds.
- FPSWeapon: Added parameters to limit and scale spring and bob input velocity. This is useful if you change player velocity and spring/bob motion then freaks out - springs can now be limited and scaled back without having to adjust all values.

Bugfixes and misc. changes
- SimpleScript removed and replaced by vp_FPSPlayer
- DemoScript removed and replaced by the scripts FPSDemo and FPSDemo2, using the new demo framework
- The new preset & state system implementation is designed to be more forgiving to beginners (e.g. not resetting component changes upon play)
- Many (if not most) 'private' methods made 'protected' to allow for extending the system without altering the core classes
- Renamed "RefreshSettings" to "Refresh" in all classes. shorter and sweeter.
- vp_FPSCamera: Weapon sounds no longer cut off when switching weapons after recently firing a shot
- vp_Bullet: Bullets are now only added to the decal manager if they have a renderer, for mobile games that may not use decals.
- vp_ComponentPersister: Heavy modification of the component persister to make it work with the new state system and make it stable when loading another level.
- vp_ComponentPreset: Major rewrite of the preset system to work with textassets and the new state system.
- vp_FileDialog: Improvements to the file dialog class to make it work better with various warning message box states.
- vp_FPSCamera: Removed a questionable hard coded camera position offset. This change will probably mess with existing presets and projects but it's better long term.
- vp_FPSCamera: Angle property added to clamp pitch values better (replaces setangle)
- vp_FPSController: Support for making the charactercontroller (i.e. collision) smaller when crouching.
- vp_FPSController: Fixed a floating point issue with MotorThrottle. It didn't come to rest at 0.0f properly which messed with the anti-bump feature.
- vp_FPSController: Velocity property added.
- vp_FPSController: Added a check to prevent moveDirection from becoming NaN which would cause a crash.
- vp_FPSController: Jump force range increased in the controller editor
- vp_FPSShooter: Fixed a bug with next allowed fire time being set to a large value by default which sometimes prevented the weapon from being fired for a very long time
- vp_Layer: Added "Props" layer at position 28. Used in this case by the new DamageHandler for respawn logic
- vp_Timer: Changed the name of the 'At' method to 'In' which makes for more elegant readability, for example: to do something in 5 seconds: vp_Timer.In(5, DoSomething);

- vp_Timer: Small fix for more stable creation of timers

----

v1.21 RELEASE NOTES

>> IMPORTANT: BACKUP ALL EXISTING PRESETS AND SCRIPTS BEFORE INSTALL <<

General info
This is a bugfix release.

Bugfixes
- Fixed an issue with FPSShooter giving NullReferenceExceptions resulting in the FPSPlayer prefab working sporadically or not working at all in some circumstances.
- Fixed an issue where multiple weapons would be rendered inside each other.
- Fixed an issue where removing a weapon from the FPSPlayer would break the functionality of the other weapons.
- Improved zoom functionality in SimpleScript.cs.
- Fixed an issue in DemoScript.cs where the weapon layer would not be reset after leaving the 'Weapon Layer' demo screen with layer set to 'OFF'.

Special Notes
A crash can be caused by using the FPSPlayer prefab from 1.11 or earlier. This can happen since the FPSPlayer prefab used to be located in the root of the package, but the new FPSPlayer is in the "Prefabs" folder. In other words, the old one is not overwritten by installing v1.2 or above. If you have an old FPSPlayer prefab in the asset root, please delete it and only use the one in the "Prefabs" folder.

----

v1.2 RELEASE NOTES

>> IMPORTANT: BACKUP ALL EXISTING PRESETS AND SCRIPTS BEFORE INSTALL <<

General info
With version 1.2, Ultimate FPS Camera has been expanded with a battery of new shooter features including advanced recoil settings, raycasting bullets, a decal manager for bulletholes, realistic shell case physics and muzzle flashes. Several example particle FX prefabs are included. The manual has been extended to discuss all new features in extensive detail.

Special Notes
Most features from v1.11 should be intact with one exception: The "Gameplay" foldout of the Shooter editor has been removed and the "FiringRate" parameter contained in it has been moved to the new "Projectile" foldout. Any instances of the parameter "GameplayFiringRate" in your existing preset scripts must be renamed to "ProjectileFiringRate". Other than that, it's all new goodies:

New Features
- Shooter: Big update. Added support for spawning projectiles with a muzzle flash and ejection of shell casings. Multiple projectiles can be fired simultaneously, with configurable conical spread (for scripting shotgun type weaponry or manipulating accuracy at runtime).
- Shell: A shell casing script with its rigidbody physics adapted for more realistic behavior. Has a simple optimization feature to control the amount of shells left on the ground. The shells are ejected with configurable velocity + random spin and bounce with random bounce sounds. Shell ejection can be delayed for e.g. shotguns or grenade launchers.

- Muzzle flash class: Renders an additive, randomly rotated, fading out muzzleflash.
- Decal manager: A class to handle fading out and removal of decals. The max amount of decals in the scene is fixed. A decal stays on the wall forever if no more than a certain amount of decals are added. After that, a number of the oldest decals will start fading out a tiny bit for every shot fired, until eventually being removed.
- Bullet class: A generic class for hitscan projectiles. Raycasts to the target and spawns a decal, a bunch of particle fx and a random impact sound.
- Layer manager: Provides an overview of allocated layer id's in one single place. Makes it easier to debug layer issues.
- Sounds: 4 bullet impact variants and 4 shell bounce variants for random variation.
- Bullet impact particle FX for firearms: Debris, Dust, Impact and Spark prefabs.
- New example models and textures: A shell casing, 2 muzzleflashes and a bullet hole.
- DemoScript, SimpleScript and FPSPlayer prefab adapted to the new Shooter systems.
- Shooters enabled on the Mech, Sniper Breath and Turret examples.
- Zoom examples now behave more like a modern fps (although this is just a simple example). Weapon goes to the middle of the screen and movement speed and accuracy is affected.

Bugfixes
- DemoScript: Firing while crouching is now handled better.
- FPSCamera: Camera cullingmask for weapon layer handled better for compatibility with other camera layer utilizing systems and shaders (e.g. scene reflective water or camera rendered gui systems).

----

v1.11 RELEASE NOTES

>> IMPORTANT: BACKUP ALL YOUR EXISTING PRESETS AND SCRIPTS BEFORE INSTALLING THIS VERSION. <<

General info
Version 1.11 is primarily a bugfix release with a couple of additional minor features. One substantial change is that most scripts have been renamed in order to avoid problems with other assets and Unity core classes. This will probably mess with your scenes. To get back up to speed quickly, follow these steps:

1. It is recommended to backup and delete the Ultimate FPS Camera folder structure before updating / reimporting in order to avoid problems with old file versions.
2. Re-connect all FPSController, FPSWeapon and FPSShooter scripts in the Inspector, where it says "Missing (Mono Script)".
3. If your old presets cause an error about the wrong component type, inside each preset text file add the prefix `"vp_"` to the component type name (reflecting the new class filenames).

This is hopefully the last version for a while to require this level of re-installation.

Bugfixes
- Better class naming convention, in order to avoid collision with other assets and Unity core classes. All VisionPunk system classes are now prefixed with `"vp_"`.
- Initial camera & weapon rotation: Fixed an issue where weapon model would be offset if the player had been rotated in the editor. Controller initial rotation is now handled much more robustly.
- Anti-bump offset: Pushes the controller into the ground to prevent the character from "bumpety-bumping" when walking down slopes or stairs. This makes a huge difference in

terrain.

New Features

- Slope speed multiplier: Allows slowing down the CharacterController moving uphill, and giving it a speed boost running downhill.
- Fall sway factor if grounded: You can now make weapon vertical sway less pronounced when moving in slopes than when falling.
- Random fire sound pitch: Optionally pitches the sound of each shot slightly different to get a more organic firing sound.

----

v1.1 RELEASE NOTES

>> IMPORTANT: BACKUP ALL YOUR EXISTING PRESETS AND SCRIPTS BEFORE INSTALLING THIS VERSION. <<

General info

Version 1.1 is an interim release with a significant overhaul of the component structure. The "Weapon" part of the camera class has been moved to its own class, FPSWeapon. This will pave the way for more advanced and flexible weapon features in upcoming releases. It also results in a better object oriented structure. If you're an existing user, this means you will have to manually update your camera presets from version 1.0. See below for a walkthrough on how to do this.

New Features

- New component: FPSShooter - with an advanced recoil functionality. Lots of parameters to tweak for making your weapon kick like a mule or twist in any direction when fired.
- Timer class, a small but very powerful and easy to use timer system for scheduling all sorts of client side game events.
- 6 high quality sound effects: Earthquake, Explosion, Machinegun, Pistol, Revolver and Stomp.
- Better transitions between weapons in the demo.

Updating your existing camera presets

Old camera presets need to be split into two; one for the camera and one for the weapon settings. NOTE: This only affects camera preset scripts that you may have created yourself. All demo example presets from v1.0 have been updated in v1.1.

1. Open an existing camera preset saved from v1.0.
2. Cut and paste all lines that begin with the word "Weapon" into a new text file (and preferably save the file as a name with "Weapon" in it).
3. In the original camera preset file, remove the prefix "Camera" from all lines beginning with "Camera" (Note: be careful not to remove the name of the ComponentType, "FPSCamera"). Save the camera script.
4. In the newly created weapon preset file, remove the prefix "Weapon" from all lines beginning with "Weapon".
5. At the top of the weapon preset file, add the following line and save the script:
   `ComponentType FPSWeapon`
6. See the manual for information on how to set up FPSWeapon components on your weapons.
7. Remember to load your newly saved presets onto your components using Preset->Load in the Inspector.

----

v1.0 RELEASE NOTES

General info
Easily enhance your game with the ultra smooth motion seen in today's top first person shooters. Ultimate FPS Camera animates your camera and weapons in response to player input, resulting in super lifelike behaviour that animations alone simply cannot match! Ultimate FPS Camera feeds player movements (mouse input, walking, jumping) into the camera and weapon transforms using realtime spring physics, sinus bob and procedural noise shaking. The system uses over 50 parameters to manipulate the camera and weapon model, allowing for a vast range of complex, realtime-generated behaviors.

Features
- Spring physics: Breathe life into everything from melee weapons to guns and cockpits
- Configurable Mouse Smoothing and Mouse Acceleration: say goodbye to jerky input!
- Procedural Weapon and Camera Shakes
- External forces like Earthquakes, Shockwaves, Stomping, Impacts
- A powerful Preset System
- Persist run-time component changes
- 3-axis versatile Bob
- 4 weapon models: Pistol, Machinegun, Mace and Revolver
- Full, well-commented C# Source Code