FATEMEH SHIRVANI

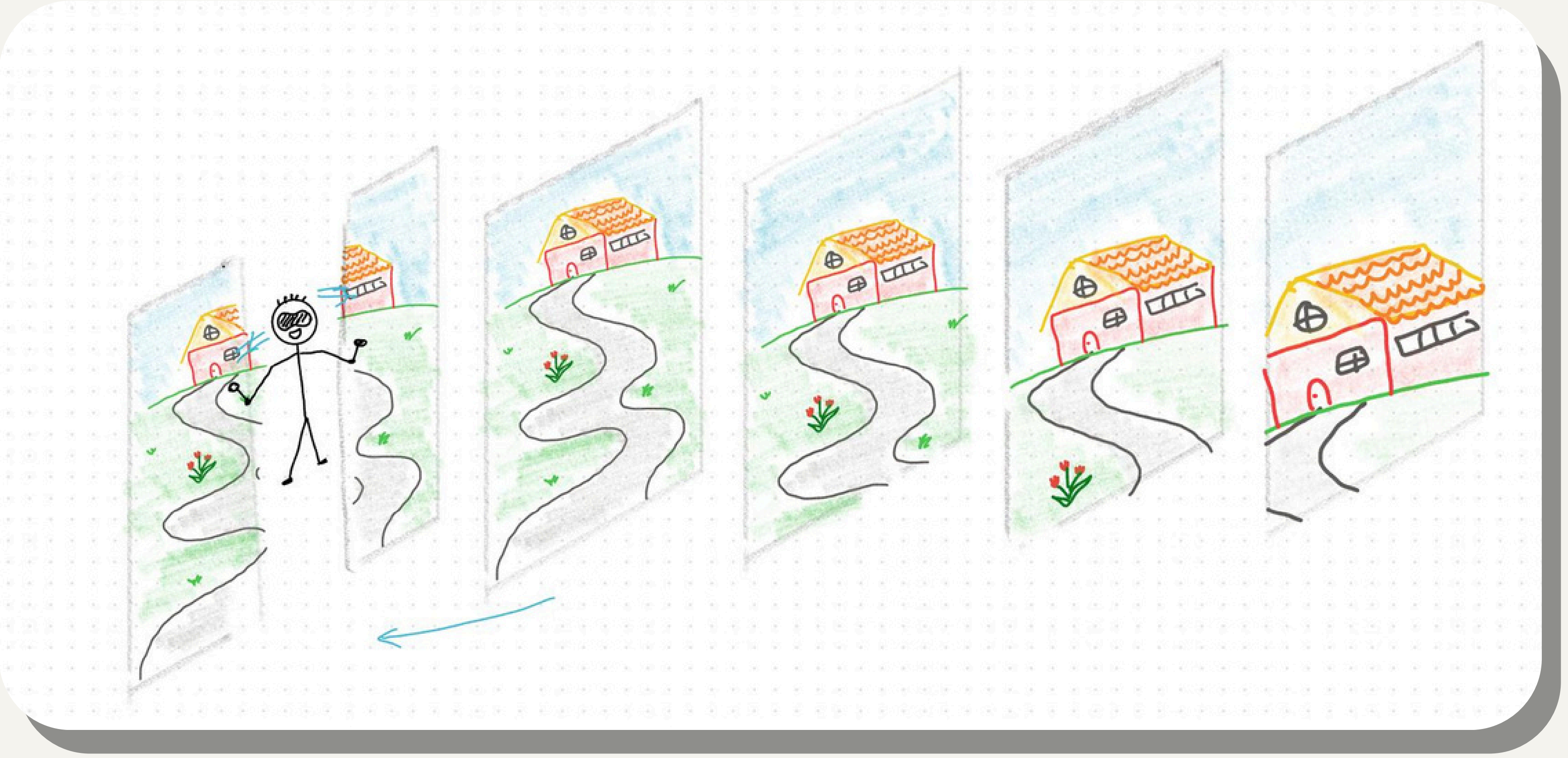
HCI FOR MIXED REALITY

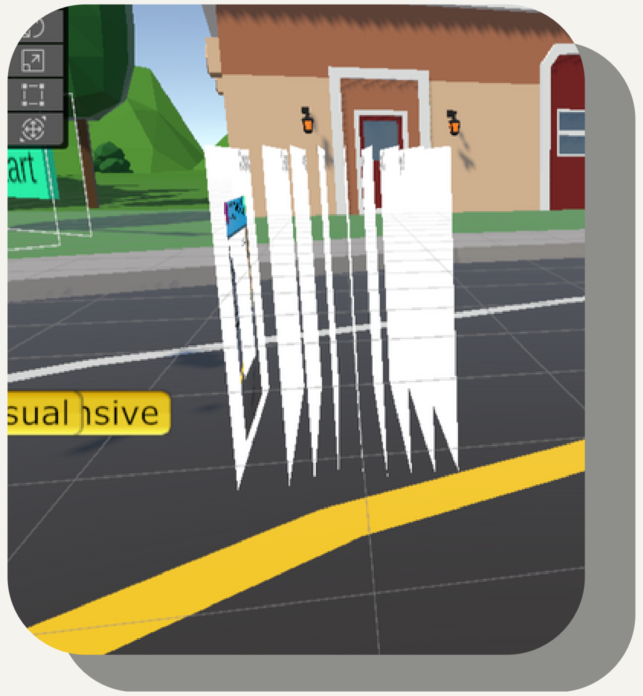
AMÉLIEN LE MEUR

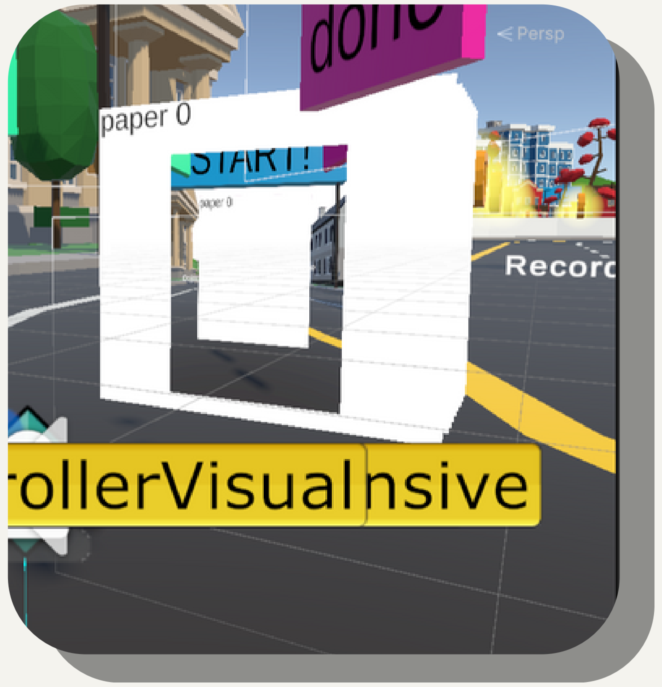
SUMMARY

- Concept
- Implementation, Challenges
- Evaluation
- Conclusion

PREVIOUS CONCEPT: FRAME SPLIT

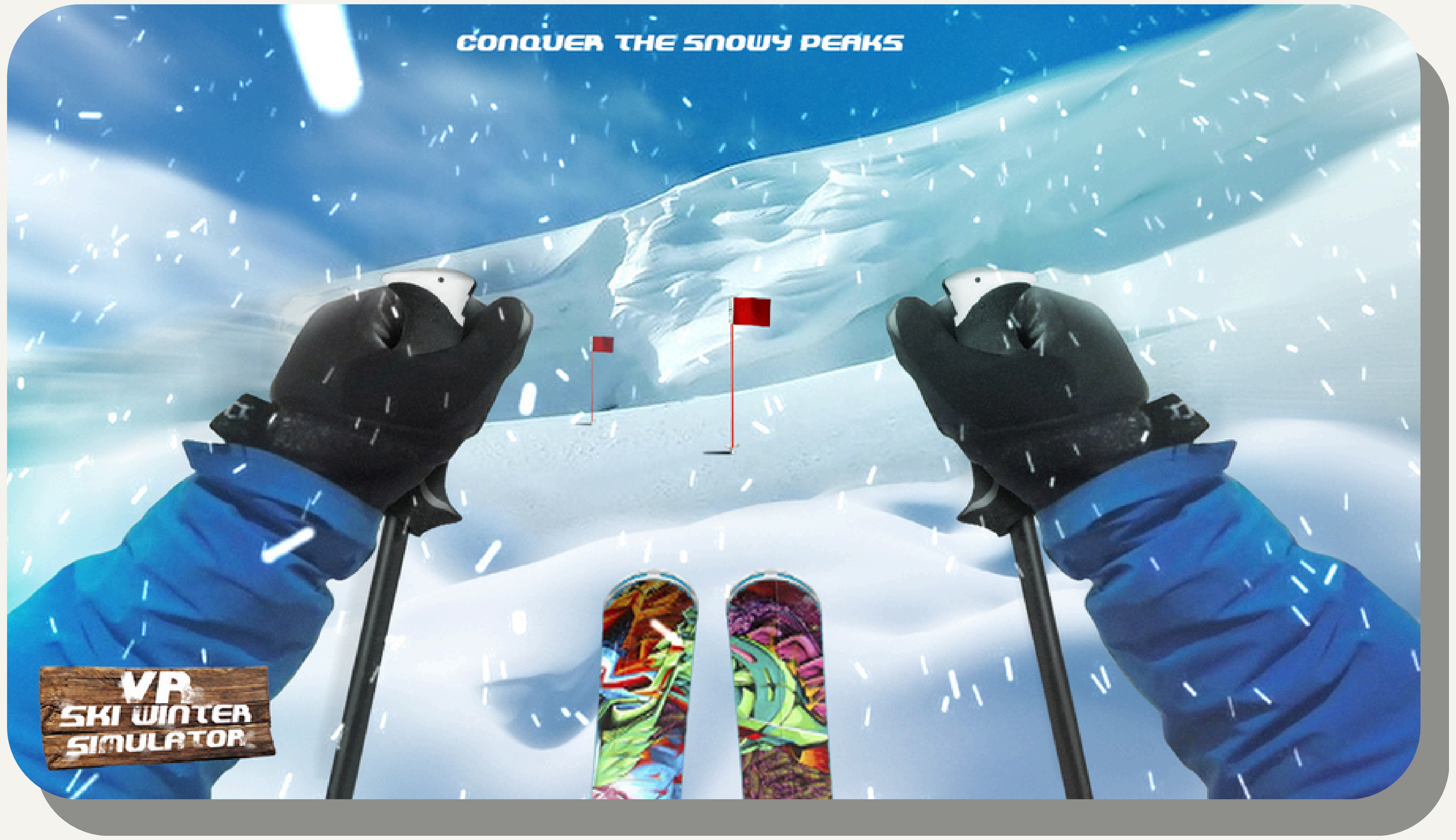






- Technical Instability from Recursive Rendering
- High Complexity, Low Interaction Payoff
- Mismatch with Embodied VR Strengths
- Feels like a video effect, not movement
- Limited Expressiveness for Navigation
- Motion Sickness
- already implemented

NEW CONCEPT: SKIING



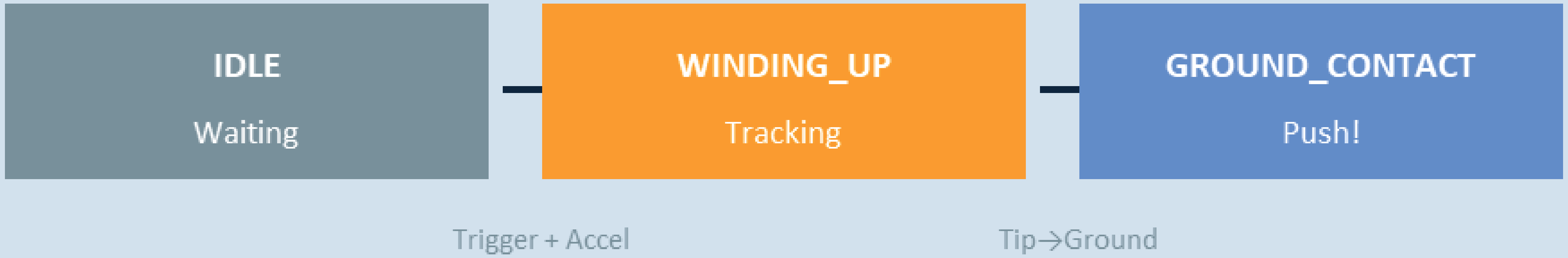
“VR Ski Simulator” - Aploft

→ User uses two poles to pull themselves forward

IMPLEMENTATION

Challenge 1: Push Detection

- How to detect a valid ski pole push?
- Simple trigger press too sensitive
- Velocity-only detection unreliable



IDLE

Waiting

WINDING_UP

Tracking

GROUND_CONTACT

Push!

Trigger + Accel

Tip→Ground

float gain = Mathf.Min(pushPower + pushImpulse, maxPushGain);
_velocity += headForward * gain;

Start tracking when:

- Acceleration > threshold
- Hand in FRONT of face
- Trigger held

Apply push when:

- Pole tip touches ground

Power calculation:

gain = pushPower (min speed)+ impulse

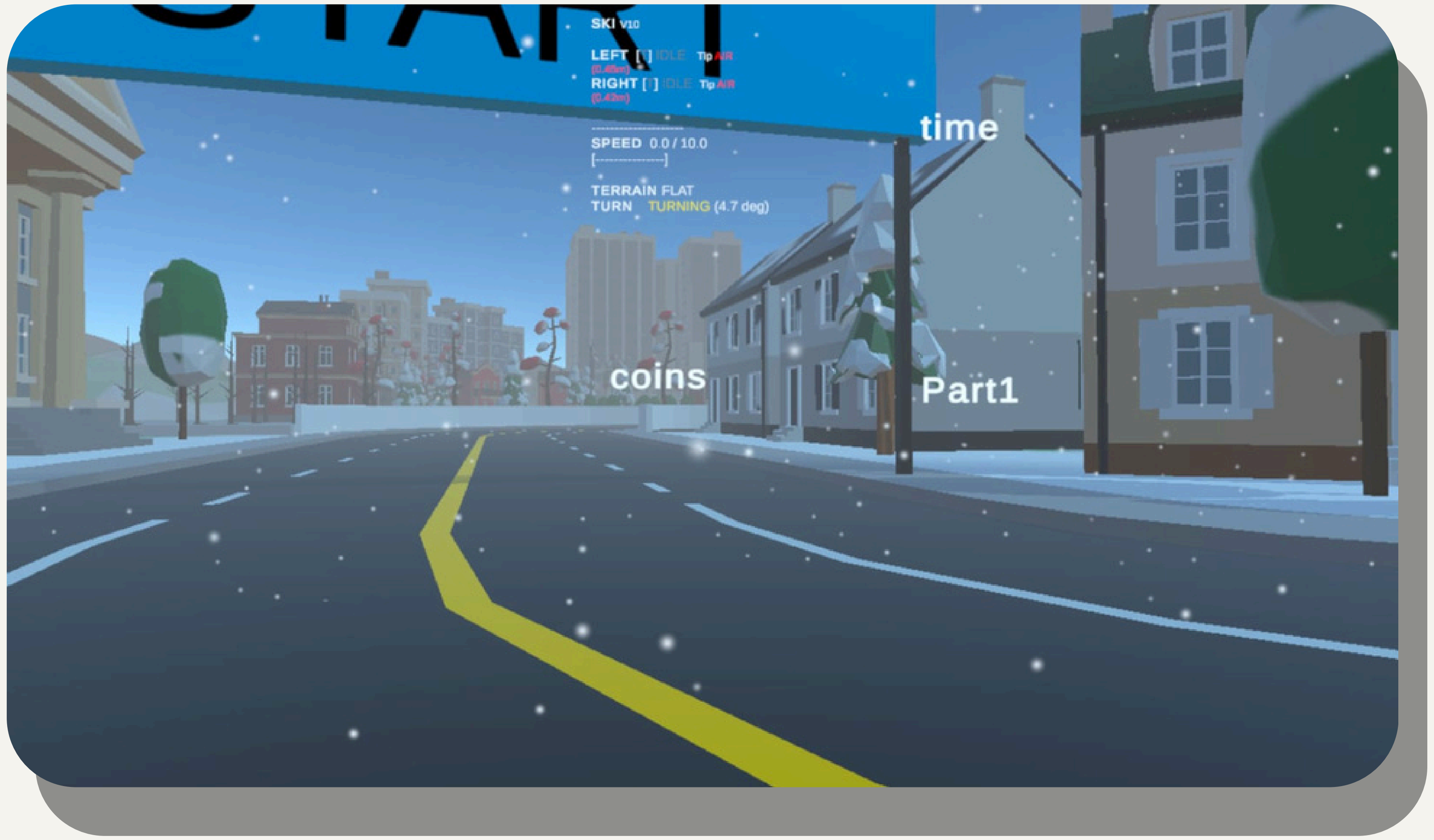
impulse accumulates from backward hand velocity

IMPLEMENTATION

Problem

Challenge 2: Tip ground detections

- Tips never detected as "grounded"
- VR tracking heights differ from expected
- Fixed distance checks failed



A pole is considered grounded when a downward raycast from its tip detects ground within a small height tolerance.

IMPLEMENTATION

Challenge 3: actually move where we want

•No hip tracker = can't detect body orientation

•Only HMD + 2 controllers tracked

•Needed intuitive steering mechanism

Attempted approach

•Tried pole tip positions relative to body

•Failed: User looks around while skiing!

Head direction ≠ body orientation

Solution:

- Direction = head forward (horizontal)
- Turn head → turn movement
- Velocity steers via RotateTowards()
- Turn responsiveness: 120°/sec

Turning reduces speed for stability, more controal, and realism

- Sharper turn → more speed loss

IMPLEMENTATION

Challenge 4: Gravity & Ground Following

Problem:

- Unity gravity caused constant sinking

We needed gravity for airborn condition

Solution:

- On ground: Snap Y to groundY
- In air: Apply custom gravity manually

Challenge 5: Hill & Slope Physics

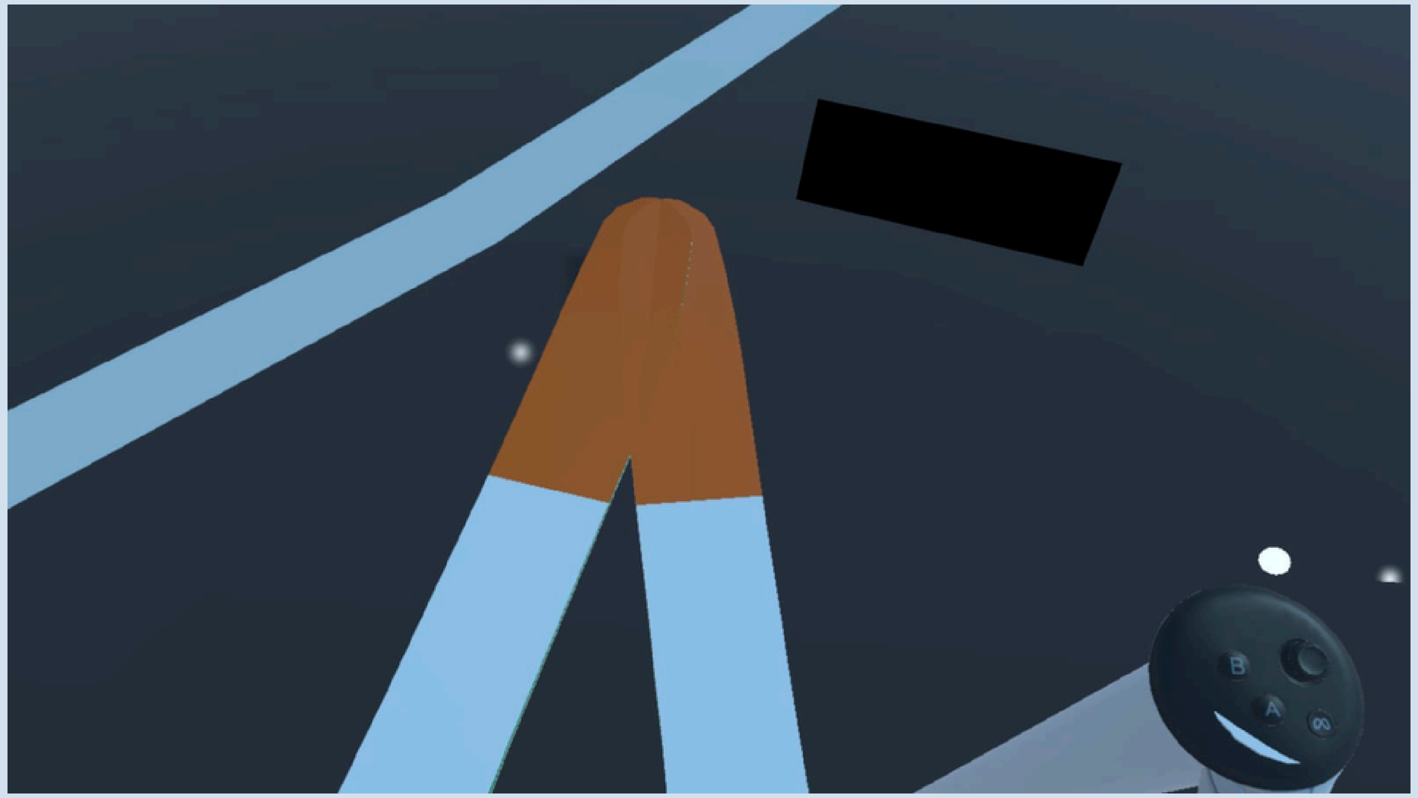
- Raycast returns ground normal
- slopeAngle = Angle(normal, up)

Project velocity onto slope plane

Terrain	Friction	Effect
Flat	Normal (0.3)	Base slowdown
Uphill	2.5× higher	Harder to climb
Downhill	0.3× + boost	Accelerate!

Airborne motion

- Ground detected via downward raycast
- Loss of ground → enter airborne state
- Horizontal velocity preserved in air
- Vertical motion controlled with custom gravity



Jumping over ramps

- Triggered at slope discontinuities (ramps)
- Requires minimum speed threshold
- Launch direction aligned with slope forward
- Smooth landing by snapping back to ground

Braking

- Braking mapped to controller grip input
- Grip increases effective friction
- Speed reduced smoothly over time
- Works consistently on flat and sloped terrain

USER TEST

Protocol: Make the user go around the track three times to give them the time to be acquainted to the locomotion technique

→ 3 participants

Main things to take away:

1min30s

Average time around the track

Motion Sickness

There was a bit of motion sickness towards the end but it was low

Easy/Fun to use

Useers thought it was pretty easy to use

THANK YOU FOR LISTENING !