BeeMaster Engine
BEEMASTER LAB

TECHNICAL NOTE LL-001
VERSION 1.0 · NOVEMBER 2026

LAB LOG #001 CHARACTER MOVEMENT · TESTING

Walking Speed and the Step-Up Problem

Why a capsule character controller climbed stairs at 4 m/s but refused the first step at 1.4 m/s, what we changed, and what we measured afterwards.

BeeMaster Lab, OpenLabWork openlabwork.com/notes Companion to the video “Lab Log #001”

it passed... at the wrong speed

ABSTRACT

Most real-time engines move a walking character as a capsule and climb small steps with a three-move sequence: rise by the maximum step height, move forward, settle down. Our implementation of this sequence passed its stairs scenario at 4.0 m/s. When characters later had to *walk* up the same stairs at 1.4 and 1.15 m/s, every one of them stopped at the first step and never recovered. At a 60 Hz simulation step, a walking character advances only about 2.3 cm per frame; after such a small move the capsule’s rounded bottom settles onto the edge of the step rather than its top, and the contact response of a sphere on a corner pushes it backwards whenever its centre is still behind the edge. We describe the geometry, the change that resolved it, two related defects in foot placement that the climb exposed, and a measurement run of six characters over 600 steps each: 3,880 planted foot-steps with zero planted-foot slip, no ground penetration, and the lower foot on its own step in every sampled stair step, at a cost of about 50 μs per character per frame in a debug build. The broader lesson is about testing: a movement test that passes at one convenient speed says nothing about the others.

Keywords: character controller · capsule · step-up · contact normal · fixed timestep · foot planting · regression testing

1 Why stairs

Stairs are rarely the point of a game, and that is exactly why they matter to us. A character in a game world should simply deal with the things around it: a step, a ledge, a door, a box to carry, a chair to sit in. In many projects each of those becomes a small project of its own, with a dedicated animation, a trigger volume, a script and values tuned by hand until the result looks acceptable. Multiplied by every object in a game, that work is where a small team’s time goes.

BeeMaster is being built on a different bet: that an engine should understand what things are and how a body meets them, so that the natural interaction happens by default and the developer goes deeper only when they want to. Walking up stairs is the first and simplest test of that idea. This note documents what happened when we ran it.

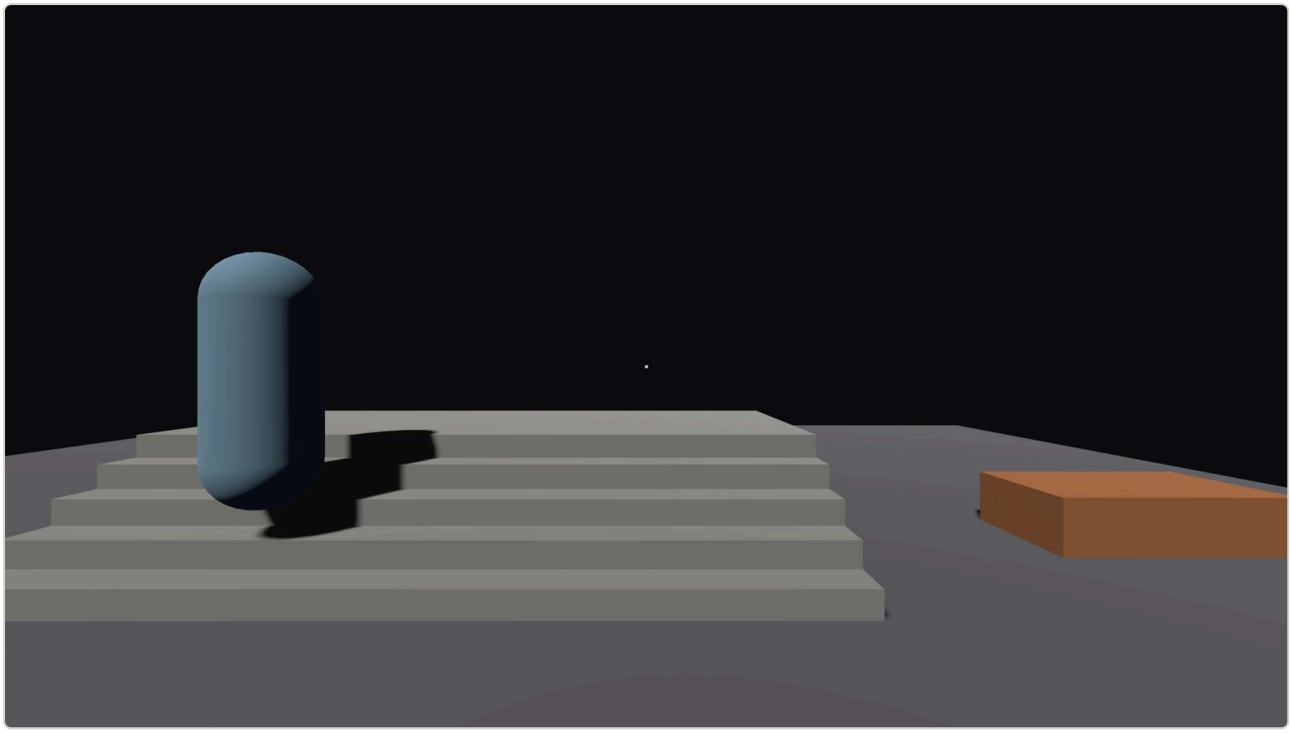


Figure 1. The stairs fixture in the engine's grey-box test scene, with the capsule that represents a walking character for collision. Test geometry, not finished art.

2 Background: how a capsule climbs a step

For collision purposes, most engines do not simulate a walking character as a body with legs. They use a *capsule*: a cylinder capped by two half-spheres. The shape is cheap to test against the world, has no edges to snag on small bumps, and its rounded bottom slides naturally over uneven ground. Engine documentation commonly exposes the climbing behaviour as a single parameter, such as a step offset or maximum step height [2–4].

The usual step-up sequence is performed every simulation step: (1) **rise** by the maximum step height, (2) **move** forward by the frame's horizontal displacement, and (3) **settle** down onto whatever supports the character. In BeeMaster the maximum step height is 0.30 m; anything taller is treated as a wall. The simulation runs at a fixed 60 steps per second, a common choice that keeps physics deterministic and independent of rendering rate [5].

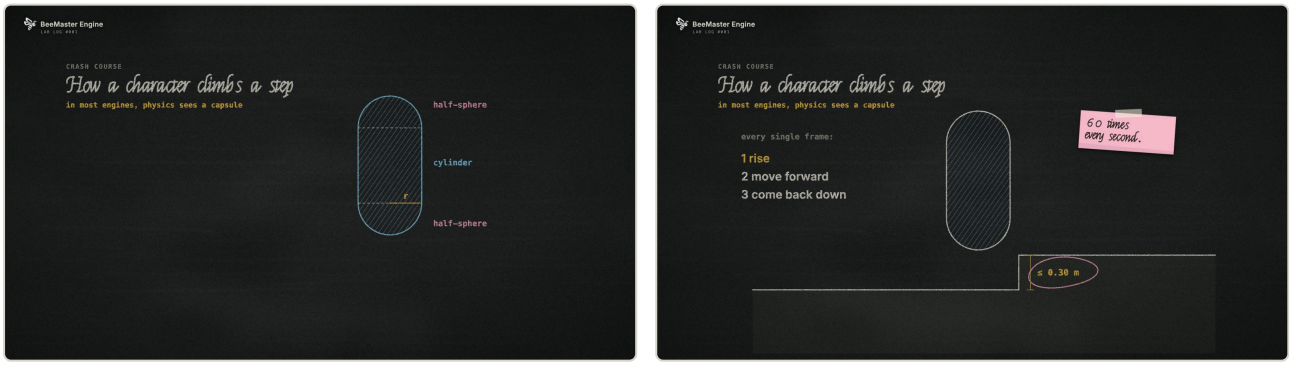


Figure 2. Left: the capsule, a cylinder with two half-spheres of radius *r*. Right: the three moves performed every frame to climb a step of at most 0.30 m.

3 Observation: a test that passed

Our first stairs scenario placed a player character and one non-player character on a short flight of stairs. Both walked up, walked back down and stopped at a ledge taller than the step limit. The scenario passed. Its movement speed was 4.0 m/s, which is a run, not a walk.

A day later we built a harder scene: six characters moving around a small test level, each exercising something different (walking, carrying a box, keeping its gaze on a target, limping, and two that must climb the stairs). The two stair characters moved at walking speed, 1.4 and 1.15 m/s. Both reached the first step and stopped there permanently.



Figure 3. Left: the six-character test level in the engine (box-shaped test figures). Right: schematic of the same failure; at 4 m/s the capsule climbs, at 1.4 m/s it is refused at the first edge on every frame.

4 Analysis: a sphere on a corner

4.1 How far a character moves in one frame

With a fixed step of 1/60 s, the horizontal displacement per frame is

```
Δx = v / 60                                   (1)
```

Speed v (m/s)	What it is	Δx per frame (cm)
0.50	careful step	0.83
1.15	slow walk (Courier)	1.92
1.40	walk (Climber)	2.33
2.00	brisk walk	3.33
3.00	jog	5.00
4.00	run (original test)	6.67

At a walk the forward move is roughly a third of what the original test exercised ($6.67 / 2.33 \approx 2.9$).

4.2 Where the capsule lands

After rising and moving forward by only about 2 cm, the capsule settles. Its centre of the lower half-sphere is still behind the step's edge, so its rounded bottom does not come down on the top of the step; it comes down on the edge itself. Let the edge be the origin, the lower half-sphere have radius r, and its cen-

tre be a horizontal distance d behind the edge, with $0 < d < r$. The sphere touches the corner at a height

$$h = \sqrt{(r^2 - d^2)} \tag{2}$$

above the edge, and the contact normal (the direction in which a collision response pushes the sphere) points from the corner to the sphere's centre:

$$n = (-d, \sqrt{(r^2 - d^2)}) / r \tag{3}$$

Its horizontal component, $-d/r$, points backwards whenever the centre is behind the edge. Resolving the contact therefore moves the capsule back off the step. The step is refused, and on the next frame the same sequence produces the same result. The character is not slow; it is stuck in a stable loop. This is general contact geometry [1], not a property of any one engine.

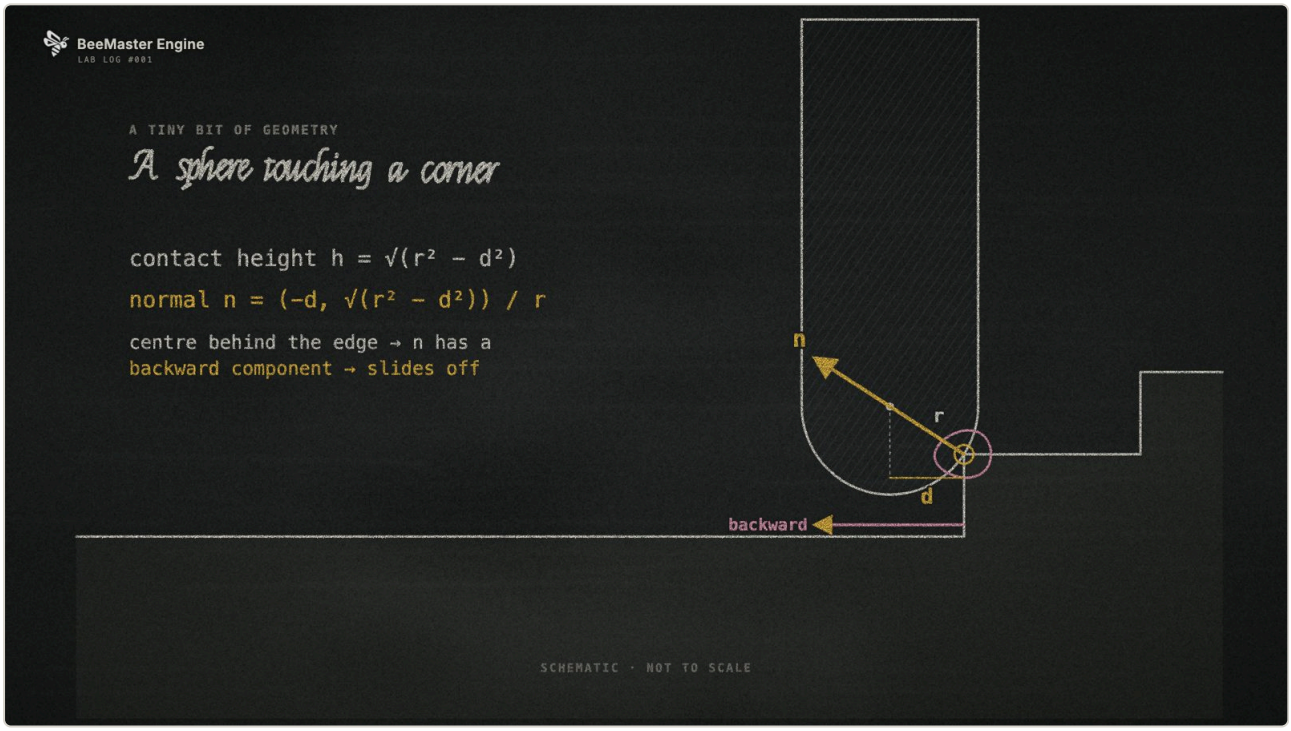


Figure 4. The capsule settling on the edge. The contact normal n points from the corner to the centre; while the centre is behind the edge, n has a backward component and the capsule slides back off.

4.3 Why the original test did not see it, and why a box never would

At 4.0 m/s each frame advances 6.7 cm, about three times further, and in our scenario that was enough for the step to be accepted. We have not mapped the exact speed threshold, which depends on capsule radius, step height and the settle logic; rather than tune for it, we now test across the whole speed range (Section 8). A box-shaped body behaves differently: a flat bottom meeting a square corner produces a normal that points straight up, with no backward component. A test that used a box, or a run, could never have exposed the defect.

> FINDING
>
> Our stairs test passed because it moved at a speed nobody actually walks at.

5 The change

The resolution was small. When the top of the step is already under the character's footprint, within one step's height, the character is set down there, the same way a box would stand on it. With this rule in place, the Climber and the Courier walked up the stairs at 1.4 and 1.15 m/s.

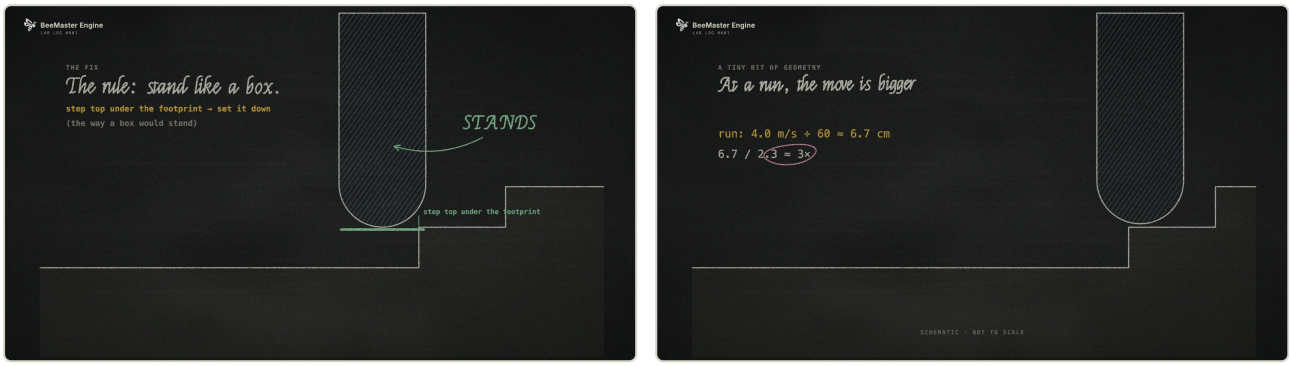


Figure 5. Left: once the step top is under the footprint, the capsule is set down and stands. Right: at a run the move per frame is 6.7 cm, which is why the original scenario passed.

6 What climbing exposed: the feet

Once the characters could climb, we examined their feet frame by frame. On the stairs, a foot that was supposed to be planted (standing still on a step) jumped 12.5 cm in a single frame, and the leg rotated 31° with it. In a still image this is invisible; in motion it is the moment a character stops feeling as if it has weight.

The cause was a question of trust. Foot placement had been asking the animation where a foot *should* be planted instead of observing where the foot actually *was*, and whether the leg could still reach it. The rule is now: a foot plants where it really is, and only within its leg's reach. If the body has moved on, the foot neither snaps nor slides; it lifts, steps and lands again.

A second surprise concerned the collider. Its flat footprint could already be standing on the next stair while the character's hips were still above the previous one, a full step ahead. Body corrections are therefore measured from the ground under the body, not from where the collider reports its support.

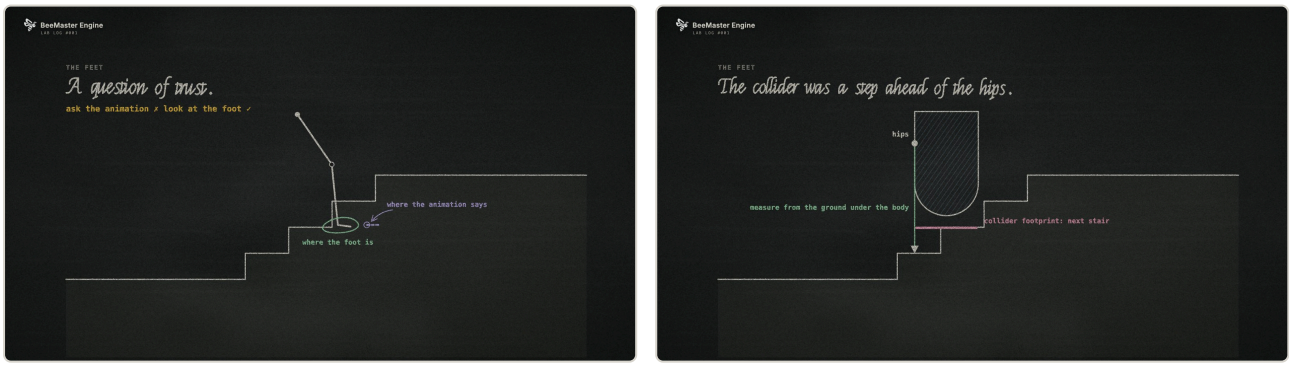


Figure 6. Left: the animation's target (dashed) versus the foot's real position. Right: the collider footprint already on the next stair while the hips are still over the previous one.

7 Measurement

7.1 Method

Measurements were taken from the engine's own trace output, sampled at 60 samples per second: positions of the hips, knees and ankles, and the planted or stepping state of each foot. Six characters each walked 600 steps through the test level. Values were recorded in a debug build on our development machine. The figures below are measured values from that run; they are not projections.

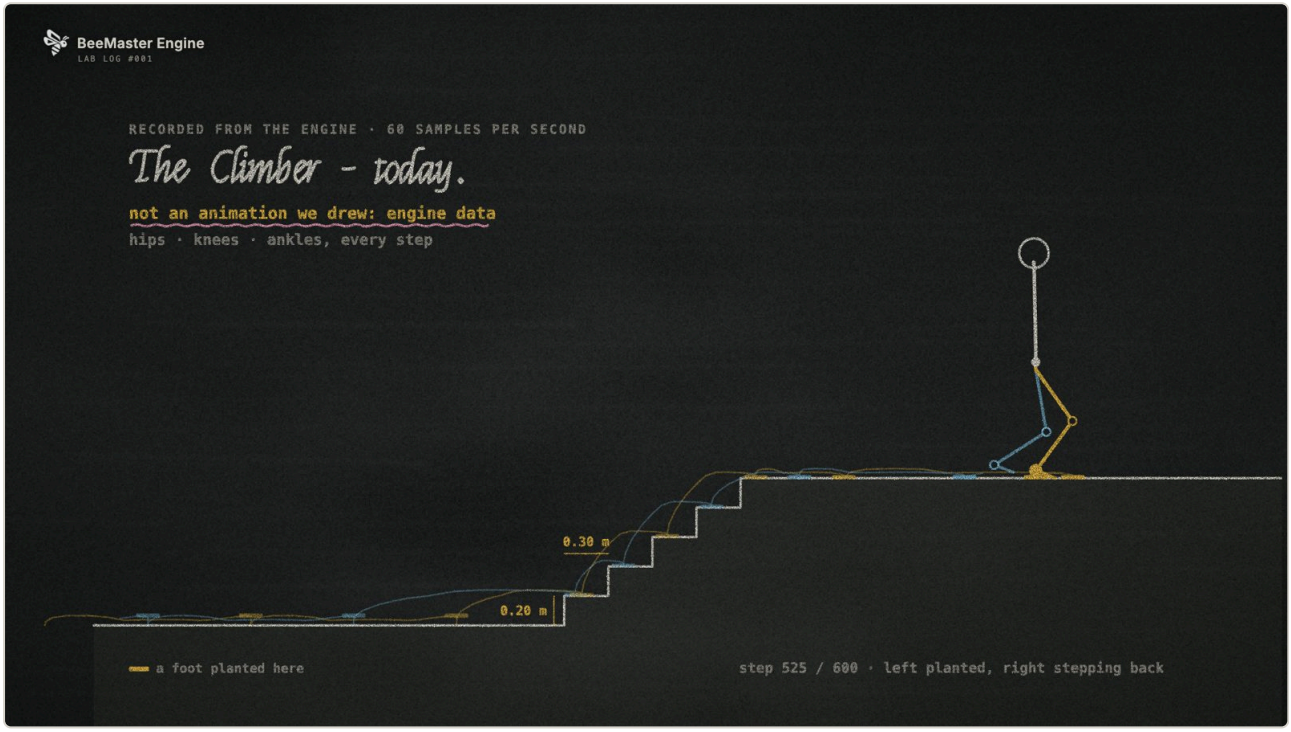


Figure 7. The Climber walking up the stairs, drawn from recorded engine data: joint traces at 60 samples per second; each mark on a step is a moment a foot planted.

7.2 Results

Measure	Before	After
Characters × steps	—	6 × 600
Planted foot-steps / planted-foot slip	—	3,880 / 0 m
Feet into the ground	—	0
Lower foot on its own step (sampled stair steps)	61/82 (74%) · 83/100 (83%)	184/184 · 220/220
Largest planted-foot jump in one frame	12.5 cm · 31°	0 m
Cost per character per frame (debug build)	—	≈ 50 µs

"Before" values are from the run that exposed the defects; "—" marks measures that were not recorded in that form before the change.

8 Lessons for testing movement

The code change was a few lines. The lesson was larger. A movement test that passes at one convenient speed tells you nothing about the others. Since this episode, everything that moves in BeeMaster is tested:

- **across the speeds a game actually uses**, including the slow end, not just a single value;
- **with the real shape of the character**, not a convenient box;
- **by looking at the feet**, frame by frame, not only at whether the body arrived.

Each of these is now part of our movement test scenarios, so this class of defect is checked whenever movement changes.

9 Limitations and next steps

The analysis in Section 4 is two-dimensional and treats the step edge as a sharp corner; real geometry can be bevelled, which changes the contact height but not the sign of the backward component while the centre is behind the edge. We have not characterised the exact speed at which acceptance changes. Measurements come from one test level and one debug build. Stairs are the first interaction on our list; ledges, boxes and chairs follow, and we will report them the same way.

10 References

1. Ericson, C. *Real-Time Collision Detection*. Morgan Kaufmann, 2005.
2. Unity Technologies. *CharacterController.stepOffset*. Unity Scripting API documentation.
3. Epic Games. *UCharacterMovementComponent: MaxStepHeight*. Unreal Engine API documentation.
4. NVIDIA. *Character Controllers: Auto Stepping*. PhysX SDK documentation.
5. Fiedler, G. “Fix Your Timestep!” *Gaffer On Games*, 2004.