An Overview of the Handy-cam And Leaning (HAL) Head Tracking Technique

August 31, 2011

Please note: the following content has been taken from a draft of my thesis, which is still a "work in progress"

The HAL technique has two key aspects. The first allows a player to invoke an in-game lean movement by either rolling or leaning their heads sideways. An intended use of this game movement is to allow players to stealthily spy on enemies from behind obscuring objects. Likewise, it provides an alternative approach for utilising cover during a gun fight. While such a form of control is common in many FPS titles, most are controlled through the use of keystrokes – typically assigned to 'Q' and 'E'. The proposed advantage of using head movements is that it is arguably more intuitive; you lean-to-lean. Gestural input also allow for partial lean movements, which could provide a tactical advantage over typical on/off leaning. This form of multimodal control also provides a way of distributing the rich set of game control over a more input forms. This allows for more simultaneous control. It also frees up two of the most physically predominant keys for use with other common game commands.

HAL's second aspect is to make game look like its viewed from the perspective of a handheld camera by adding a wobble to the virtual view. Much like films that employ a similar technique[1], the intended purpose of this effect is to make the game experience feel more realistic. Whilst many existing games already incorporate such a wobble through the use of a pre-animated camera shake, HAL's handy-cam effect works by coupling the player's head movements to the camera, thereby reinforcing the conceptual idea that the virtual camera is strapped to the protagonist's head. Whilst there are similar head-techniques that exist for looking around[2], to the authors knowledge this is the first example of where this form of gestural input is used only for aesthetic purposes.

0.1 Logic

The logic for the HAL technique can be understood in part as a transformation from a player's head position and orientation (6DOF) into a *camera offset* vector. This vector is added to the virtual camera to slightly alter its position and orientation, thereby creating the handy-cam effect. This vector avoids altering the camera along the 'forward' axis, as these movements were found to negatively impact on the appearance of the weapon model.

The player's 6DOF is also used to derive a normalised *leaning value*, where 0 represents the neutral, 'no leaning' position. When equal to 1, it indicates that the player wants to perform a full right lean, while -1 indicates a full left lean. This lean value is then used to drive the virtual lean movement, although how this is then accomplished is specific to each game implementation. In the case of using the Source engine, which does not include a leaning movement by default, we chose to map this value to a sideways player offset, a gun movement and a slight reduction in the field of view (FOV). These latter two features were included in such a way as to partially recreate the *iron sighting* effect; a technique previously found to be popular amongst gamers [1].

Figure 1 provides a visual representation of how both these transformations are achieved. From this diagram it can be seen that the *camera offset* and *leaning* values are derived by applying a series of filters to the raw head data. The behaviour of each of these filters is in turn controlled by one or more variables, allowing for the technique to be tuned by both ourselves and the user. For the remaining part of this section, we describe each of the filters along with their use.

[1] Examples include "Blair Witch Project" and "Bourne Ultimatum"

[2] An example of this can be seen in ARMA 2

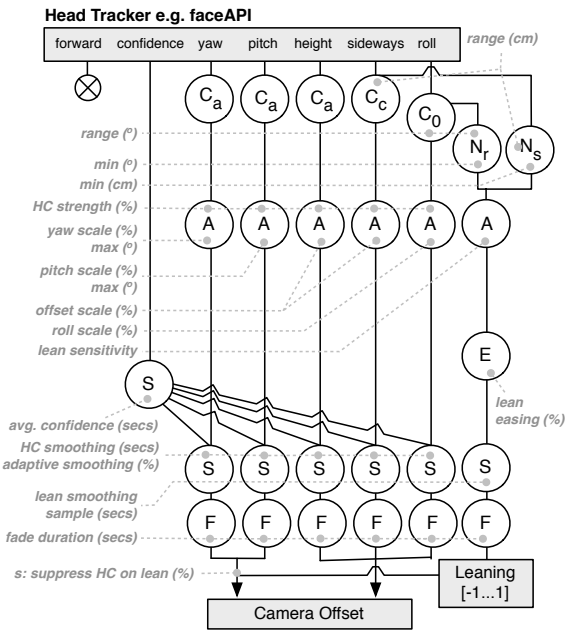

Figure 1: A illustration of how the head data, which includes 6DOF and a tracking confidence value, is converted into a camera offset and a normalised leaning value. Each circle represents one of the following filters: *C* - center, *N* - normalise, *A* - scale, *E* - ease, *S* - smooth and *F* - fade.

0.1.1 Centering

The 6DOF head data provided by the head tracker is stated relative to an arbitrary zero position. To avoid needing to make assumptions about what this zero position is or indeed where the user's head is relative to it, we instead compute an offset relative to the user's "neutral" position. Whereas we computed this offset in the past by applying a dampening value to an accumulated offset [1], we have since found it more effective to simply compute a running average. This approach allows for the value to be determined without manual intervention from the user, whilst also catering for slight position shifts over time. To avoid having the intentional head movements (i.e. the lean movements) alter this neutral position, we weight their influence based on how far they are from the neutral position like so:

$t_s := 0;\ v_s := 0;\ a := 0;$
proc center$(v, \Delta t, a)\ \equiv$
  $w := 1;$
  if $t_s > t_i\ \wedge\ a \in \{\text{ROLL}, \text{SIDEWAYS}\}$
    then
      if $a =$ ROLL then $n := 0;$ fi
      if $a =$ SIDEWAYS then $n := a;$ fi
      $w := 1 - \text{clamp}(|v - n|/r, 0, 1);$ fi
  $v_s := v_s + v \times w;\ t_s := t_s + \Delta t \times w;\ a := v_s/t_s$
  return$(v - a)$.

In this function v represents the latest head value for axis a, while Δt represents the lapse of time since the last head vector. For the *SIDEWAYS* axis, the amount that a new value influences the running average (a) is inversely–weighted by its difference to the running average. Consequently large movements will have less influence, to the extent that movements greater than a given range (r) (2cm for C_h – see Fig. 1) it will not influence the running average. With the *ROLL* axis, the value is similarly inversely–weighted (2° for C_r), but relative to 0. This relies on the observation that

the a 0^o roll value typically represents a straight head, regardless of the setup. These weightings are only used after a short initialisation period (i = 0.5 secs) has occurred. At any time, the neutral position can reset by setting the summed value (v_s) and time ($time_{sum}$) values to 0.

0.1.2 Normalising

To control how much movement is required to invoke a lean, the incoming data is normalised over a tuneable *range* (20° for N_r and 6cm for N_s – see Fig. 1). By doing this it enforces a clear movement maximum, which should help to prevent users from overly exerting themselves – a phenomenon experienced previously when no maximum was enforced [1]. Prior to this normalising, a *minimum* (2° for N_r and 2cm for No_s) is subtracted from the head value, as way of introducing a leaning threshold. The resulting dead zone created around the neutral position helps to ensure that small unwanted leaning movements do not interfere with the player's normal aim.

0.1.3 Global and Individual Scaling

The scaling filter multiplies the incoming data by one or more tuneable values (each initially set to 1 – see Fig. 1). Its impact on the HAL technique is twofold. With regards to the leaning, the scaling value effectively acts as a sensitivity parameter. For instance, when set higher it takes less movement to reach a full lean position. Of note, it is important that this scaling is applied after the normalising to ensure the size of the previously mentioned dead zone remains unaffected by the chosen scale. With regards to the handy-cam aspect, this filter controls how predominate the effect is. For instance, a scaling of 0 would mute the handy-cam effect entirely, while a value of 1 would result in 1:1 head-to-camera movements. In the case of the pitch and yaw axes, the filter takes an additional *max* parameter (both initially set to 0). When set to a value greater than 0, it ensures that the data is scaled such that it never surpasses the specified maximum, whilst still maintaining a relatively linear scaling around 0. To do this the following expression is used:

$$y = 6((2x+3)/6)^2 - 4((2x+3)/6)^3 - 1 \quad [0 \leq x \leq 1.5]$$

In this x represents the magnitude of the incoming value, divided by our *maximum* and clamped to be within 0 and 1.5. The magnitude of our resulting value is then computed by multiplying y and our *maximum* together.

0.1.4 Smoothing

To cater for any noise in the head data, the values are smoothed over a specified length of time. When deciding upon the optimal size of this duration, there is an tradeoff between responsiveness and jitteriness. In our previous approach [1]we chose to specify this duration via the size of the sample set. Having experienced our system out in the wilds though, we came to realise that this approach fails to produce a consistent sampling duration due to the unavoidable differences in user's hardware speeds. To rectify this, the sampling is instead specified via a *duration* (0.2 *secs* for both the handy-cam and leaning – see Fig. 1), while the sample size is allowed fluctuate between machines. Based on the preliminary reactions to the HAL technique, we further modified the filter to make it scale the handy-cam smoothing *duration* based the reported confidence of the tracking software (smoothed over 1*sec*). Based on the average confidences reported in the first HAL study, we used 10th and 90th percentiles values (0.48 and 0.7) to modify the duration on the fly ($duration = 1 + c(1 - (conf_{avg} - 0.48)/0.22)$, where c – initially set to 1.0 – controls how much confidence-correction occurs). The addition of this logic means the system is less jittery under bad tracking conditions, without compromising its responsive when the conditions are better. As the tracking confidence typically drops off during large movements, including those involved in performing a lean, it allows the system to gracefully suppress any erratic movements the tracker may produce.

0.1.5 Easing

To make the leaning feel more fluent when a dead zone is used and to also provide a finer level of control close to the neutral position, an ease-in/ease-out curve is applied to the data. Defined by the expression $f(v) = 3\|v\|^2 - 2\|v\|^3$, where the input value v is clamped between -1 and 1, this curve reduces smaller movements and amplifies the larger ones. In the event a dead zone is not being used, the use of this easing can appear to have a *snapping* effect around the

neutral position. As such, we allow for the level easing to be controlled by a tuneable *ease amount* (e = 1) such that the magnitude of the resulting value is defined by $e \times f(v) + \|v\|(1 - e)$.

0.1.6 Fading

To avoid abrupt jumps in the viewpoint when the user's head is first acquired following a period of non-tracking, we fade the data in over a specified *fading duration* ($2secs$). Likewise, if the system subsequently loses track of the player head, the last known data is faded out over the same duration as before. Because the data has already been centered relative to the user's neutral position and because at this position both the leaning value and handy-cam offset will be 0, implementing this filter can be achieved by simply varying an fading value between 0 (faded out) and 1 (faded in). In our implementation of this filter, we have chosen to use the same easing curve as mentioned before ($3x^2 - 2x^3$).

0.1.7 Other Elements

Following some concerns from players, the technique was modified to allow for the handy movements (namely the pitch and yaw components) to be suppressed during a lean. As shown at the bottom of Fig. 1, the amount of suppression is controlled by a tuneable value (s), along with the computed lean value (e.g. $yaw = yaw(1 - s\|lean\|)$). This modification was made to assist with the player with their aim during a lean.

References

[1] *Head Tracking in First-Person Games: Interaction Using a Web-Camera*, Torben Sko and Henry Gardner, INTERACT 2009, p342–355, Springer