# THE CAMERAS OF UNCHARTED DRAKE'S DECEPTION

prepared by

Jason Gregory, Lead Programmer and Travis McIntosh, Lead Programmer Naughty Dog, Inc.



# Agenda

- Overview
- The Render Camera
- Game Cameras
- Blending and the Camera Stack
- The Camera Manager
- Follow Camera

- Camera Collision
- Aim Camera
- Melee Camera
- Camera Additives
- 3D
- Future Directions

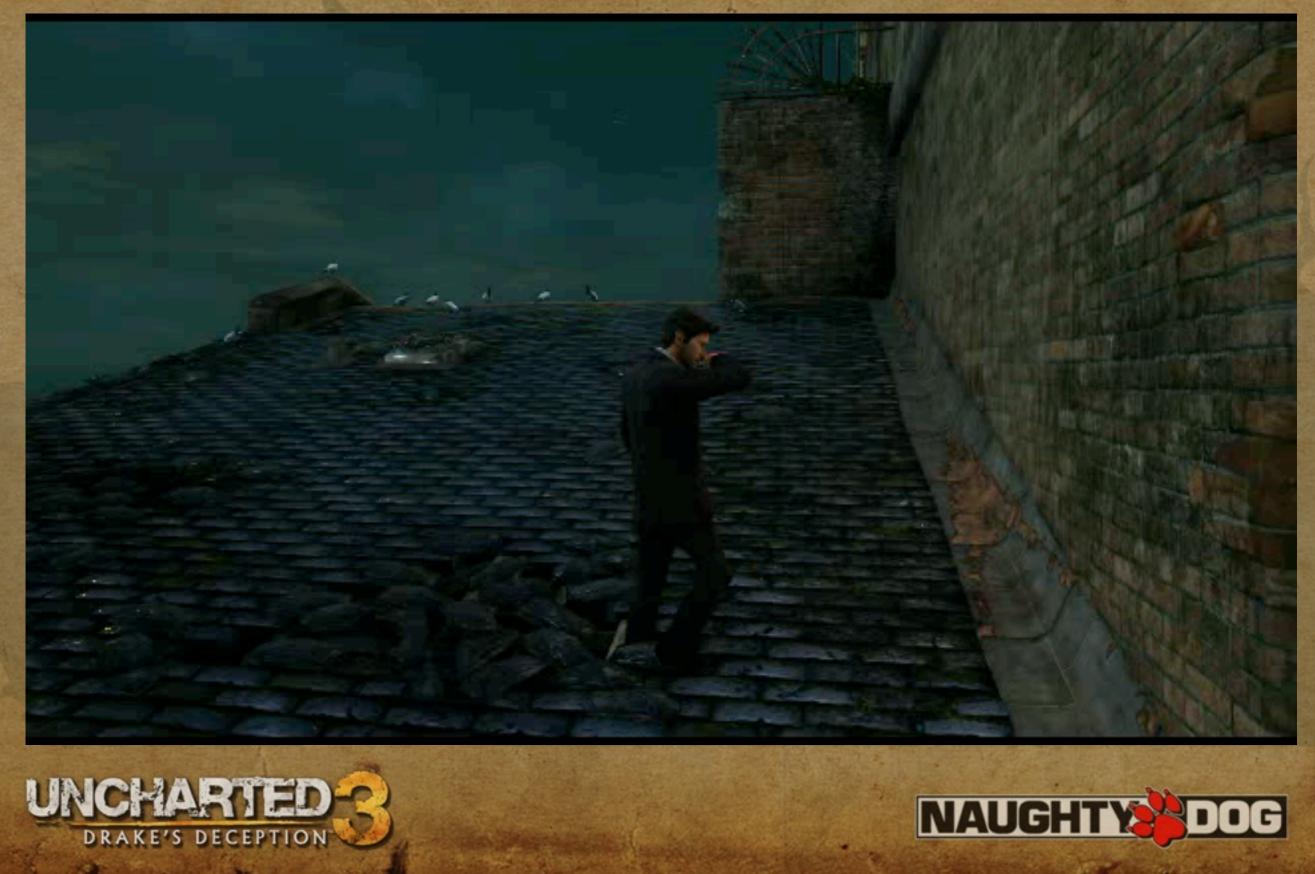












#### Overview

- **Duties** of typical game camera system:
  - Approximate model of real-world camera
  - Control position, orientation, zoom, depth-of-field, other parameters **over time**
  - Support various camera types (behaviors)
  - Manage cross-blends between cameras
  - Detect and resolve collisions





#### The Render Camera

- Low-level interface to rendering engine
  - Parameters include:
    - position, orientation, aspect ratio, field of view (FOV), frustum clip planes, projection properties (ortho, perspective), 3D parameters (e.g. interocular distance)
- One render camera for full-screen
  - One render camera for 3D (rendered twice, offset left and right for each eye)
  - Two render cameras for split-screen





#### Game Cameras

- Render camera API inconvenient for game programmers and designers
- Game camera abstraction more useful
  - C++ class = distinct camera behavior
  - C++ instance = logical camera in world
  - Blending between instances = cross-fade between camera locations, settings and/or behaviors





#### Game Cameras

- **31 camera types** in Uncharted 3
  - Evolved extensively over the series
  - We'll explore a handful of these today
- Three categories:
  - Code-driven cameras
  - Designer-controlled cameras
  - Debug cameras





### **Code-Driven Cameras**

- Follow camera
- Edge camera
- Aim camera
- Cover camera
- Melee camera
- *etc*.





## **Designer-Controlled Cameras**

- Examples:
  - Fixed camera
  - Pivot camera
  - Spline camera
  - Animated camera
  - *etc.*
- Cameras placed in level editor (Charter) and activated via script
- Designers can also override settings of code-driven cameras





## **Debug Cameras**

- Free-flying camera
- Simple "no clip" follow camera
- Stick camera (simple follow cam)





#### Game Cameras





#### Game Cameras

Persistent Camera Requests For Screen #0 Name Type Priority Num contributing states: 0 (index 255) Active Camera Stack For Screen #0 --- kCameraManualBase [ 114] (1.00, 0.00, fade --- kCameraFollow [3296] (1.00, 1.00, fade



) (16.7, 14.7, 13.5 : -0.0, 0.2, 0.0, 1.0 : 75.0) 0.3 2. ) (16.7, 14.7, 13.5 : -0.0, 0.2, 0.0, 1.0 : 75.0) 0.3 2.

Wednesday, November 30, 2011

UNCHARTED

DRAKE'S DECEPTION

### **Blending and the Camera Stack**

- Cross-fading between cameras implemented using a simple stack
  - Each new camera is pushed onto the stack
  - Fades up from 0.0 to 1.0 (100%) over time
  - Blended with camera(s) below it on stack
  - Implicit contribution of non-top cameras is  $(1 \beta)$ , where  $\beta$  is contribution of top camera
  - Once a camera is no longer contributing, it is automatically removed from the stack



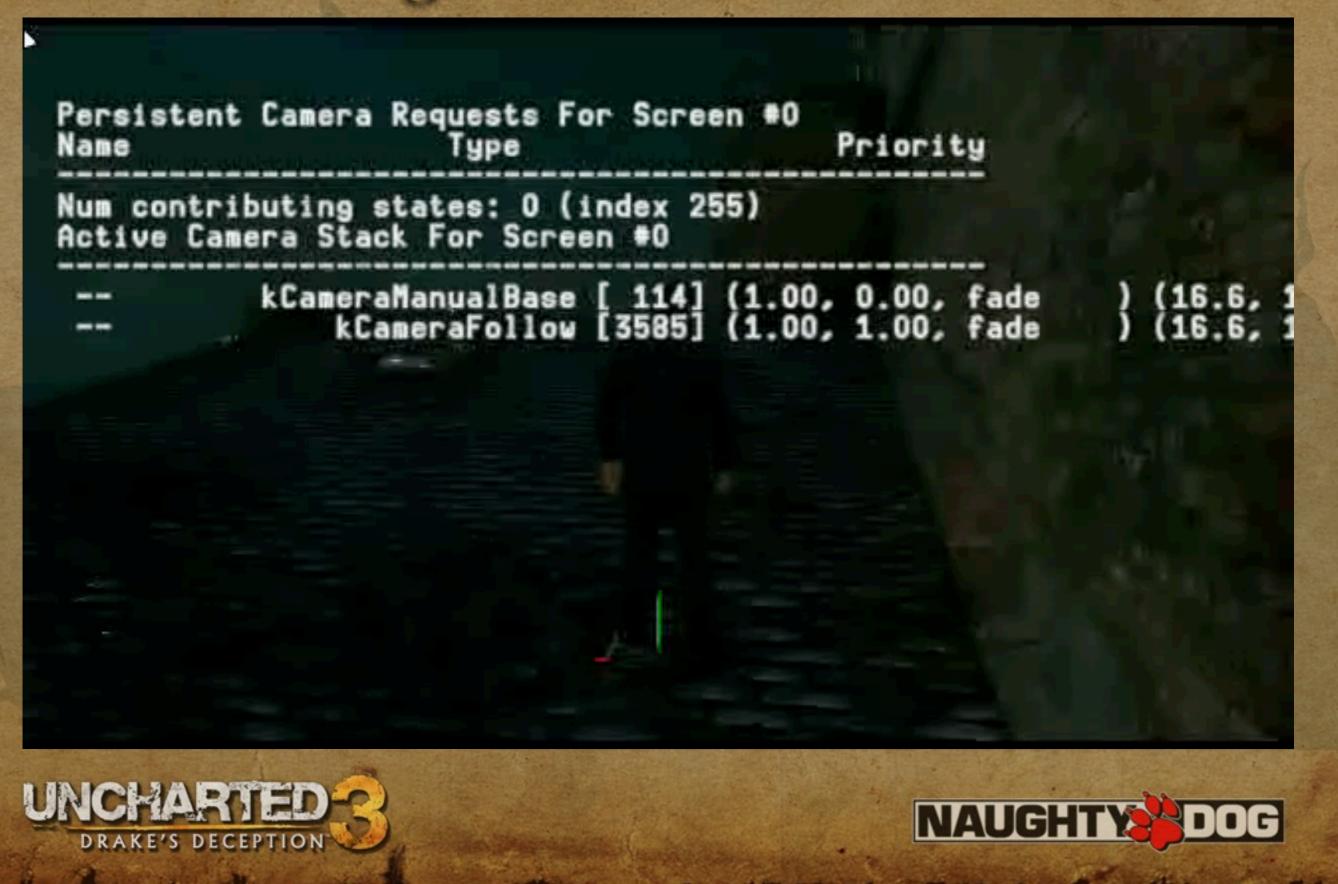


# **Blending and the Camera Stack**





## **Blending and the Camera Stack**



• Simultaneous camera requests are resolved by the camera manager • Simple priority-based system: 1. Most code-driven cameras (follow, edge, cover) have lowest priority 2. Designer-specified cameras are next 3. Special code-driven cameras (aim, death) have highest priority





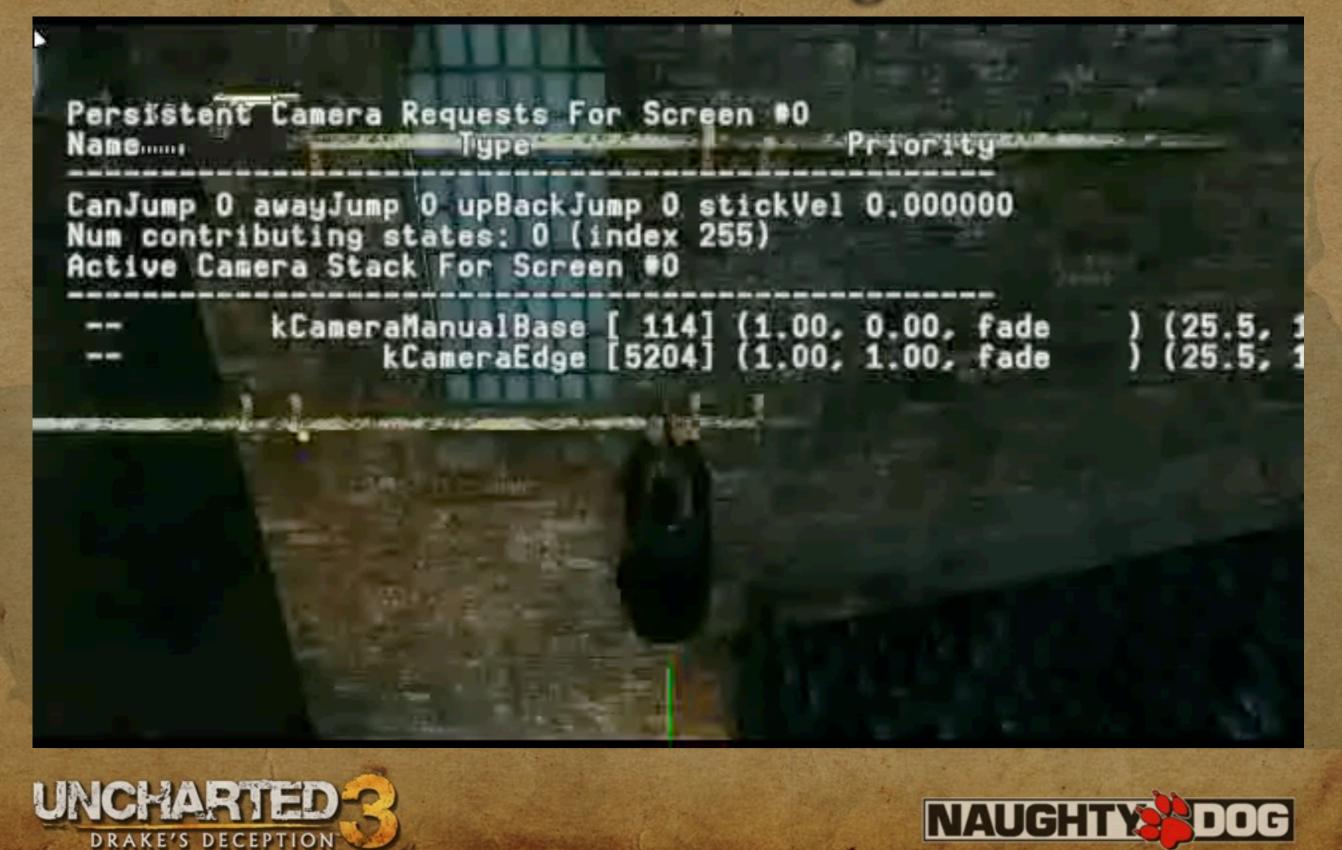
- Code-driven cameras requested every frame
- Designer cameras requested only periodically (via script)
- Therefore designer camera requests are maintained on a **persistent list**
- Every frame, camera manager:
  - finds highest-priority designer request (if any)
  - compares to highest-priority code-driven request
  - highest priority of these two "wins"
- Designers can **disable** or **abandon** their camera requests in script











### **Fixed Camera**





#### **Fixed Camera**

Persistent Camera Requests For Screen #0 Priority Name Type Camera-entity-2677 CameraControlFixed Num contributing states: 0 (index 255) Active Camera Stack For Screen 40 designer:15 kCameraManualBase [ 114] (1.00; 0.00; Fade kCameraFollow [11242] (1.00 0.02; Fade kCameraFixed [11428] (0.98; 0.98; Fade 0.00 0.00 Use to move the emblems. Cycle between emblems by pressing UD UD or down.





### **Pivot Camera**





### **Pivot Camera**

Persistent Lunana Equests For Screen 10 Priority Name. ss upstairs-fight-controller-1 CameraControlAnimated designer:12 camera-pivot\_dual\_fight-1 (sid #x1bac78e8) designer:12 Player including of content 1 Player successive planness 1 DISABLED MOVE LIST SET TO: meles disabled-list-pub-upstairs Brawl mode ENABLED! UpdateGamera: NULL Npc combo count: 0 Nus contributing states: 0 (index 255) AnimAction phase = 0.419357 Active Camera Stack For Screen +0 kCasereMenuelBase [ 114] (1.00, 0.00, feds ) [5.7, 5.4, -19.4 ; 0 kCaserePtvet [19392] [1.00, 1.00, feds ] [5.7, 5.4, -18.4 ; 0 1-75.01-02





76 01 0

107-1200





Persistent Camera Requests For Screen #0 Priority Name Type spline-can-yen-temp-west-1 CameraControlSpline designer:12 CanJump 0 awayJump 0 upBackJump 0 stickVel 0.000000 Num contributing states: 0 (index 255) Active Camera Stack For Screen #0 ) (-168.5, -40.9, -75.9 : 0.2, 0.1, -0.0, 1.0 : 75.0) 0. ) (-168.5, -40.9, -75.9 : 0.2, 0.1, -0.0, 1.0 : 75.0) 0. kCameraManualBase [ 114] (1.00, 0.00, fade kCameraSpline [7398] (1.00, 1.00, fade









CanJump 0 awayJump 0 upBackJump 0 stickVel 0.000000 tt = 0.197906 current index = 2 u = 0.614657 controlPoint tt = 0.285714 prev controlPoint tt = 0.142857

spline-cam-yem-temp-west-1

spline-can-yen-temp-west-1



NAUGHTY

Wednesday, November 30, 2011

3

# Debug Cameras





## **Debug Cameras**







## **Blending Debug Cameras**

- Debug cameras are handled in a special way on the camera stack:
  - The debug fly camera **masks** the cameras below it on the stack, rather than **blending** with them
  - All cameras below the debug fly cam are oblivious to its presence on the stack, and therefore never blend out
- Also, we keep a second debug fly camera at the very bottom of the stack, as a safeguard in case all other cameras get popped



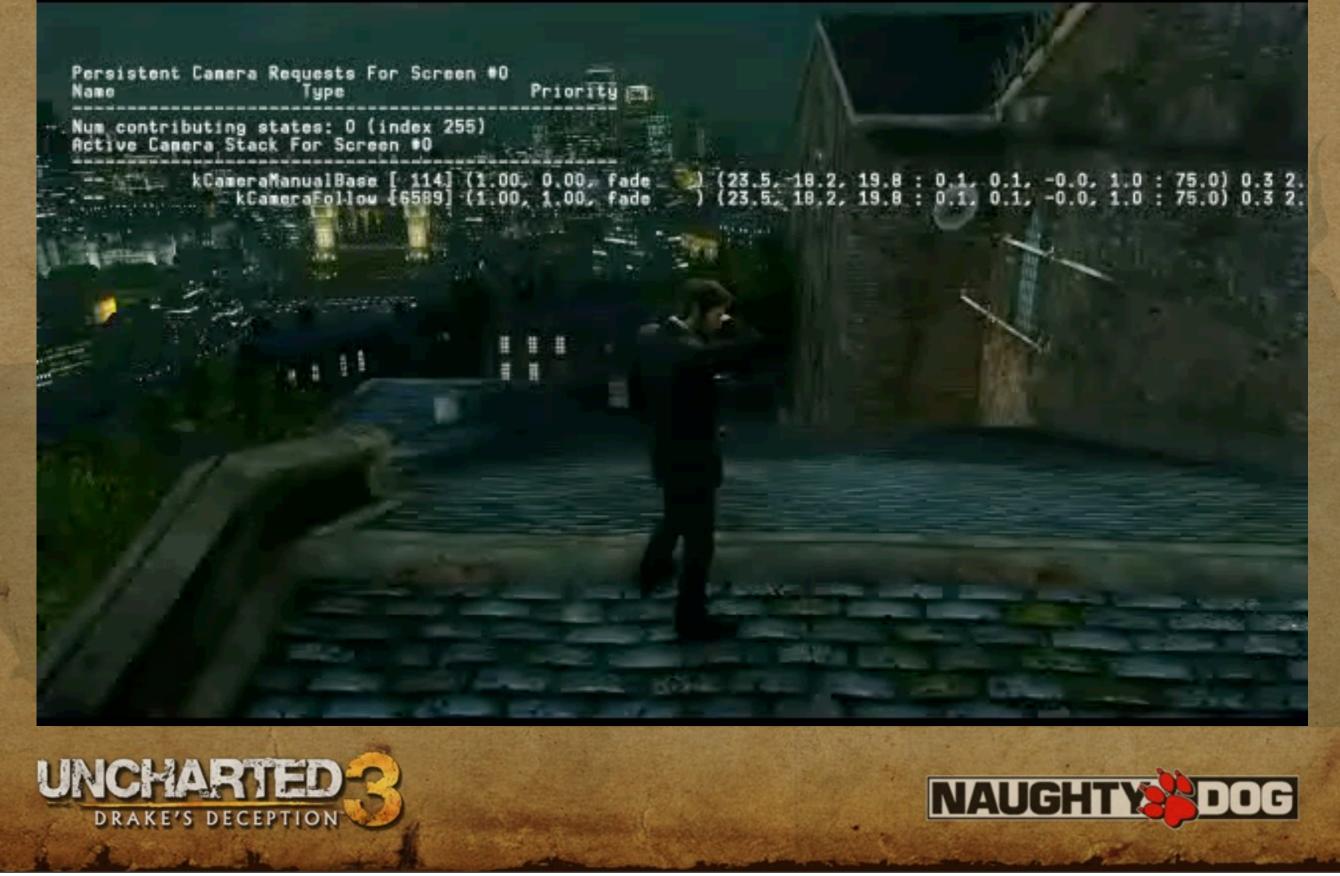


# **Blending Debug Cameras**





# **Blending Debug Cameras**



#### Follow Camera

- More than 50% of player's time spent in the follow camera
- Most difficult camera type to get right
  - **Target point** is controlled by game (follows the player)
  - **Orientation** is player-controlled (right stick)
  - But camera can also auto-rotate to follow player (in some situations)
  - Excellent collision resolution is crucial (and very tricky!)





#### Follow Camera

- Simple horizontal rotation about player
- Vertical rotation controlled by two splines:
  - target point spline
  - camera position spline
- Splines can be edited in-game





## Follow Camera





## Follow Camera



## **Follow Camera: Collision**

- Two fans of **sphere casts** (horizontal and vertical)
- Re-position and/or re-orient camera based on which direction has most free space





## **Follow Camera: Collision**





## Follow Camera: Collision



## Follow Camera: Modes

- Two rotation modes
- In traversal...
  - auto-rotate to follow player
- In combat...
  - **never rotate the camera**, because this would interfere with player's targeting





## Follow Camera: Traversal





## Follow Camera: Traversal

Receistent Camera Requests For S Name Type

Num contributing states: 0 (index 255) Active Camera Stack For Screen #0

> kCameraFollow [2469] (1.00, 0.00, fade } (-26.1, -3.7, 117.5 : -0.0, 0.9, -0.0, -0.4 : 75.0) 0. kCameraFollow [2469] (1.00, 1.00, fade } (-26.1, -3.7, 117.5 : -0.0, 0.9, -0.0, -0.4 : 75.0) 0.





## **Follow Camera: Combat**

Persistent Camera Requests For Screen #0

Num-contributing states: 0 [index 255] Active:Camera Stack For Screen #0

> kCarecoManualBase [ 114] (2 00, 0 00) Fade kCarecoManualBase [ 114] (2 00, 0 00) Fade

} {-84.7, 0.1, 8.4 : 0.0, 0.4, -0.0, 0.9 : 75.0} 0.3 2.2 } {-84.7, 0.1, 8.4 : 0.0, 0.4, -0.0, 0.9 : 75.0} 0.3 2.2





#### Follow Camera: Smoothing

- Ideal post-collision camera position jitters quite a lot
  - we **smooth** this with a simple spring
- Spring-mass system in a viscous fluid:
  - $m (d^2x/dt^2) + c (dx/dt) + k^2x = 0$
  - Critically damped to prevent oscillation
  - Solution:  $(A + Bt)e^{-kt}$
- Spring generally much tighter in combat than in traversal mode





## **Designer-Overridden Settings**

- Designers can control follow camera settings via script, to tailor behavior for specific areas in the game world
- Can also control preferred angle of follow camera
  - Useful at conclusion of in-game cinematic,
  - or to direct player's attention toward a point of interest





# **Designer-Overridden Settings**





## **Designer-Overridden Settings**

Priority



Scent Camera Requests for Screen #0

CameraHanualBase [ 114] (1.00, 0.00, fade kCameraFollow [ 263] (1.00, 1.00, fade

Num contributing states: 0 (index 255) Active Campra Stack For Screen 90

NAUGHTY

95.9 0.0.0.8. -0.0. 95.9 0.0.0.8. -0.0.

#### Aim Camera

#### • Requirements:

- Resolve collisions without changing aim angle
- Maintain **center of screen** when blending back and forth with other camera types











#### Aim Camera







## Aim Camera: Collision

- Collision done via a few simple sphere casts
- Camera position interpolates between:
  - ideal "far" camera position, and
  - a secondary "close" camera position



















- When procedural cameras won't cut it, animators can control the camera directly
- In Maya, special locators called action pack references (apReferences) are used to export custom animations
- An apRef can be constrained to a Maya camera, allowing the camera's movements to be exported to the game









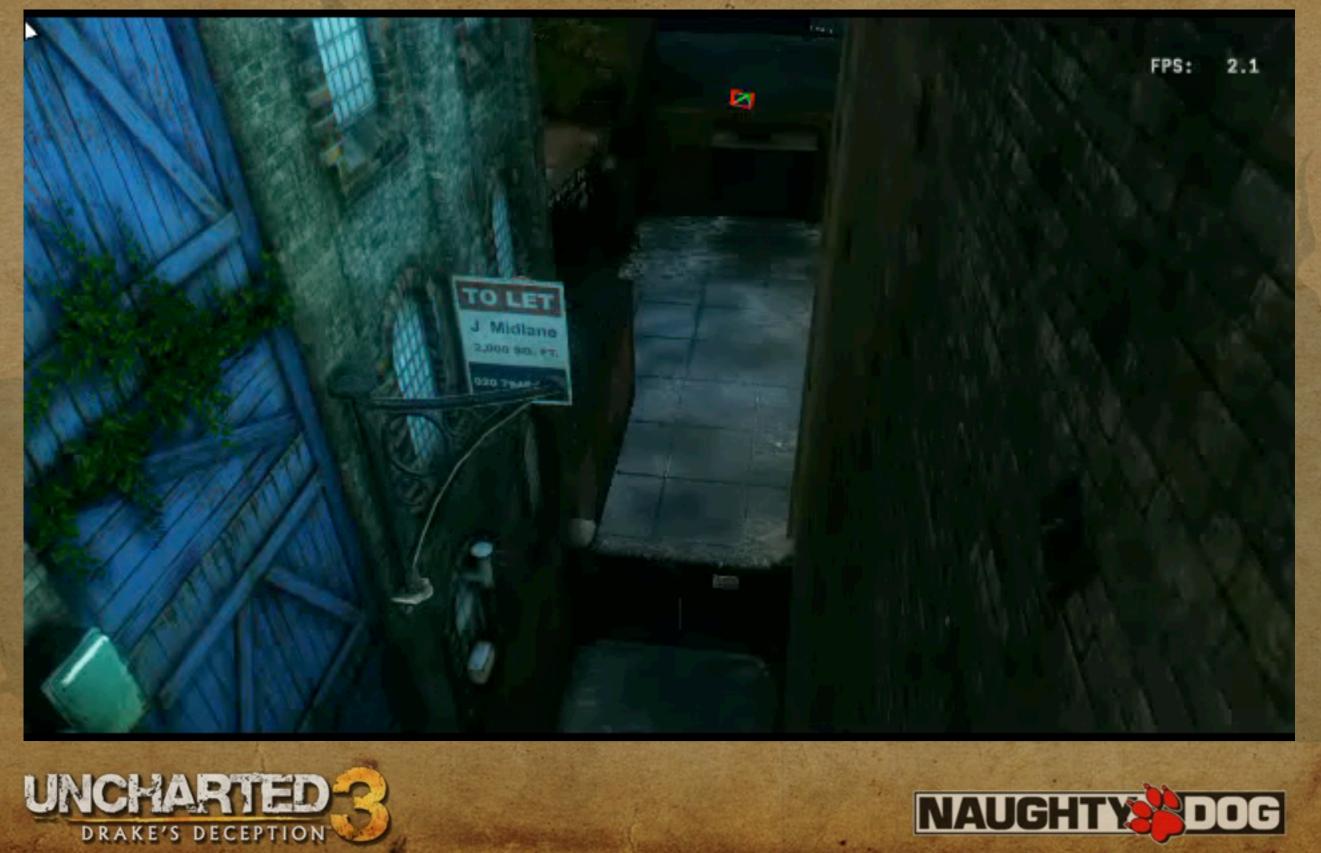


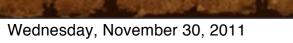














#### **ApRef as Common Reference**

 An apRef can also be used to provide a common point of reference between the various actors in a scene



Actor1

apRef1



Actor2

#### **ApRef as Common Reference**

 An apRef can also be used to provide a common point of reference between the various actors in a scene







- Animated cameras are used in melee
- Multiple camera paths are authored for each melee move
- At run time, ray casts filter out any animated camera paths that would collide with game world:
  - select an animated path if possible,
  - else fall back to procedural melee cam









Priorate

Persistent Camera Requests For Screen #0 Name

Player melee combo counter: 1 Player successive punches: 0 DISABLED MOVE LIST SET TO: melee-disabled-list-pub-downstairs Brawl mode ENABLED! Npc combo count: 0 Num contributing states: 0 (index 255) AnimAction phase = 0.022792 Active Camera Stack For Screen #0

> kCameraManualBase [ 114] (1003; 0703; 40de ) kCameraMelse [22007] [1 00, 1.00, fade

3 7. 2.8. 8.3 : 0.1. 0.8, -0.1. 0.8 : 75.0) 0.3 2.2 ) (3.7. 2.8. -9.8 : 0.1. 0.6. -0.1. 0.8 : 75.0) 0.3 2.2

wall Ap wall enter





Persistent Camera Requests For Screen #0 Name Type Priority

Player melce combo counter: 1 Player successive punches: 1 DISABLED MOVE LIST SET TO: melce-disabled-list-pub-downstairs Brawl mode ENABLED! UpdateCamera: NULL Num contributing states: 0 (index 255) AnimAction phase = 0.000000 Active Camera Stack For Screen #0

kCameraManualBase [ 114] (1.00, 0.00, fade kCameraMelee [22360] (1.00, 0.07, fade kCameraMelee [22498] (0.93, 0.93, fade 1, 2,9, -7.7 : -0.1, 0.9, -0.2, -0.5 : 75.0) 0.3 2.

) (6.1, 2.9, -7.7 : -0.1, 0.9, -0.2, -0.5 : 75.0) 0.3 2. ) (5.8, 2.9, -7.5 : -0.1, 0.9, -0.2, -0.4 : 75.0) 0.3 2 ) (6.1, 2.9, -7.7 : -0.1, 0.9, -0.2, -0.5 : 75.0) 0.3 2











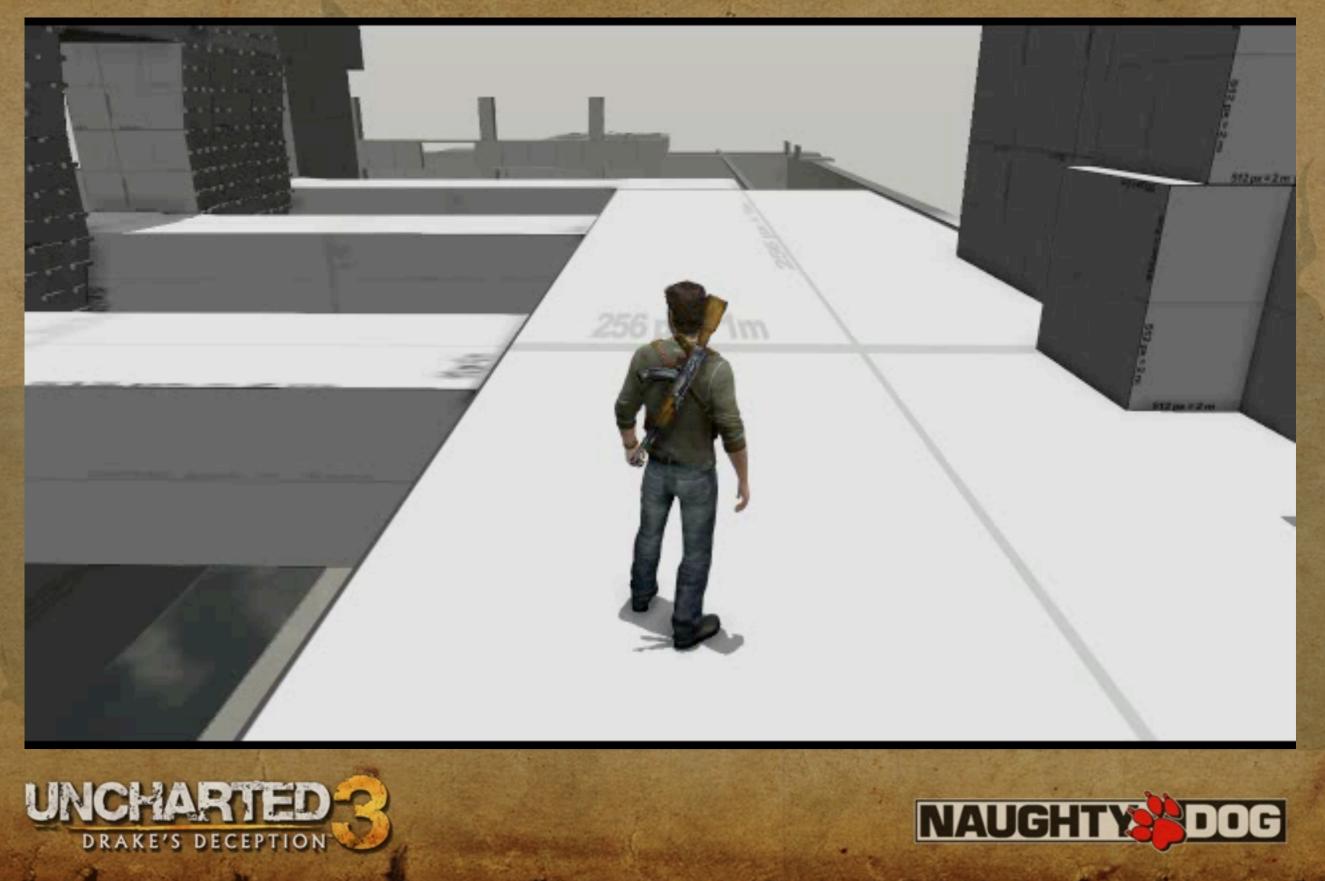
- Animator-authored camera "shake" effects implemented using additive animations
  - Authored as a "regular" camera animation in Maya
  - Applied as translational and rotational deltas at runtime
  - Therefore a given additive shake can be applied virtually anywhere in-game











- Very effective when used carefully
- Useful for small camera movements only
  - Low-frequency and small movements "read" well to most players
  - High-frequency or large movements are too easily misinterpreted as frame-rate hitches or camera bugs

 Applied after camera collision, so large movements can also put camera back into collision!





## **Stereoscopic 3D**

- 3D effect achieved by offsetting the primary render camera left and right, and re-rendering the scene for each eye
- Inter-ocular distance controls relative parallax differential between the eyes
  - Player can configure this in the options menu
  - Kept fixed throughout the game





## **Stereoscopic 3D**

- Zero-plane distance represents the plane in 3D space at which the eyes converge
  - i.e. where objects will appear to coincide with the physical TV screen
  - In U3, we generally never wanted objects to "pop out" of the screen — all 3D effects go "into" the screen
  - Zero-plane distance automatically adjusted, by reading the depth buffer, to guarantee no "pop out"





## **Future Work**

- Camera collision currently done on a per-camera basis
  - This means collision detection/resolution is imperfect during cross-fades
  - Would like to add a final collision "clean up" pass perhaps single sphere cast from camera to target
- Would like better camera pre-vis tools
- Maintaining screen center when blending between cameras is currently done on a per-camera basis
  - Would like a general solver framework, so new cameras could be added easily without having to rewrite the screen center maintenance code for each one



## Workshop

Please join me in the workshop session immediately following this talk for further discussion and Q&A

jason\_gregory@naughtydog.com



