Hawk: 3D Gestured-Based Interactive Bird Flight Simulation

Dartmouth Computer Science Technical Report TR2009-647

Thomas Yale Eastman
Advisor: Fabio Pellacini

June 2009

Hawk: 3D Gestured-Based Interactive Bird Flight Simulation

Thomas Yale Eastman
Advisor: Fabio Pellacini

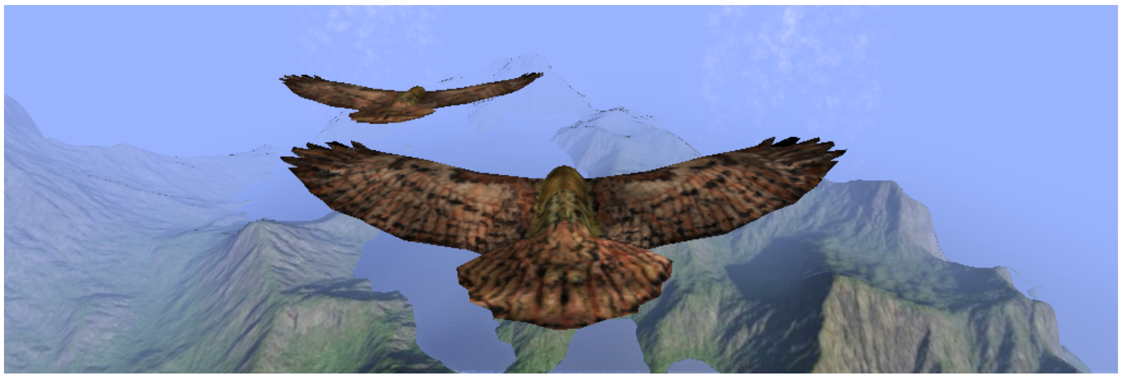
Figure 1: Player Flying Alongside an AI Hawk

Abstract

Control interfaces provide the most tangible connection between human users and computer software. This link is especially important in interactive real-time applications, like games and simulations, because users desire efficient controls that allow them to maximize their interactivity and immersion with the software. Traditionally, interfaces have been largely limited to keyboards and mice. Recently, however, technological advances have made motion-sensitive devices not only available to mainstream consumers but have also lifted restrictions limiting devices to two-dimensional motion. This work presents a 3-dimensional motion-sensitive interface alongside a natural application. Players can control a soaring red-tailed hawk and perform various intuitive flight maneuvers using two Nintendo Wii Remotes© (Wiimotes).

1 Introduction

Computer interfaces have long been restricted to simple input devices, most notably the keyboard and mouse. For many uses, these restrictions are not an issue. The keyboard is excellent for typing and the mouse's two-dimensional restriction is not a concern since it maps movement onto the two-dimensional computer screen. Nevertheless, computer graphics technology was only limited to two-dimensional image generation for a short time. Now, 3D software is commonplace, ranging from Google Earth© to complex video games. In these applications, the standard keyboard and mouse can only sometimes provide an efficient and intuitive mapping from user desire to software interpretation. This is especially apparent in computer games, where complex actions in 3-dimensional space must be controlled by two-dimensional inputs. This results in multiple disappointing gaps between player intention and software design. First, users must trace a steep learning curve to train their body to ignore obvious responses to visual stimuli, like players physically leaning around corners in a game, and learn new ones, like moving their thumbs slightly. Second, video game designers are limited to recycling non-intuitive reactions players have already learned or crafting games that are already limited to obvious mappings from keyboard or mouse or controller to game actions.

To explore and address these restraints, we examine the three-dimensional motion-sensitive interface provided by the Wiimote with the intent of creating a natural and intuitive control scheme for soaring bird flight. Section 2 will examine the Wiimote and it's output interpretation. Section 3 will explore our bird choice and its behaviors. Section 4 will discuss the flight model and its relationship to the target behavior. Section 5 will highlight various aspects of the system's implementation. Section 6 will conclude and propose future work.

2 Input Controls

Motivated by our goal to provide a natural and intuitive interface for soaring bird flight, an appropriate input device was researched and found. We settled on the Nintendo Wiimote for a variety of reasons. First, it has been used extensively in real-time interactive applications due to its role as the primary input controller for the Nintendo Wii video game console. That history, we hoped, would be a good indicator for the quality of accelerometer data and overall reliability. In addition, the Wiimote is commonplace due to the Wii's popularity, so they are not too expensive. Importantly, the Wii and Wiimote communicate using the open Bluetooth Wireless Protocol, which computers often use for wireless input devices as well. Finally, the Wiimote contains two complementary methods of detecting motion. It contains both an ADXL330 3-

axis accelerometer and a PixArt optical sensor for use in conjunction with an infrared sensor bar. Unfortunately, as we'll discuss later, the infrared camera was not useful to us due to the goal that users find flight natural. Restricting users to always have the screen in view of the Wiimote's camera was simply not feasible given the wide range of possible input positions. So, our input for controlling the player's avatar comes solely from the accelerometer.

Figure 2: Nintendo Wii Remote

The Wiimote's accelerometer reports acceleration data from three axes aligned with the Wiimote's casing. In the standard orientation, with the Wiimote flat on a level surface with its buttons facing up, the accelerometer will report approximately [0,0,1], which demonstrates that it has been normalized to gravity and that the z-axis is positive normal to the button-face. (Figure 2) With gravity always present during usage sessions, the Wiimote is a functional tilt sensor. Note, however, that this is of no help when trying to orient the Wiimote in space. The Wiimote lying on the table will report the same value regardless of its rotation through the z-axis on the table. As we will see in the next section, tilt sensing provides a large subset of the data required for natural soaring. However, dynamic acceleration measurements are required for flapping. Unfortunately, this means that tilt sensing is, at best, unreliable during flapping since the Wiimote is moved quickly in all three axes during flapping. In addition, linear acceleration cannot be easily discriminated from gravitational tilt during small time frames, which means that the two modes of accelerometer interpretation are largely incompatible. These issues are also exacerbated by the Wiimote's noisy accelerometer readings and Bluetooth communication latency. Not surprisingly, Nintendo has noted these issues and will release the Wii Motion Plus, a Wiimote add-on with 3 gyroscopes, shortly after this work's completion. The add-on will certainly help discern rotational from linear motion. Unfortunately, we are without the add-on.

3 Bird Choice

Still motivated by the set goal of providing an interface for natural and intuitive flight, we sought out a bird with flight characteristics that can be readily performed by humans. Identifying turning, diving, soaring, and flapping as the key bird flight behaviors that humans can mimic without trouble, we chose a bird species that performs them often and with appealing style, the Red-Tailed Hawk. These hawks live and hunt in areas where large soaring wings with high aspect ratios might be hazardous, so they have shorter wings than some soaring birds but make use of numerous wingtip slots to increase lift and reduce wingtip vortex lag (Savile). (Figure 3) The hawks primarily hunt from perches, of which they are very protective. Trees, especially dead ones, tall rocks, and other high perches allow the hawk to spot small mammals, especially squirrels, with minimal exertion. When hunting from a perch, the bird must dive and flap rapidly to catch prey. However, when soaring, the hawk will rarely flap its wings, instead relying on thermals to retain altitude. For hawks, soaring is not as effective for hunting as using a perch but is nevertheless a common behavior for hawks due to its importance for territory control and courtship displays (Ballam).

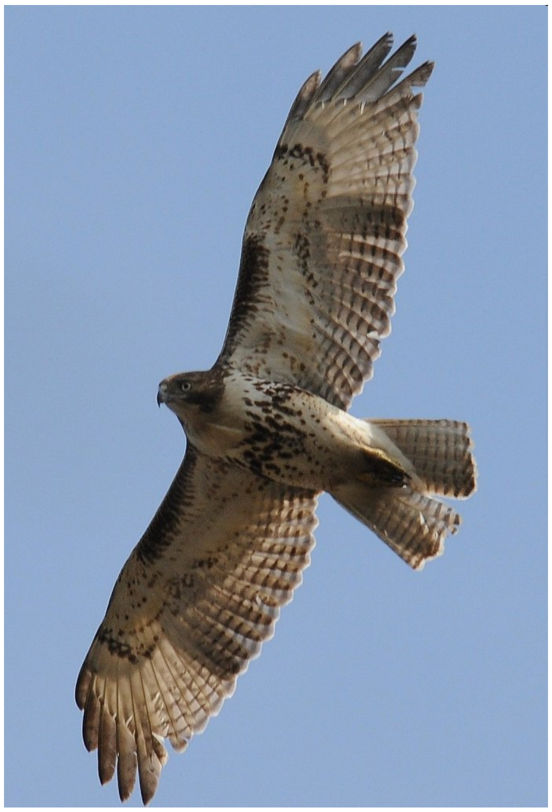
Figure 3: Red-tailed Hawk

Territory defense is very important for the hawks due to the usual scarcity of good hunting perches. Hawk pairs controlling territory with few perches can deplete prey in the surrounding area and thereby reduce their reproductive fitness. This problem highlights the real value of land control: the perches and not total area (Janes). To protect their territory, hawks will attack low-flying invaders and especially perch-stealing invaders by diving, screaming, and attempting to knock the trespasser from the air or perch. Since diving is the preferred method of attack, the defender must exert great effort to gain altitude for each attack. The importance of height during attacks means that soaring birds have not only that advantage but can also see intruders more readily. Trespassers are usually juvenile hawks without territory or mate, and so must use hidden perches and avoid soaring flight while living in contested regions. Soaring is also important for courtship displays, which often involve multiple vertical dives and stalls (Fitch).

4 Flight Model

Having identified the Red-Tailed Hawk as the player's avatar, we can match its behaviors with our human-analogous flight abilities, soaring and flapping. Although these actions align well between species, we made some changes to facilitate human enjoyment. Most notable among these is to increase the flight speed by a factor of approximately five. Soaring, especially without flapping, lacks the excitement usually associated with flight. The same process of merging behaviors applies to mapping the

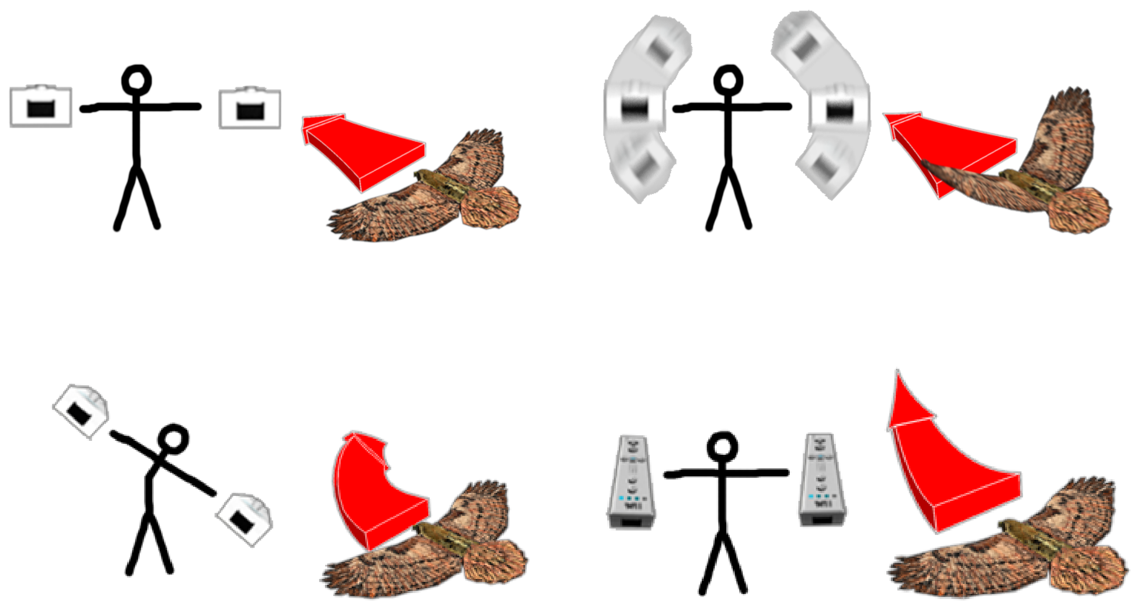

Figure 4: Flight Gestures

control scheme from Wiimote to avatar actions. There are three major flight gestures that are transferred across the interface: tilting left/right, pitching up/down, and flapping. (Figure 4) A fourth avatar behavior, diving, is induced automatically as a result of player diving behavior, specifically non-flapping pitching down. Turning and flapping (Figure 4cd), as well as their opposites, can be combined along their spectra to produce soaring behavior. Unfortunately, as discussed before, flapping flight cannot be combined with other methods due to the Wiimote's inaccuracy during dynamic motion contexts. Despite the seeming simplicity and nice orthogonality of Wiimote gestures shown in Figure 4, playtesters had trouble holding the Wiimote at right angles. This problem prompted the creation of a basic and largely hidden training function to convert casual user grips to the well-defined basis where the Wiimote accelerometer axis align with gravity and the player's facing direction. The goal of this training function is, as simply as possible, to have the user hold the Wiimote as they desire and have the flight occur as they expect. The requirement of simplicity removes the possibility of multiple calibration steps, which precludes full orientation-independent training, but the system settled upon provides a natural training solution with only the requirement that the long axis of the Wiimote (IR camera face) be aligned with the vertical. Upon simple instruction, most users adopt the correct grip (Figure 6), though the system will properly train on any rotation of the Wiimote about the X and Y accelerometer axes. Note that those changes produce different tilt readings, while yaw rotations, as mentioned previously, will not change the direction of gravity relative to the accelerometers. Because of this limitation, he conversion matrix from the trained accelerometer values to the standard one can be computed simply. First, the rotational change in the two monitored axes is calculated relative to the standard base.

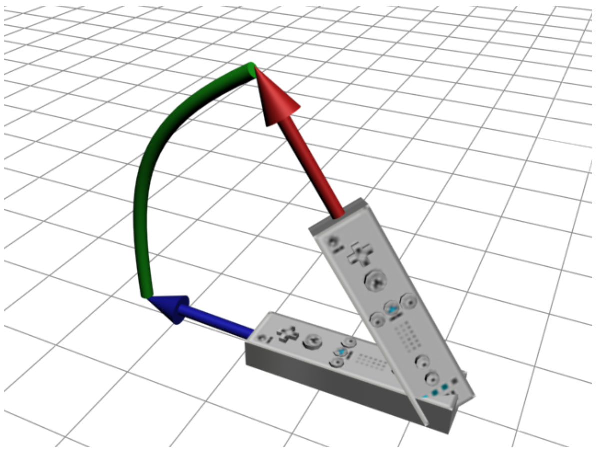
Figure 5: Change of Base

$$roll = \tan^{-1}\left(\Delta\vec{b}_x, \Delta\vec{b}_z\right)$$
$$pitch = \tan^{-1}\left(\Delta\vec{b}_y, \max\left(\Delta\vec{b}_x, \Delta\vec{b}_z\right)\right)$$

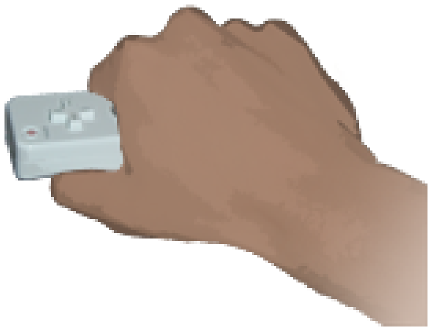
Figure 6: Standard Grip

From this information a rotation matrix M_R is constructed against which any new accelerometer data b_c is multiplied to convert it from the new base to the standard one.

$$\begin{bmatrix} roll \\ pitch \\ 0 \end{bmatrix} \Rightarrow M_R$$

$$\vec{b}_s = M_R \vec{b}_c$$

This simple and continuous conversion provides the necessary tolerance to allow players to immerse themselves in the simulation without worrying about the control mechanic.

The final aspect of the flight model is that governing the flight of the hawk through the air. There are four major forces acting on powered flying objects: lift, drag, thrust, and gravity. In the following equations, $\hat{u}$ represents the hawk's up vector (green in Figure 7). Likewise all symbols with hats are unit vectors. F represents the forward vector of the bird (yellow in Figure 7) and v represents the hawk's velocity. c

with a subscript stand for various coefficients. Gravity is the simplest force here since it is independent of the bird's orientation.

$$F_{Gravity} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

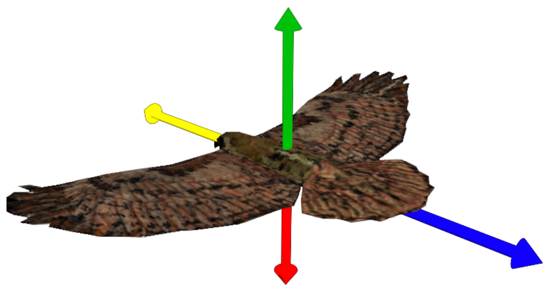

Figure 7: Flight Forces

Lift is generated by the airfoil-shaped wings of the hawk as air flows over and under the wings. Lift is only produced as air flows in the direction of the wings' angle of attack and is scaled by the lift coefficient of the surface.

$$F_{Lift} = -c_l \hat{f} s \left(\hat{f} \cdot v \right)$$

Drag is the force opposing motion through a fluid and dependent on the square of the velocity and the cross section of the object perpendicular to that movement. For this reason, I split the drag into two main components: the leading edge drag (blue in Figure 7) and the falling drag (green or red directions in Figure 7). The leading edge drag is the force that slows the hawk down in normal flight. Not surprisingly, the bird is streamlined in this direction and has a very low cross section, resulting in a very low drag coefficient, c_{dl}.

$$F_{LeadingEdgeDrag} = -c_{d_1}\hat{f}s\left(\hat{f}\cdot v\right)$$

The other component of drag slows movement of the hawk through its vertical axis when its wings are spread. This time, the cross section of the bird is at its greatest and so the drag coefficient is much higher, despite the similarity to the previous equation. This drag is especially important during sharp banking, where the hawk's speed must be retained while shifting direction.

$$F_{FallingDrag} = -c_{d_2}\hat{u}s\left(\hat{u}\cdot v\right)$$

Finally, flapping provides a forward force. Bird flapping produces a tremendously complicated series of vortices that serve to lift and propel the hawk forward, but we have chosen to simplify the model here both for time constraints and to make the interface more transparent for users.

$$F_{Flapping} = c_f\hat{f}$$

Occasionally, the bird will enter a thermal, which provides a vertical force similar to the opposite of gravity.

5 Implementation

To facilitate the creation of an application for this system, we chose to use Garage Games' *Torque Game Engine Advanced* because it was readily available with source code. This provided the framework for adding a new hawk character, large terrains, and Wiimote inputs. The engine is written in C++ and supports the TorqueScript scripting language. The primary addition made was the integration of the Wiimote into the input system and the analysis of its accelerometers. This was done in both C++ and TorqueScript. The C++ side contains the low-level interface, change of base function, and calls script functions to report changes in the accelerometers. The script side, on the other hand, does most of the motion analysis. It does this in two main parts: the steering updates and flapping activation. As mentioned before, the first step in motion analysis is rotating the accelerometer data to fit the standard base. Once that function is complete, the code is independent of how the user is holding it and can more easily analyze motion since the standard axes align with gestures more clearly. Turning and pitching map directly from the X- and Y-axis, respectively, since they are usually controlled and stable motions, allowing tilt-sensing to be accurate. The accelerometer values are then used to set the steering angles for the bird. Similarly, they control the wing and tail angles on the hawk model. Flapping, however, is detected using linear motion through the Z-axis and so is handled very

Figure 8: Low Soaring

differently. Flapping detection has three simultaneous requirements: the direction of the flap must alternate each stroke, both wings must be moving in the same direction, and consecutive strokes are close together. As noted previously, steering is limited during flapping because tilt-sensing is inaccurate during dynamic acceleration. As a consequence of these values, the hawk's diving behavior is also started or stopped, based simply on the magnitude of the hawk's forward speed projected along the negative Z-axis. Another aspect of this system are the AI hawks, which simulate player inputs to reach random destinations above the level. Simply put, they lazily turn towards targets when far away, flap when at low speeds, and will circle to gain altitude instead of flapping at high angles of attack. During normal play, their most obvious feature is a red particle trail, which gives away their position, which would otherwise be impossible to see due to the diminutive size of the hawks.

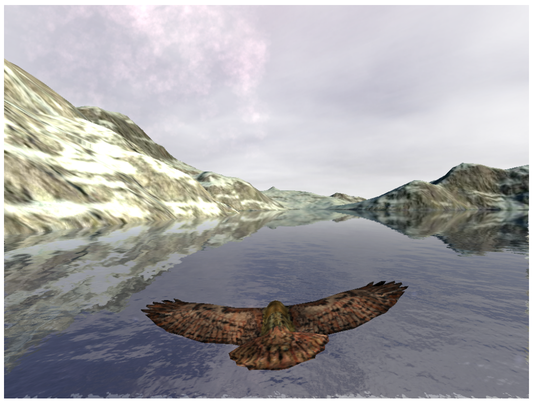
Figure 9: Arctic Generated Terrain

The next major addition is the terrain system. Although the Atlas terrain system was disabled and unsupported, we reimplemented it because the default terrain could not hold enough detail to properly display the approximately 13 kilometer square territories needed. Using L3DT, a terrain generation program, we designed multiple island landscapes.

The final and most visually important component is the hawk model itself. Modeled in 3D Studio Max, it is composed of 436 triangles weighted for animation control by 8 bones. It has a variety of animations that are blended together while in flight, including axis-separated rotations of the tail and distinct sequences for each wing. The hawk's texture is based on feather drawings and photos of red-tailed hawks.

Other minor code additions include a Wiimote debug display, thermal generation, graphical user interface, and numerous bug fixes.

6 Conclusion and Future Work

We believe that all three of the initial goals: natural flight, clear flight model, and immersive art, have been met with this work. Soaring is achieved with Wiimote tilt-sensing and is supplemented by change-of-base functionality that reduces user training drastically. The flight model is clear and intuitive, as well as based on actual physics principles. And lastly, the aesthetics of the game contribute considerably to the user's sense of flight. In addition, we learned a great deal about hawk behavior and flight, modeling, animation, accelerometer analysis, and game engine programming.

Some aspects of the design did not go as planned, especially concerning the game engine and Wiimote. TGEA's stability, art pipeline, and inconsistent feature set left much to be desired. Large amounts of time were spent attempting to make the engine do what it supported already. Also, the Wiimote sensors were not as accurate as initially hoped. Although Nintendo is addressing the problems soon with the Wii Motion Plus add-on, the Wiimote itself is truly useful only in limited applications, unlike the ideal 3-dimensional motion-sensing device that it is often billed as.

Future work extends logically from this work. A full game can be constructed with this flight model as its base, with territory as contested property between players and AI hawks. In addition, there are opportunities to enhance the flight model by adding support for more complicated behaviors like taking off, landing, barrel rolls, and stalls. Further examination into the best animation to represent flapping would be fruitful as well. Currently, the flaps correspond quite directly to player motion but can therefore look awkward because hawks rarely flap like humans. Similarly, more accurate flight gestures could be encouraged with visual positive feedback. Behaviors like proper wing strokes and wing angles could be taught this way. Furthermore, the current behaviors and new ones would be all enhanced by the use of the Wii Motion Plus or a more accurate position and acceleration sensor.

References

ANALOG DEVICES. 2006. ADXL330: Small, Low Power, 3-Axis ±3*g* *i*MEMS® Accelerometer. http://www.analog.com/static/imported-files/data_sheets/ADXL330.pdf

BALLAM, JM. 1984. The use of soaring by the Red-tailed Hawk (Buteo jamaicensis). The Auk. ISSN 0004-8038.

FITCH, HS. 1946. Behavior and food habits of the red-tailed hawk. The Condor, p205 ISSN 0010-5422.

HEDEMSTROM, A. Migration by Soaring or Flapping Flight in Birds: The Relative Importance of Energy Cost and Speed. Philosophical Transactions: Biological Sciences, 342(1302), p353. The Royal Society

JANES, SW. 1984. Influences of territory composition and interspecific competition on red-tailed hawk reproductive success. Ecology, p862. ISSN 0012-9658

LEE, J.C. 2008. Hacking the Nintendo Wii Remote. Pervasive Computing, IEEE 7(3) p39

NINTENDO WORLD REPORT. 2006. Nintendo and PixArt Team Up. http://www.nintendoworldreport.com/newsArt.cfm?artid=11557

POORE, SO. 1997. Wing upstroke and the evolution of flapping flight. Nature, 387 p799. ISSN 0028-0836

RAYNER, J. M. V. 1979. A New Approach to Animal Flight Mechanics. J Exp Biol 80, p17

SAVILE, DBO. 1957. Adaptive evolution in the avian wing. Evolution, p212. ISSN 0014-3820.

THOMAS, ALR. 1993. On the Aerodynamics of Birds' Tails. Philosophical Transactions: Biological Sciences, 340(1294) p361. The Royal Society.

WU, J. 2003. Realistic Modeling of Bird Flight Animations. ACM Transactions of graphics, 22(3) p888. ISSN 0730-0301

Image Credit

Figure 2:
http://upload.wikimedia.org/wikipedia/commons/b/bc/Wii_Remote_Image.jpg

Figure 3:
http://upload.wikimedia.org/wikipedia/commons/c/c4/Red_tailed_hawk_saoring_Maryland_USA.jpg