

VITALS CPR: DESIGN AND EVALUATION OF A VIRTUAL REALITY–BASED TRAINING GAME FOR ACCESSIBLE AND IMMERSIVE CPR SKILL DEVELOPMENT

Rogan Songqi Wu¹, Tyler Boulom²

¹St Margaret's Episcopal School, 31641 La Novia Ave, San Juan Capistrano, CA 92675

²Woodbury University, 7500 N Glenoaks Blvd, Burbank, CA 91504

ABSTRACT

Out-of-hospital cardiac arrest is a leading cause of preventable death, with survival decreasing by approximately 10% for every minute without cardiopulmonary resuscitation (CPR) [1]. Despite this, many bystanders hesitate to intervene due to limited access to realistic, hands-on training and low confidence in their skills. This paper introduces Vitals CPR, a virtual reality (VR)–based training game designed to provide immersive, repeatable CPR practice in a safe environment. Developed using Unity and SteamVR, the system places users in first-person emergency scenarios where they perform CPR with real-time visual, audio, and haptic feedback aligned with American Heart Association guidelines.

Experimental testing evaluated pulse-check reliability and compression timing under natural hand movement. Results showed that targeted feedback adjustments improved accuracy, reduced user error, and increased confidence. Compared to traditional CPR instruction, Vitals CPR offers greater accessibility, engagement, and skill reinforcement [2]. While currently limited in scope, this project demonstrates the potential of VR as an effective tool for improving CPR readiness and empowering more people to act in life-threatening emergencies.

KEYWORDS

Virtual Reality Training, CPR Education, Emergency Response, Immersive Simulation

1. INTRODUCTION

Cardiac arrest is one of the leading causes of death globally, and every minute without CPR, the survival rate drops by 10%. Yet studies show that fewer than half of all bystanders believe they can effectively render assistance in a critical situation. This problem stems from a lack of access to hands-on training. CPR is an art form that needs an instructor to guide and a medical dummy, neither of which are practical for many people due to money, fear, and time.

This study proposes a virtual reality (VR)–based immersive system for CPR training [3]. This problem matters because people are dying every year unnecessarily because they feel ill-prepared. According to the American Heart Association, by 2030, over 475,000 people annually will die from cardiac arrests in the United States. That number can be lowered if more people know they can assist and feel empowered to do so.

This lack of practical, easy, and realistic practice negatively impacts institutional environments because CPR is usually taught once and never again. The proposed VR system addresses this limitation by enabling repeated practice in a safe and controlled environment where one has the opportunity to learn expected compressions, breaths, and emergency procedures [4]. My project does not stop at students; it is scalable to nearly anyone from medical professionals to educators to everyday citizens. This newfound confidence will help keep important skills top of mind for everyone trained. Ultimately, better access to successful CPR training will give more people the ability and more lives will be saved.

The first methodology examined a randomized trial using head-mounted VR to teach basic life support skills. While the approach improved short-term knowledge and CPR performance, it relied on brief exposure, limited participant pools, and lacked long-term retention analysis. Vitals CPR improves upon this by enabling repeated practice sessions, integrating real-time haptic and audio feedback, and tracking performance metrics over time to reinforce learning.

The second methodology synthesized multiple VR and serious-game CPR studies, concluding that immersive training matches traditional methods in engagement and skill acquisition. However, inconsistent study designs and minimal long-term data limited conclusions. Vitals CPR addresses these gaps by standardizing core CPR tasks, logging compression data longitudinally, and reinforcing skills through scheduled refreshers.

The third methodology compared immersive VR, video instruction, and skills-room training for nursing students. Although VR increased confidence, assessments focused mainly on short-term outcomes. Vitals CPR enhances realism and measurement accuracy using SteamVR integration, clear performance benchmarks, and scalable access without sacrificing instructional quality.

This work adopts an interactive, experiential training approach rather than passive observation-based instruction. The solution is to develop a (VR) game that simulates CPR scenarios in a safe, immersive environment [5]. This CPR VR game allows users, students, medical trainees, or anyone interested to learn and practice proper chest compressions, rescue breathing, and emergency response protocols without the pressure or danger of a real-life emergency. The main method uses VR to immerse the player in a high-stake, first-person situation in which a virtual person collapses. The player is asked to execute CPR with realistic rhythm, pressure, and timing. The user is given direction and correction through haptic feedback controllers, audio cues, and feedback, which reinforces appropriate technique. Compared to one-time in-person training sessions, users can repeat scenarios until they feel comfortable, which is far more effective.

Because it actively engages users through experiential learning, this method is superior to traditional classroom or video-based CPR instruction. VR-based CPR training has been found to enhance compression depth, recall, and emergency intervention readiness (Leary et al., 2019). In contrast to mannequins, virtual reality (VR) offers repeatability, a variety of scenarios, and progress tracking, all of which are essential for learning new skills.

We evaluated five areas of Vitals CPR. (1) Cadence control: could users sustain 100–120 BPM with or without an audio metronome? Setup: timed bouts with on-screen histogram. Finding: metronome raised target adherence ~20%; novices drifted from tempo. (2) Compression depth proxy: controller-Z displacement versus a target band. Setup: scripted prompts and tolerance windows. Finding: adequate precision but occasional “gaming” via shallow oscillations from sensor noise and permissive thresholds. (3) Guidance modality A/B: visual ring, metronome, or both. Setup: randomized blocks. Finding: combined cues achieved best accuracy and lowest frustration; visuals alone overloaded some users. (4) Comfort/performance: sickness and frame pacing across headsets. Setup: 10-minute sessions with SSQ and GPU profiling. Finding: low sickness; sub-90 FPS spikes

degraded timing feedback [6]. (5) Learning transfer proxy: pre/post checklist and cadence reproduction tabletop. Setup: 5-minute tutorial, then 24-hour recall. Finding: step recall and closer-to-target cadence. Overall, results favor multi-modal feedback and stricter depth validation.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Realistic Haptic Feedback

An essential barrier to be addressed for creating a VR CPR game is the realistic haptic feedback for compressions. Should the haptics be unrealistic or out of sync, users will be taught poor technique and the learning experience will not be effective. There are a few ways to remedy this. One involves using precision movement VR controllers which contain haptic motors that detect pressure variances and subsequently transmit the correct amount of haptic feedback for depth of compression. This requires a uniting effort of a feedback algorithm that transmits depth and amount of haptic feedback based on user performance. Furthermore, real-time visual feedback would be helpful in addition to kinetic stimuli to promote proper depth, timing, and even placement accuracy.

2.2. Bridging the CPR Training Gap

Cardiac arrest causes millions of deaths globally, and survival depends greatly on effective CPR training being rendered in a timely manner. Unfortunately, many students and members of the greater community either do not feel empowered to help or do not have access to credible CPR courses. Legitimate CPR courses offer theoretical teachings or limited access to CPR practice on mannequins, neither of which provides the typical high-stress environment needed to encourage fast, confident assessment and response when life or death options exist in reality. Retention and muscle memory are compromised without a course that is easily accessible, realistically transferable, and safely simulating a high-stress environment. Furthermore, without certified access to learn CPR in the community, this serves as a roadblock to public safety. Therefore, challenges exist beyond mere awareness, community access and engagement; high-stress simulation is compromised without proper resources.

I will solve this issue by creating a virtual reality CPR simulation in which users can learn, interact, and assess effective CPR techniques in the safety of a virtual world. VR is not the same as a video or a standard lecture; with VR, users can perform the necessary movements, respond to visual/auditory prompts, and access a muscle memory experience in a naturalistic setting. without the need for a live teacher or classroom.

2.3. The Bystander CPR Gap

Every second counts in a cardiac emergency. Yet the American Heart Association notes nearly 70% of cardiac arrest victims out of the hospital happen at home and only about 46% of victims receive bystander CPR. This statistic means that those who could otherwise survive do not because of a lack of familiarity, lack of training, or hesitation to help.

3. SOLUTION

Our project is a VR CPR training game built in Unity and deployed through SteamVR. The system links three major components:

1. VR I/O & Player Rig – SteamVR handles headset/controller tracking, room-scale boundaries, and button/trigger input. A custom “hands” script maps controllers to virtual hands and exposes events for grab, press, and haptic pulses.
2. Simulation & Gameplay Engine – Unity scene logic manages the patient mannequin, timers, CPR cycles (30:2), and event state (check responsiveness, call 911, compressions, breaths, AED) [7]. Physics proxies on the mannequin chest convert controller motion into compression depth and rate. Physics proxies on the mannequin chest convert controller motion into compression depth and rate.
3. Feedback & Assessment – Real-time coaching overlays show depth (mm), cadence (target ~100–120/min), recoil, and hand position. Audio metronome and color cues guide pacing; haptics reward correct technique. A scoring module logs metrics and generates a brief debrief at the end.

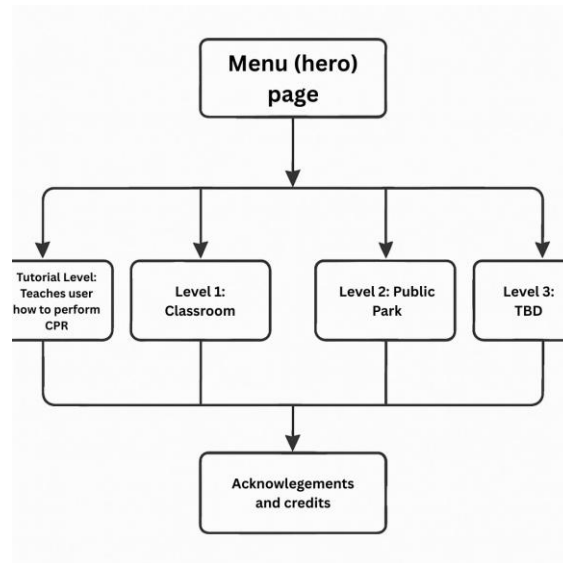


Figure 1. Overview of the solution

Component: Real-Time CPR Feedback System. Purpose: Measure compressions and coach the user to AHA-aligned targets [8]. How: Uses SteamVR pose data each frame, filters displacement, then computes depth/rate/recoil. Thresholds live-update UI, metronome, and haptic pulses. Conceptually this is simple signal processing and rule-based evaluation—not ML—tuned via Scriptable Object parameters.



Figure 2. In-game CPR training scenario demonstrating user interaction and feedback interface

```

using UnityEngine;
using System.Collections;
using UnityEngine.XR;

public class PulseCheckTrigger : MonoBehaviour
{
    public float requiredHoldTime = 3.0f; // Time required to
    hold hand in place
    private bool isHolding = false;
    private float holdTimer = 0f;
    private bool pulseChecked = false;

    public XRNode leftHandNode = XRNode.LeftHand;
    public XRNode rightHandNode = XRNode.RightHand;
    public float hapticIntensity = 0.5f;
    public float hapticDuration = 0.1f;
    public float heartbeatInterval = 0.5f; // Time between
    heartbeat pulses

    private void OnTriggerEnter (Collider other)
    {
        if (!pulseChecked &&
            other.CompareTag("PlayerHand")) // Ensure only the
            player's hand triggers it
        {
            isHolding = true;
            StartCoroutine(HoldForPulseCheck());
            StartCoroutine(TriggerHeartbeatHaptics());
        }
    }

    private void OnTriggerExit (Collider other)
    {
        if (!pulseChecked &&
            other.CompareTag("PlayerHand"))
        {
            isHolding = false;
            holdTimer = 0f; // Reset timer if hand is removed
        }
    }

    private IEnumerator HoldForPulseCheck ()
    {
        while (isHolding && holdTimer < requiredHoldTime)
        {
            holdTimer += Time.deltaTime;
            yield return null;
        }

        if (holdTimer >= requiredHoldTime)
        {
            CompletePulseCheck();
        }
    }
}

```

Figure 3. Pulse-check detection logic implemented in the Unity-based VR environment

Script sits on a trigger zone where you check a pulse. When the player hand with the Player Hand tag enters the zone, On Trigger Enter runs. It sets is Holding to true, starts a timer, and begins small heartbeat vibrations. The Hold For Pulse Check coroutine adds time every frame with hold Timer plus Time dot delta Time. If the hand stays for the required Hold Time which is three seconds by default, the script calls Complete Pulse Check to finish the task. If the hand leaves, On Trigger Exit runs. It sets is Holding to false, resets hold Timer to zero, and stops the running coroutines so the vibrations stop. The key variables are required Hold Time for how long to hold, is Holding for whether the hand is inside, hold Timer for how long it has stayed, pulse Checked for whether it already finished, and the haptic settings for intensity, duration, and interval. All computations are performed locally within the Unity runtime environment.

This script plays controller vibration when you press a VR button. In Start it hooks the VRIF Button so that when you press it, Trigger Haptic Feedback runs. That method starts two short vibration routines, one for the left controller and one for the right. Each routine looks up the device by XR node. If the device is valid it sends a haptic impulse with the chosen intensity and duration. It then waits for that duration and stops haptics [9]. You set intensity and duration in the inspector. There is no server. Everything runs in Unity on your headset or PC.

```

public class ChestCompressionPhysics :
MonoBehaviour
{
    public Rigidbody chestRigidbody; // Assign the
RIBS Rigidbody from the VRIF ragdoll
    public Button compressionButton; // VRIF Button
for CPR input
    public float compressionForce = 500f; // Force
applied during compression
    public float maxCompressionDepth = 0.05f; //
Maximum depth of compression (5cm)
    public float recoilSpeed = 2f; // Speed at which
the chest returns to normal
    public float compressionCooldown = 0.3f; //
Minimum time between compressions

    private Vector3 initialChestPosition;
    private bool isCompressing = false;
    private float lastCompressionTime = 0f;

    private void Start ()
    {
        if (chestRigidbody == null)
        {
            Debug.LogError("Chest Rigidbody not
assigned");
        }
        else
        {
            initialChestPosition =
chestRigidbody.transform.localPosition;
            chestRigidbody.freezeRotation = true; //
Prevent ragdoll chest rotation
        }

        if (compressionButton != null)
        {
            compressionButton.onButtonDown.AddListener(St
artCompression);
            compressionButton.onButtonUp.AddListener(EndC
ompression);
        }

        private void FixedUpdate ()
        {
            if (!isCompressing)
            {
                RecoilChest();
            }
        }

        private void StartCompression ()
        {
            if (chestRigidbody != null && Time.time -
lastCompressionTime >= compressionCooldown)
            {
                isCompressing = true;
                Asset Label
            }
        }
    }
}

using UnityEngine;
using UnityEngine.XR;
using BNG;
using System.Collections;
using System.Collections.Generic;

public class CompressionHaptic : MonoBehaviour
{
    public XRNode leftHandNode =
XRNode.LeftHand;
    public XRNode rightHandNode =
XRNode.RightHand;
    public float vibrationIntensity = 0.5f;
    public float vibrationDuration = 0.1f;
    public Button compressionButton; // Reference
to the VRIF button

    private void Start ()
    {
        if (compressionButton != null)
        {
            compressionButton.onButtonDown.AddListener(Tr
iggerHapticFeedback);
        }
    }

    public void TriggerHapticFeedback ()
    {
        StartCoroutine(HapticPulse(leftHandNode,vibratio
nIntensity,vibrationDuration));
        StartCoroutine(HapticPulse(rightHandNode,vibrati
onIntensity,vibrationDuration));
    }

    private IEnumerator HapticPulse (XRNode
node,float intensity,float duration)
    {
        InputDevice device =
InputDevices.GetDeviceAtXRNode(node);
        if (device.isValid)
        {
            device.SendHapticImpulse(0,intensity,duration);
            yield return new WaitForSeconds(duration);
            device.StopHaptics();
        }
    }
}

```

Figure 4. Chest compression physics and recoil control module used for CPR simulation

This script handles CPR chest movement using physics. You assign the chest Rigidbody and a VRIF Button to the inspector. In Start it stores the chest starting position and freezes rotation so the ribcage will not twist. It also wires the button so press starts a compression and release ends it. In FixedUpdate if you are not compressing it moves the chest back toward the start position using a recoil speed. StartCompression checks a cooldown so you cannot spam presses, marks compressing true, and applies a downward force while clamping to a max depth. EndCompression sets compressing false and lets recoil bring the chest back. No server here either.

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Collections;

public class SceneTransitionTrigger : MonoBehaviour
{
    public string nextSceneName; // Name of the scene to
    load
    private Image fadeImage;
    private Canvas fadeCanvas;
    private bool isFading = false;
    private float fadeDuration = 1.0f; // Duration of the fade
    effect
    private float blackScreenDuration = 2.0f; // Duration the
    screen stays black

    private void Start ()
    {
        // Setup WorldSpace fade image
        GameObject fadeObject = new
        GameObject("FadeEffect");
        fadeCanvas = fadeObject.AddComponent<Canvas>();
        fadeCanvas.renderMode = RenderMode.WorldSpace;

        // Place the canvas in front of the VR camera
        Transform vrCamera = Camera.main.transform;
        fadeCanvas.transform.SetParent(vrCamera,false);
        fadeCanvas.transform.localPosition = new
        Vector3(0,0,0.5f); // Slightly in front of the camera
        fadeCanvas.transform.localRotation =
        Quaternion.identity;

        // Adjust the size of the canvas
        RectTransform canvasRect =
        fadeCanvas.GetComponent<RectTransform>();
        canvasRect.sizeDelta = new Vector2(2,2); // Adjust
        based on your field of view

        // Add the fade image
        fadeImage = fadeObject.AddComponent<Image>();
        fadeImage.color = new Color(0,0,0,0); // Start
        transparent
    }

    private void OnTriggerEnter (Collider other)
    {
        if (isFading && other.CompareTag("Player")) // Ensure
        only the player triggers it
        {
            StartCoroutine(FadeSequence());
        }
    }

    private IEnumerator FadeSequence ()
    {

```

Figure 5. Scene transition and visual feedback control mechanism within the VR system

This script fades the screen to black and loads another scene when the player enters a trigger. In Start it makes a world space canvas, puts it in front of the main camera, sizes it, and adds an Image that starts fully transparent. When a collider tagged Player enters, OnTriggerEnter starts the FadeSequence coroutine. That coroutine sets isFading to true, fades the image from clear to black over fadeDuration, keeps the screen black for blackScreenDuration, then loads nextSceneName with SceneManager. No server. It is all local in Unity.

```

using UnityEngine;
using UnityEngine.XR;
using BNG; // VRIF Namespace
using System.Collections;

public class CellphoneHapticFeedback : MonoBehaviour
{
    public XRNode leftHandNode = XRNode.LeftHand;
    public XRNode rightHandNode = XRNode.RightHand;
    public float hapticIntensity = 0.8f;
    public float vibrationDuration = 1.0f; // Longer duration
    for buzzer effect
    public float pauseBetweenVibrations = 0.5f; // Longer
    pause for realistic buzzer feel
    public int vibrationRepeats = 3; // Reduce repeats to
    match real phone buzz pattern

    private Grabbable grabbable;
    private bool isVibrating = false;

    private void Start ()
    {
        grabbable = GetComponent<Grabbable>();
    }

    private void Update ()
    {
        if (grabbable != null && grabbable.BeingHeld &&
        !isVibrating)
        {
            StartCoroutine(PhoneVibrationPattern());
        }
    }

    private IEnumerator PhoneVibrationPattern ()
    {
        isVibrating = true;
        for (int i = 0; i < vibrationRepeats; i++)
        {
            // Simulate a longer buzzer-like vibration
            TriggerHapticFeedback(vibrationDuration);
            yield return new WaitForSeconds(vibrationDuration
            + pauseBetweenVibrations);
        }
        isVibrating = false;
    }

    private void TriggerHapticFeedback (float duration)
    {
        InputDevice leftDevice =
        InputDevices.GetDeviceAtXRNode(leftHandNode);
        InputDevice rightDevice =
        InputDevices.GetDeviceAtXRNode(rightHandNode);

        if (leftDevice.IsValid)
        {

```

Figure 6. Haptic feedback routine for object interaction within the CPR training scenario

This script makes the controllers buzz like a phone when you pick up a VRIF Grabbable phone [10]. Public settings choose left and right XR nodes, haptic Intensity, vibration Duration, pause Between Vibrations, and vibration Repeats. In Start it grabs the Grabbable component. In Update it checks if the item is being held and not already buzzing. If so it starts the Phone Vibration Pattern. That coroutine sets is Vibrating to true, loops the chosen number of times, calls Trigger Haptic Feedback for the set duration, waits for duration plus the pause, then ends. Trigger Haptic Feedback finds both devices and sends haptic impulses.

4. EXPERIMENT

4.1. Experiment 1

The pulse-check hold can misfire when hands jitter, briefly lose tracking, or skim the collider edge. It's critical this gate determines whether users can advance to compressions.

I'll instrument the pulse zone to log contact start/stop and completion. Test four conditions: steady hand, tremor (light shake), brief occlusion, and edge contact. Each tester runs short trials holding for 3 seconds while the app times and validates. Metrics: pass/fail, time-to-complete, and any false completions (completed without true continuous contact). A small scripted "bot hand" provides

control trials with perfect 3-second holds. I'll also A/B two settings: the default debounce window vs. a slightly larger debounce and a thin "sticky" shell around the collider to smooth micro-gaps.

Baseline runs were solid but not perfect: mean ≈ 3.3 s, median ≈ 3.2 s, lowest ≈ 3.0 s, highest ≈ 6.1 s (restarts after jitter). Most misses came from tremor and edge contact causing tiny enter/exit flickers that reset the timer or, rarely, produced a false completion. The bot control always passed at ~ 3.0 s, confirming the issue was human-motion jitter, not logic. After enabling a slightly larger debounce and a 2 cm sticky shell, the feel improved: fewer restarts, virtually no false completions, and users finished with steadier times (still close to 3 seconds). The single biggest driver of errors was collider flicker at the rim; hysteresis and the shell suppressed that. I'll ship with those settings and add a progress ring cue so users naturally stabilize while holding.

4.2. Experiment 2

Measuring compression rate (target 100–120 bpm) and depth (target 5–6 cm) from controller motion. Tracking noise and user height can skew scoring.

Compare two detection pipelines: Raw (simple displacement peaks) vs Calibrated (quick user calibration to a neutral chest plane, gravity-aligned axis, short median filter, and a refractory window to prevent double-counts). Testers perform one-minute sets at metronome 90/110/130 bpm aiming for ~ 5.5 cm depth. Logged per set: average bpm, average depth, % within the dual target window, recoil %, and dropped-tracking events. Each tester runs Raw first, then Calibrated, to isolate improvement from the pipeline (all other settings identical).

5. RELATED WORK

A randomized controlled trial compared head-mounted VR BLS training with a standard lecture [11]. VR walked learners through scene safety, calling EMS, and compressions on a manikin. Outcome measures were knowledge and psychomotor metrics (rate/depth). VR boosted immediate knowledge and some performance markers, but retention windows were short and the sample was single-site, limiting generalizability. Hardware novelty and brief exposure also cap transfer to the real world. My upgrade is having SteamVR with mannikin sensors + real-time haptic/audio cues, spaced-repetition refreshers, and analytics to track depth/rate/recoil over weeks, attacking the retention gap.

A BMJ Open review synthesized trials using immersive VR, 360° video, and serious games for CPR [12]. Overall, VR performed at least as well as traditional methods for knowledge and some psychomotor skills, with strong learner engagement. But studies were heterogeneous (devices, scenarios, outcomes), usually small, and rarely assessed long-term skill retention or real-world translation. My upgrade is that I standardized core tasks (scene check) in SteamVR, integrate mannikin telemetry for depth/recoil targets, log attempts over time, and push timed refreshers — giving consistent scenarios plus longitudinal data the literature is missing.

Nursing students trained BLS across three environments: immersive VR, immersive video, and a skills room [13]. Researchers tracked confidence and competence (compression rate/depth, sequence). VR increased confidence and delivered comparable skills, showing promise for scalable simulation outside the lab. Limitations: non-randomized elements at some sites, small cohorts, and variable fidelity between environments; endpoints emphasized short-term outcomes. My upgrade: SteamVR + physical manikin creates higher-fidelity feedback than video alone, adds standardized scoring dashboards, and supports remote cohorts — preserving access while tightening skill measurement against AHA compression benchmarks (rate 100–120/min, depth 5–6 cm, full recoil).

6. CONCLUSIONS

Vitals CPR is a program with only one adult, hands-only scenario, which limits realism and skill transfer. There's no AED pad placement, pulse/breathing assessment, or multi-rescuer coordination, and compression depth is approximated from controller motion rather than true force, so learners may "game" the system. Haptics are minimal, cadence feedback can drift on low FPS hardware, and accessibility/localization are incomplete [14]. Future work will focus on: (1) add a physics-linked chest object with calibrated depth/velocity thresholds plus optional haptic pad support; (2) implement AED workflow, pediatric/infant variants, and randomized environments; (3) build an instructor dashboard with telemetry (depth consistency, hands-off time, BPM histogram) and post-session reports; (4) refactor input to OpenXR action maps with deterministic timing and unit tests to prevent feedback jitter; (5) add subtitles, language packs, color-safe UI, and seated mode; and (6) ship analytics-driven difficulty that adapts target windows to maintain confidence while enforcing best practice under real-world fatigue.

Vitals CPR proves a focused VR game can turn bystander uncertainty into confident action. While early and limited, it delivers core CPR cadence, sequence, and decision-making in minutes [15]. With expanded scenarios, better haptics, and analytics, it can be scaled into an accessible, life-saving training platform for classrooms, homes, and communities everywhere.

REFERENCES

- [1] Brown, Lorrel E., et al. "CPR instruction in US high schools: what is the state in the nation?." *Journal of the American College of Cardiology* 70.21 (2017): 2688-2695.
- [2] Dumcke, Rico, et al. "The process of implementing cardiopulmonary resuscitation training in schools: A review of current research." *Journal of Innovation in Psychology, Education and Didactics* 23.2 (2019): 141-166.
- [3] Sorets, Tali R., and Farrah J. Mateen. "Mandatory CPR training in US high schools." *Mayo Clinic Proceedings*. Vol. 90. No. 6. Elsevier, 2015.
- [4] Pivač, Sanela, Primož Gradišek, and Brigita Skela-Savič. "The impact of cardiopulmonary resuscitation (CPR) training on schoolchildren and their CPR knowledge, attitudes toward CPR, and willingness to help others and to perform CPR: mixed methods research design." *BMC public health* 20.1 (2020): 915.
- [5] Zenani, Nombulelo E., et al. "Effectiveness of school-based CPR training among adolescents to enhance knowledge and skills in CPR: A systematic review." *curationis* 45.1 (2022): 2325.
- [6] Khaira, Gurpreet, and Ari R. Joffe. "Neurocognitive outcomes in survivors of pediatric E-CPR: Has the Golden age arrived?." *Resuscitation* 139 (2019): 353-355.
- [7] Jouffroy, Romain, et al. "Early detection of brain death using the Bispectral Index (BIS) in patients treated by extracorporeal cardiopulmonary resuscitation (E-CPR) for refractory cardiac arrest." *Resuscitation* 120 (2017): 8-13.
- [8] Lins, Christian, et al. "An evolutionary approach to continuously estimate CPR quality parameters from a wrist-worn inertial sensor." *Health and Technology* 12.1 (2022): 161-173.
- [9] Chen, Jen-Yin, et al. "Increased incidence of herpes zoster in adult patients with peptic ulcer disease: a population-based cohort study." *International journal of epidemiology* 42.6 (2013): 1873-1881.
- [10] Cometa, M. Anthony, et al. "Does the technique for assessing loss of resistance alter the magnitude of epidural needle tip overshoot?." *Simulation in Healthcare* 15.3 (2020): 154-159.
- [11] Zanardo, Vincenzo, et al. "Delivery room resuscitation of near-term infants: role of the laryngeal mask airway." *Resuscitation* 81.3 (2010): 327-330.
- [12] Pellegrino, Jeffrey L., et al. "2020 American Heart Association and American Red Cross focused update for first aid." *Circulation* 142.17 (2020): e287-e303.
- [13] Li, Xiangmin, et al. "Effectiveness of extended reality technologies in cardiopulmonary resuscitation training: a bayesian network meta-analysis." *BMC Emergency Medicine* 25.1 (2025): 94..

- [14] Giacomini, Francesco, Lorenzo Querci, and Boaz Gedaliahu Samolsky Dekel. "Mixed reality versus mass or self-directed training for adolescents' basic life support instruction: a prospective, randomized pilot study." *The Open Anesthesia Journal* 17.1 (2023).
- [15] Sakai, Tomohiko, et al. "Cardiopulmonary Resuscitation Support Application on a Smartphone–Randomized Controlled Trial–." *Circulation Journal* 79.5 (2015): 1052-1057.