# Advanced Camera Control

IMGD 4000

Original source: Phil Wilkins (Sony Playstation Entertainment). "Designing and Implementing a Dynamic Camera System", *Game Developer's Conference*, San Francisco, CA, USA, 2008.

Note: if you don't notice camera, then it is working well!

"An ideal virtual camera system, regardless of genre, is notable by the *lack* of attention given to it by the viewer"

From Introduction to:
M. Haigh-Hutchinson,
*Real-Time Cameras: A Guide for Game Designers and Developers*, Morgan Kaufmann 2009.

God of War 2 trailer https://www.youtube.com/watch?v=GjYbK_-w9pM

## Camera Objectives

- Flexible and designer driven
  - Allow game designer to provide player experience from variety of perspectives
- Smooth
  - No jarring transitions
- Not require player intervention
  - Player should not have to manually adjust camera to see game
- No collision
  - Designer must constrain so doesn't go through walls

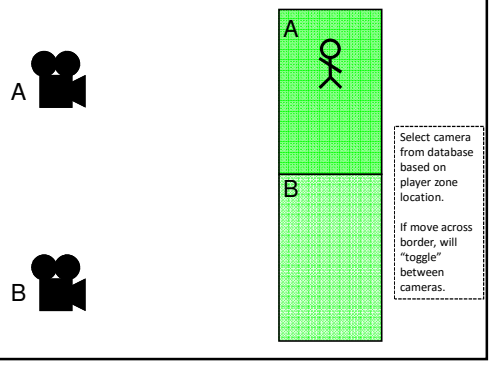
## Overview

- Zoning – deals with use of spatial database to select "right" camera
- Dynamics – calculations for a single, dynamic camera
- Blending – smooth out transitions between cameras
- Rails – constraining camera to path

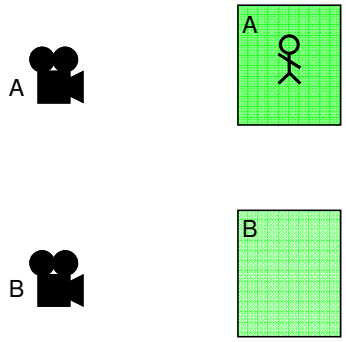
## Zoning : Objectives

- Have multiple stationary cameras
  - Cameras in fixed location
- Chosen by player position
  - Active camera is based on where player is
- Design so that cameras can "cover" where player is
- Switch automatically to right camera

## Zoning : Design



## Zoning : Design



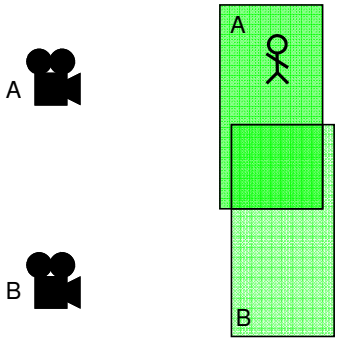

Non-overlapping zones.

Switch to camera when player ENTERS zone.

Provides hysteresis.

But may not "cover" all areas well

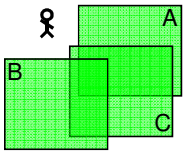
## Zoning : Design




Alternative is overlapping zones .

When enter overlap, switch to new camera

## Zoning : Implementation



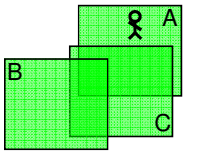

Camera A

Camera B

Camera C

Each frame, query spatial DB and get cameras. Assume unordered (don't want assumption about underlying db)

## Zoning : Implementation




Camera A

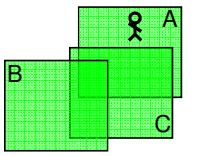
Camera C


If simply switch to new camera, will toggle between A and C every frame.

## Zoning : Implementation

- Submission List
  - List of all cameras that were submitted last frame
  - Used to distinguish newly submitted cameras from old ones
  - New cameras inserted at top
    - So, effectively sorted by age

## Zoning : Implementation





Query Result

A

B

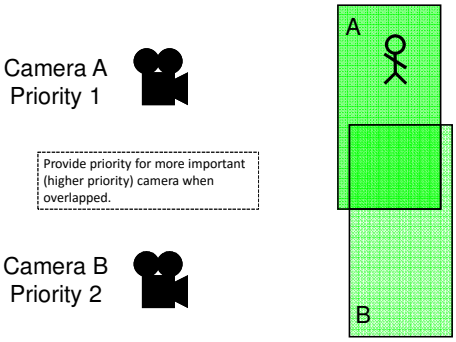
C

Submission List

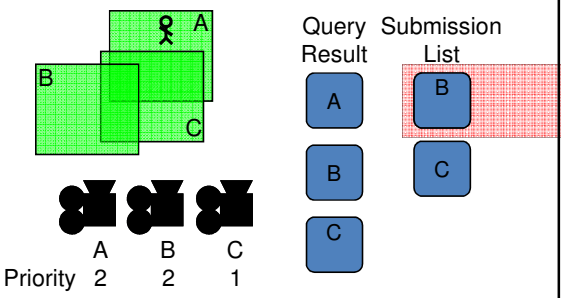
B

- In this example, player moves from A, to C, to B.
- If then to just C, top entry would change.

## Zoning : Implementation




## Zoning : Implementation




## Zoning Implementation

- Submission List (with priorities)
  - Insert and delete entries to match query results
  - Unless query result was empty
  - Sorted by priority
  - Then by age
  - Top entry is active camera

## Outline

- Zoning (done)
- Dynamics (next)
- Blending
- Rails

## Dynamics : Objectives

- Camera impacts 3 properties of avatar as it appears on screen
  - Position – where camera is focused impacts where on screen avatar appears (e.g., center? bottom right?)
  - Size – how far away camera is impacts how big avatar appears (e.g., takes up full screen, takes up tiny portion)
  - Angle – angle of camera from avatar orientation impacts what representation avatar has (e.g., profile? top-down?)

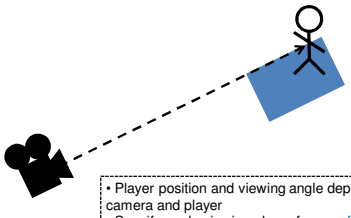
## Dynamics : Design

Define "safe zone" (rectangle) on screen where player avatar will be.
- Make these resolution independent (represent screen, not pixels)

Note, can be "point" if player always there.

When avatar oustide of zone, move camera. How?

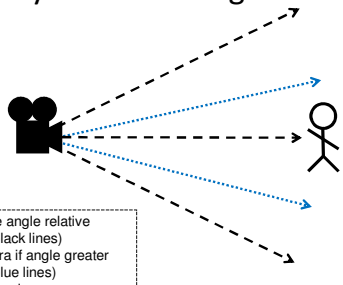
## Dynamics : Design



- Player position and viewing angle depend upon **angle** between camera and player
- Specify angle viewing player from as fixed value

- But result will be camera moves around lots (background moves) → can be disconcerting

## Dynamics : Design

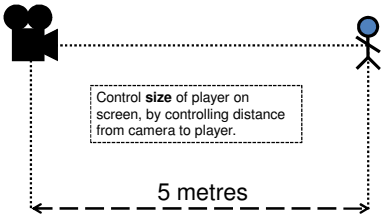


- Instead, calculate angle relative camera location (black lines)
- Only move camera if angle greater than constraints (blue lines)

→ Camera will move less

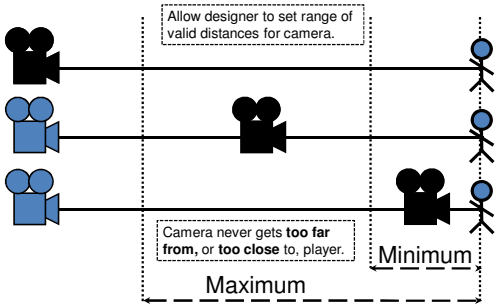
- This is like "high water mark" and "low water mark" in algorithms

## Dynamics : Design



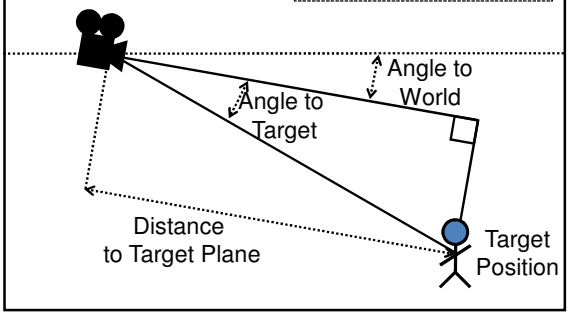

Similar to angle, often don't want as fixed value (see next slide).

## Dynamics : Design




## Dynamics : Implementation

SUMMARY: Let designer control:
position of player on screen
angle looking (orientation of camera)
size (distance from camera)



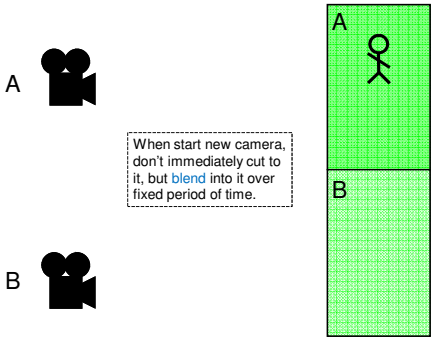

## Outline

- Zoning (done)
- Dynamics (done)
- Blending (next)
- Rails

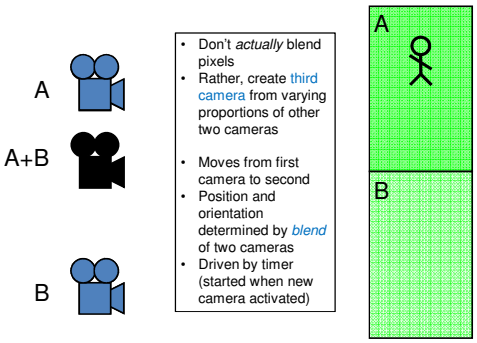
## Blending : Overview

- Blending – smooth out transitions between cameras
- Three aspects:
  - Timers – track and update each blend
  - Ease – controls the smoothness of blend
  - Blend Space – defines what a blend between two cameras does

## Timers : Design



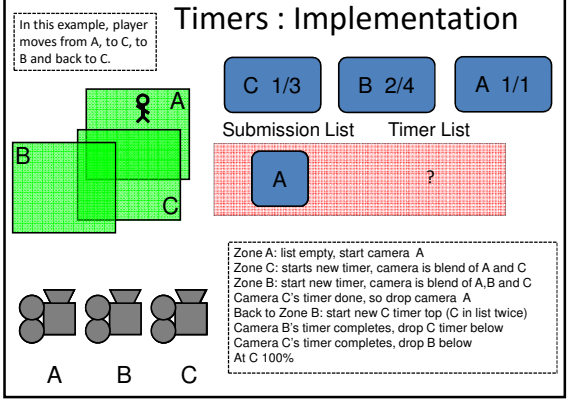

## Timers : Design



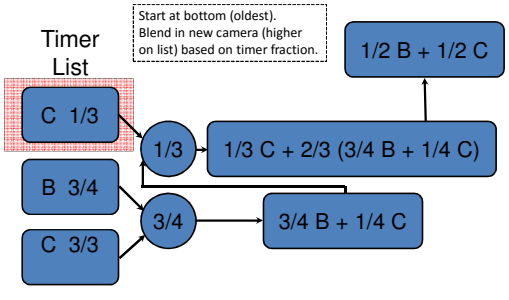

## Timers : Implementation

- Timer List
  - Entry is camera fading in
  - New timers inserted at top
  - Camera can have multiple timers in list
    - This happens if player moves quickly between cameras
  - First-In, First-Out (FIFO)
  - When timer completes, all timers below it are removed

## Timers : Implementation



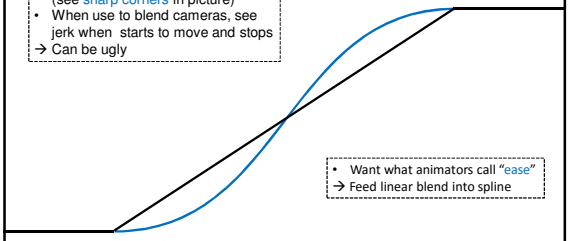

## Timers : Implementation



## Ease : Design

- Using as-is, get simple linear blend (see sharp corners in picture)
- When use to blend cameras, see jerk when starts to move and stops

→ Can be ugly

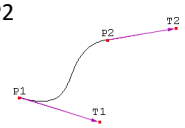


- Want what animators call "ease"

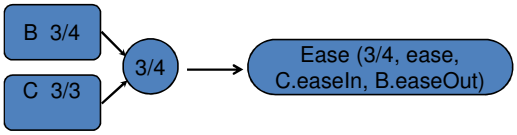
→ Feed linear blend into spline

## Ease : Implementation

- *Hermite Spline* http://cubic.org/docs/hermite.htm
  - Used to smoothly interpolate between key-points (e.g., camera A to camera B)
- Fixed endpoints at P1 & P2
- Controllable tangents
- ease = [0 , 1]
  - 0 means no ease (linear)
  - 1 means full ease
- Ease-in from P1 tangent, and Ease-out from P2 tangent



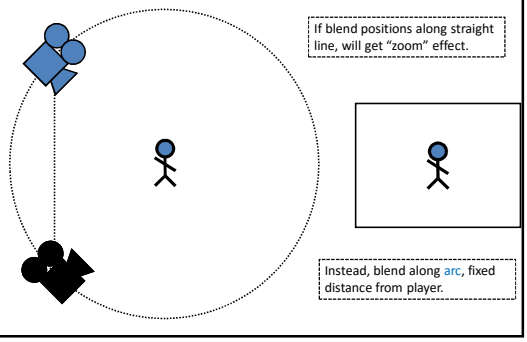

## Ease : Implementation




Apply `Ease()` when calculate blend factor between two cameras

`ease` from 0 to 1
`easeIn` from old camera (C)
`easeOut` from new camera (B)

## Blend Space : Design



If blend positions along straight line, will get "zoom" effect.

Instead, blend along arc, fixed distance from player.

## Outline

- Zoning (done)
- Dynamics (done)
- Blending (done)
- Rails (next)

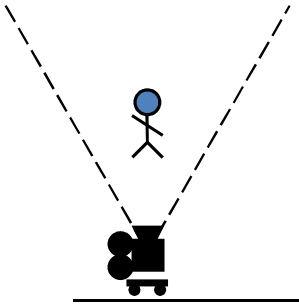
## Rails : Objectives

Want camera on a track → idea borrowed from film industry. Construct rails, put camera on little cart (a "Dolly").

## Rails : Design

- Rail can be curve (e.g., spline – numeric function compose of polynomials)
- Dolly is point on spline

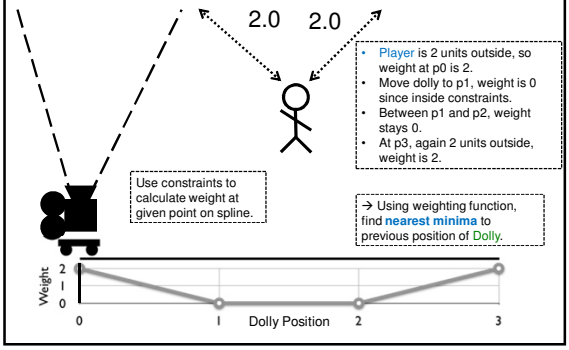
## Rails : Design



Only move Dolly by enough to keep player within constraints defined by camera.

Player is free to move within constraints, but when Player moves outside, Dolly moves to compensate as best it can.

## Rails : Implementation

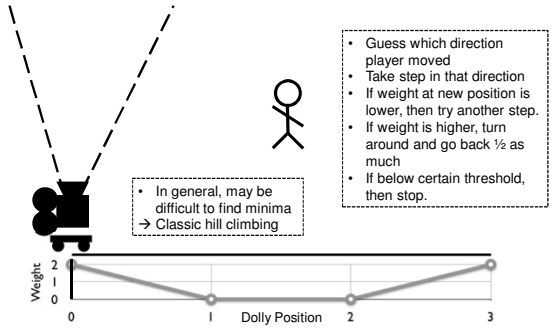



- Player is 2 units outside, so weight at p0 is 2.
- Move dolly to p1, weight is 0 since inside constraints.
- Between p1 and p2, weight stays 0.
- At p3, again 2 units outside, weight is 2.

Use constraints to calculate weight at given point on spline.

→ Using weighting function, find **nearest minima** to previous position of Dolly.

## Rails : Implementation




- Guess which direction player moved
- Take step in that direction
- If weight at new position is lower, then try another step.
- If weight is higher, turn around and go back ½ as much
- If below certain threshold, then stop.

- In general, may be difficult to find minima
→ Classic hill climbing

## Rails : Implementation

- Can experiment with weights
  - Distance from Player to Dolly
    - Classic drag/push camera down corridor
  - Amount Boss obscures Player
  - Number of minor characters out of frame
  - …
- Also, can combine Dolly technique with earlier ones

## Other Stuff (not Discussed)

- Dealing with multiple targets
  - Framing fights, using multiple targets
- Dynamic target definition, and calculation
  - Target changes, fade to different targets
- Overriding cameras at arbitrary points to focus on dynamic areas of interest
  - Different camera "states"
- Physical post effects like shake and sway

God of War 2 trailer https://www.youtube.com/watch?v=GjYbK_-w9pM
What techniques can you identify?