Real-Time Jump Detection using IMU Sensors for VR Snake Game

Nick Ping
Stanford University
450 Jane Stanford Way, Stanford, CA, USA
nping2@stanford.edu

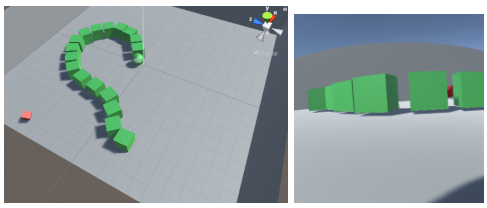

Figure 1. A third-person view (left) and a first-person view (right) of the VR Snake game. The green chain of cubes represents the virtual snake and the red box represents the collectible.

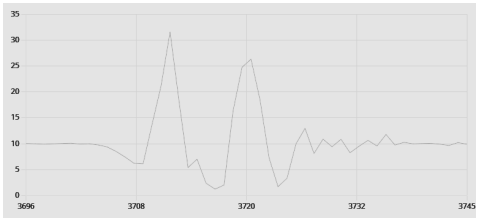

Figure 2. Acceleration magnitude over time for a sample physical jump action.

1. Introduction

The classic class of arcade games *Snake* first began in 1976 under the name *Blockade*. Numerous variations have since been developed, all building upon the core mechanics of navigating a growing virtual snake.

I implement a variation of the classic *Snake* arcade game in virtual reality (VR) using the Unity game engine. In this version of *Snake*, the player plays as the continuously advancing snake in the first person, steering using gaze direction. Red collectibles appear one at a time in random locations across the map; collecting them increases both the player's score and snake lengths. The player will lose the game if they run into a wall or the snake's own tail and become unable to move at the same speed for a sufficiently long time. Players can physically jump to trigger an in-game jump, allowing them to avoid collision with the tail.

2. Jump Detection

For Jump Detection, I utilize an Inertial Measurement Unit (IMU) accelerometer with the Arduino Teensy 4.0 Development Board. I compare different detection strategies including free fall detection, and flight time height estimation.

2.1. Free Fall Detection

During a jump sequence, the player will be momentarily in free fall where only gravity acts upon player and the attached IMU. This causes the acceleration magnitude from the accelerometer sensor readout to briefly approach zero.

2.1.1 Implementation

Due to sensor noise and human movement during travel, the actual accelerometer sensor magnitude reading will not be perfectly zero. To address this, I select a sufficiently low threshold which detects the jump when the acceleration vector magnitude dips below. Through experimentation, a fairly generous magnitude threshold is 0.5 g.

2.1.2 Results

This simple jump detection strategy is easy to implement and fast to run on the Teensy development board. However, this method is not very robust due to many sources of false positives including detecting squats as jumps.

2.2. Flight Time Detection

If I require an estimate of the jump height as well as jump detection, I may also use the flight time to obtain an estimate of jump height [1]. A jump action typically involves two spikes in acceleration magnitude marking the takeoff and landing phases of the jump (Figure 2). By measuring the timing delays between takeoff and landing, we may obtain an estimate of the jump height (Equation 1) [2].

$$\Delta y = (9.81 \text{ m/s}^2)\frac{t^2}{8} \tag{1}$$

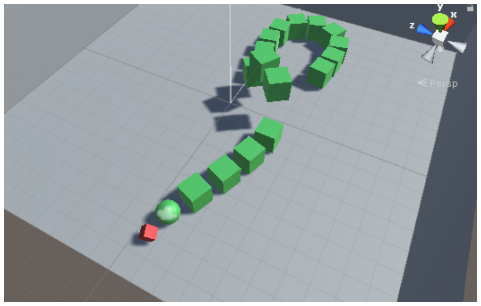

Figure 3. A third-person view of the snake after a jump.

2.2.1 Implementation

I select a sufficiently high threshold for acceleration magnitude. When the acceleration first exceeds the threshold, a timestamp of the takeoff time is recorded. Once the acceleration exceeds the threshold for the second time, we should be in landing phase and compute the time difference between takeoff and landing. Then we may obtain an estimate of jump height using Equation 1. I also implement a decay counter that will timeout the jump. This prevents mistaken detections where only a single acceleration magnitude spike is detected. A jump is detected if the calculated jump height is above a chosen threshold.

2.2.2 Results

This jump detection strategy is more complex compared to the Free Fall Detection, but runs almost as fast on the Teensy development board. The method is more robust compared to the Free Fall Detector but suffers from higher rate of false negatives from not detecting either the takeoff spike or landing spike.

2.3. Detector Fusion

Both Free Fall Detection and Flight Time Height Estimation strategies address independent aspects of the jump, so we may combine the two strategies that requires both a momentary free fall and exceeds a jump height threshold.

2.3.1 Results

Since this detection strategy requires more conditions to be fulfilled, the false negative rate is much higher. For playability, I choose to generously allow a higher false positive rate. Hence the final version uses Free Fall Detection only and not incorporate Flight Time Height Estimation.

3. Software

3.1. Gaze Directed Steering

I utilize the IMU gyroscope measurements to calculate the associated world coordinate rotation quaternion and Euler angles. The measured world up direction rotation angle is used to update the lateral snake position (Equation 2)

$$\Delta x = v\sin(\theta_y), \quad \Delta z = v\cos(\theta_y), \quad v = \text{velocity} \quad (2)$$

3.2. Snake Growth

I maintain a body instance list and body position First-In-First-Out (FIFO) buffer to implement an append-able trailing body. For smooth animation, the position buffer is larger than the body instance list by a constant factor C. At every frame of the game where the snake is moving, the camera position gets pushed into the position buffer and the k body instance position will update with the $(k+1)C-1$ element of the body position FIFO. The orientation of each body instance will be in the direction of its own position and the previous body instance link position.

When the snake head, at the camera position, contacts the collectible on the map we append another body instance to the body instance list and appends an additional C number of the last entry to the body position FIFO.

4. Conclusion

This implementation of a VR Snake Game is a fun and interactive way to investigate different Jump Detection strategies with an qualitative error analysis. With only a 6 Degree of Freedom IMU, jump detection can be successful but is prone to false positives as well as false negatives.

In future extensions, I would like to investigate obtaining more reliable 3D pose tracking with a light house. With more reliable 3D position data, jump detection may be much more reliable by incorporating a more reliable jump height calculation. Additional body sensors may also be included to better define the criterion for jumping including specific limb movement.

References

[1] S. Marković, M. Dopsaj, S. Tomažič, A. Kos, A. Nedeljković, and A. Umek. Can imu provide an accurate vertical jump height estimate? *Applied Sciences*, 11(24), 2021.

[2] S. Schleitzer, S. Wirtz, R. Julian, and E. Eils. Development and evaluation of an inertial measurement unit (imu) system for jump detection and jump height estimation in beach volleyball. *German Journal of Exercise and Sport Research*, 2022.