Midterm Report: BuzzWrite – learn to write, or get the buzz!

Team Members: Kareem Elsenosy (khe2011), Adam Elghayaty (ae3010), Yehia Elsisi (yme2024)

1.0 Introduction

1.1 The Importance of Handwriting in the Digital Age

Handwriting remains a fundamental skill for cognitive development, fine motor control, and personal expression. It is crucial for children learning to formulate thoughts and for individuals recovering from neurological conditions like strokes, where relearning motor skills is a key part of rehabilitation.

1.2 Challenges in Handwriting Acquisition and Rehabilitation

Traditional methods for teaching and rehabilitating handwriting often lack engagement, leading to frustration and slow progress. For individuals with motor impairments or those in post-stroke recovery, these methods can be physically demanding and require constant one-on-one supervision from therapists, which is often expensive and not always accessible.

1.3 The Potential of Virtual Reality as a Solution

Virtual Reality (VR) offers a unique and immersive platform to address these challenges. By creating a controlled, engaging, and interactive digital environment, we can transform handwriting practice from a chore into a motivating game. VR allows for precise hand-tracking, real-time feedback, and adaptive difficulty, which are ideal for skill development.

1.4 Project Purpose and Scope

The purpose of the "BuzzWrite" project is to develop an accessible and engaging VR platform for handwriting learning and rehabilitation. This project will utilize the Meta Quest 2 headset to deliver a gamified experience. The scope of this term project is to develop a Minimum Viable Product (MVP) that includes core features such as guided stroke learning, adaptive feedback, and a structured progression system.

2.0 Problem Definition and Objectives

2.1 Problem Statement

Handwriting is a vital skill that many children and individuals with disabilities struggle to learn or relearn, especially those with motor impairments or post-stroke conditions. Traditional rehabilitation is often costly, requires constant therapist supervision, and can lack engagement, leading to limited progress and motivation. There is a need for an accessible, adaptive, and engaging solution that makes handwriting training effective and fun.

2.2 Core Project Objectives (MVP)

1. **Guided Learning System:** Implement a "ghost hand" or laser guide that demonstrates the correct stroke-by-stroke motion for writing English letters.
2. **Adaptive Feedback Mechanism:** Provide real-time visual, audio, and haptic (vibration) feedback to the user when their hand movement deviates from the correct path.
3. **Gamified Progression:** Create a structured system with different levels, starting with simple strokes and progressing to full letters, to keep users engaged.
4. **Performance Metrics:** Calculate and display the user's accuracy and completion time after each letter to track progress.

3.0 Implementation Details

3.1 Development Environment & Technology

- **Hardware: Meta Quest 2:** The Meta Quest 2 is our target platform due to its standalone nature, affordability, and robust hand-tracking capabilities, which are essential for a controller-free handwriting experience.
- **Game Engine: Justification for Wonderland Engine vs. Unity:**
  While Unity is the industry standard for VR development with extensive documentation and a large asset store[1][2], we have chosen **Wonderland Engine** for our MVP. This decision is based on several key factors:
    - **Accessibility and Deployment:** Wonderland is optimized for WebXR, allowing our application to run directly in the Meta Quest browser without needing an app store download or installation[3][4][5]. This makes it incredibly easy for users to access our platform via a simple URL, which is ideal for a learning tool.
    - **Rapid Prototyping:** Wonderland's streamlined workflow and use of JavaScript/TypeScript allow for faster iteration on our core mechanics compared to the more complex project setup in Unity[6][7].
    - **Performance:** Wonderland is specifically designed to achieve high performance for VR applications within a web browser, which is sufficient for the graphical demands of our MVP[4][6].
- We acknowledge that for future expansions, especially our stretch goals involving complex multiplayer functionality and native performance optimizations, a transition to Unity with its powerful Meta XR SDKs for haptics and interaction would be a logical next step[8][9][10]. However, for the scope of this term's MVP, Wonderland provides the most direct and efficient path to a functional and accessible product.

3.2 Required Assets

- **3D Models:**
    - Stylized 3D Pen/Laser Pointer model for the user's hand.
    - 3D models/meshes for all 26 uppercase English letters.
    - A virtual "whiteboard" or writing surface object.
    - Simple models for the environment (e.g., a desk, a calm room setting).

- **User Interface (UI) Assets:**
    - A main menu panel with buttons for "Start," "Select Letter," and "Quit."
    - A letter selection grid.
    - An end-of-level display to show accuracy and time scores.
    - Visual cues (e.g., a green checkmark for success, a red 'X' for going off-path).
- **Audio Assets:**
    - Positive sound effect for correctly tracing a segment.
    - Negative/alert sound effect for going off-path.
    - Sound effect for letter completion.
    - Calm, ambient background music to aid concentration.
    - Voice prompts (e.g., "Stay on the line," "Great job!").

3.3 User Interaction Flow

The user begins in a simple, calm virtual room. A menu appears in front of them, allowing them to select a letter to practice. Once a letter is chosen, it appears large on a virtual whiteboard. A "ghost hand" holding a pen or a laser pointer appears, demonstrating the first stroke. The user then uses their own hand to trace the path. A laser from their finger follows their movement. If they deviate from the letter's path, the controller vibrates, a sound plays, and the laser turns red. Once they return to the path, the feedback stops. After completing all strokes of the letter, a results screen displays their accuracy and time, and they are prompted to choose another letter or quit.

3.4 Code and Technical Implementation

3.4.1 Boundary Detection Logic (Pseudocode)
This logic determines if the user's pointer is on the correct path.

code Code
downloadcontent_copy
expand_less

```
  // Executed every frame during the writing task

FUNCTION onUpdate():
   // Get the position of the user's laser pointer in 3D space
   userPointerPosition = getPointerPosition()

   // Get the 3D mesh object of the target letter
   letterObject = getLetterMesh()

   // Cast a ray from the user's pointer towards the letter
   isHittingLetter = raycast(from: userPointerPosition, towards: letterObject)

   IF isHittingLetter == TRUE:
      // User is on the path
      setVibration(intensity: 0)
```

```
        setPointerColor(color: Green)
        isOutsideBoundary = FALSE
        // inspect for letter checkpoints and trigger letter completion logic

    ELSE:
        // User is off the path
        setVibration(intensity: 0.8)
        setPointerColor(color: Red)
        isOutsideBoundary = TRUE
    END IF
END FUNCTION
```

3.4.2 Letter Completion Logic (Pseudocode)

This logic uses a series of invisible checkpoint colliders along the letter's path to track progress.

```
code Code
downloadcontent_copy
expand_less
  // Setup when a letter is loaded
FUNCTION setupLetter(letter):
    checkpoints = getCheckpointsForLetter(letter)
    FOR each checkpoint in checkpoints:
        checkpoint.isTriggered = FALSE
    END FOR
END FUNCTION

// Called when the user's pointer collides with a checkpoint
FUNCTION onCheckpointTriggered(checkpoint):
    checkpoint.isTriggered = TRUE
    playSound(sound: PositiveChime)

    // Check if all checkpoints have been hit
    allTriggered = TRUE
    FOR each cp in checkpoints:
        IF cp.isTriggered == FALSE:
            allTriggered = FALSE
            BREAK
        END IF
    END FOR

    IF allTriggered == TRUE:
        // Letter is complete
        endLevel()
    END IF
```

```
END FUNCTION
```

3.4.3 Accuracy Scoring Algorithm (Pseudocode)

This logic calculates the final accuracy score.

```
code Code
downloadcontent_copy
expand_less
  // Variables to track during the level
timeInsideLetter = 0.0
timeOutsideLetter = 0.0
totalTime = 0.0
T_expected = 10.0 // Example: Expected time in seconds for this letter

FUNCTION onUpdate(deltaTime):
   totalTime += deltaTime
   IF isOutsideBoundary == TRUE:
      timeOutsideLetter += deltaTime
   ELSE:
      timeInsideLetter += deltaTime
   END IF
END FUNCTION

FUNCTION calculateAccuracy():
   // Calculate speed score to penalize taking too long
   speedScoreRatio = T_expected / totalTime
   speedScore = min(1.0, speedScoreRatio)

   // Calculate path accuracy
   pathAccuracyRatio = timeInsideLetter / totalTime

   // Final accuracy is a product of path accuracy and speed
   finalAccuracy = pathAccuracyRatio * speedScore * 100

   RETURN finalAccuracy
END FUNCTION
```

3.5 The Virtual Environment

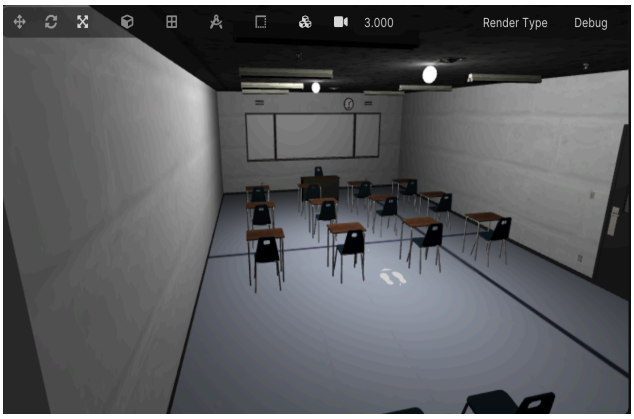
- **Background and Scene:** The environment will be a simple, non-distracting virtual space, such as a minimalist classroom or a calm, abstract void. This is to ensure the user can focus entirely on the handwriting task.

- **Audio Feedback:** As listed in the assets, audio will be a primary form of feedback. Positive chimes will reinforce correct movements, while a subtle alert sound will signal when the user has gone off-track. Voice prompts will provide clear, encouraging instructions.
- **Physics and Colliders:** The core interaction will rely on physics. The letter meshes will have colliders to detect the user's pointer. The checkpoint system for letter completion will be implemented using a series of small, invisible trigger colliders placed along the path of each letter's strokes.

3.6 Project Snapshots

(This section will be populated with actual screenshots from the Wonderland Engine editor once development progresses.)

- **Placeholder for Snapshot 1:** UI in the virtual environment classroom



- **Placeholder for Snapshot 2:** An in-engine view of a 3D letter model on the virtual whiteboard.

- **Placeholder for Snapshot 3:** Main Menu to start the program, select level or quit.

4.0 Progress to Date and Remaining Work

4.1 Completed Tasks

- Completed initial project proposal and problem definition.
- Conducted a comparative analysis of Wonderland Engine and Unity to select the appropriate development platform.
- Defined the core MVP features and user interaction flow.
- Developed detailed pseudocode for the primary game mechanics (boundary detection, completion, and scoring).
- Outlined a complete list of required 3D, UI, and audio assets.

4.2 Scope Adjustments

Based on the initial proposal, we have confirmed that the "Stretch Goals" (Multiplayer, progress tracking dashboards, cloud support) are too ambitious for the midterm and will be designated as future expansion plans. The core focus will remain entirely on delivering a polished single-player MVP.

4.3 Remaining Tasks for MVP Completion

- **[ ] Asset Creation/Import:** Create or source all 3D models, UI elements, and audio files.
- **[ ] Project Setup:** Initialize the project in Wonderland Engine and configure it for the Meta Quest 2.
- **[ ] Scene and UI Implementation:** Build the main menu and letter selection scenes.
- **[ ] Core Mechanic Programming:** Implement the boundary detection, completion, and scoring logic in JavaScript/TypeScript based on the pseudocode.
- **[ ] Level Creation:** Set up each of the 26 letters as a distinct level with its own checkpoints and expected completion time.
- **[ ] Feedback Integration:** Connect the audio and haptic feedback to the boundary detection logic.
- **[ ] Testing and Debugging:** Thoroughly test the application on the Meta Quest 2 to identify and fix bugs.
- **[ ] Video Demo:** Record a short video demonstrating the full user interaction flow.

5.0 Contribution Summary

- **Kareem Elsenosy:** Kareem contributed to the implementation of the core game logic in Wonderland Engine, including boundary detection, completion tracking, and scoring algorithms. He also helped design interactive elements of the virtual environment, worked on integrating UI components, and participated in testing and refining the gameplay experience.
- **Adam Elghayaty:** Adam created and optimized 3D assets, designed the visual style of the virtual classroom, and helped implement object interactions in the Wonderland Engine. He also contributed to coding the visual and spatial elements of the game logic, assisted with UI integration, and participated in testing to ensure a seamless user experience.

- **Yehia Elsisi:** Yehia designed and implemented all user interface elements, integrated and balanced audio feedback, and contributed to coding gameplay mechanics and performance optimization. He also co-led user testing sessions, collecting feedback to enhance usability and overall player immersion.

6.0 References

- Wonderland Engine. (n.d.). Retrieved from https://wonderlandengine.com/
- Meta Quest Developer Hub. (n.d.). Retrieved from https://developer.oculus.com/
- Class room template (n.d) Retrieved from https://sketchfab.com/