A Fine Day

Wasfiyeh Al Jorf

- According to the WHO, 15 million people suffer from strokes annually [1]
- Based on statistics from the Department of Health in Abu Dhabi (DOH), approximately 8,000–10,000 Emiratis experience a stroke each year, translating to at least one stroke occurring every hour [2]
- Initially, some 80% of all patients with stroke experience motor impairments of the contralateral limb(s). [3]

Stroke Rehabilitation:

Different for everyone depending on severity of motor impairment and comes in the form of:

- **Motor-skill exercises:** Improving muscle strength and coordination
- **Mobility training:** Restore ability to move around whether aided or not
- **Constraint-induced therapy:** Constraining the unaffected limb to promote the use of the affected limb
- **Range-of-motion therapy.** Easing muscle tension to enlarge range-of-motion [4]

Often, patients are tasked to complete some of these exercises at home

BUT

Patient adherence with home exercise programs after discharge from rehabilitation is less than ideal. [5]

Reasons for Not Adhering:

- Seclusion/Isolation while performing the exercises
- Boring because of the monotonicity and repetitiveness
- Remind the patient of their impairment

Our Solution:

What if we integrated VR (Oculus Rift) and hand-tracking (Ultraleap) to replicate motor skill and range-of-motion exercises in more engaging way while keeping track of the motion data for the therapist to analyze?!

There was a significant 4.9 higher chance of improvement in motor strength for patients randomized to VR systems [6]

Choosing the Minigames

(Work Your Motor)[7]

Traditional Therapy

+

VR Therapy

(Rewillio, Real System)[8,9]

The Minigames:

3D Whack-A-Mole

Works on:

- Hand-eye coordination
- Range-of-Motion
- Speed

Has two versions:

- Non-timed: less stressful version of the game of which the objective is to hit a predefined number of targets. First few levels so the player can slowly progress.
- Timed: a race against time to hit at least a certain number of targets before the time runs out.

Apple Picking

Works on:

- Hand-eye coordination
- Range-of-Motion
- Grasping skills
- Manipulating the environment

Goal:

Place apples that randomly spawn on a tree in front of you into the basket. The player is scored according to their accuracy, i.e. how many apples the manage to place in the basket out of the total apples spawned

Implementation

Calibration Scripts

FindArmLength:

```
...
private void Update()
    {
if (wokeUp)
        {
            if (Math.Abs(rightIM.velocity.magnitude) < 0.5 &&
Math.Abs(vLast) < 0.5)
            {
                timeSpentStationary += Time.deltaTime;
            }
            else
            {
                timeSpentStationary = 0;
            }
            if (timeSpentStationary >= 5)
            {
                measureDistance();
            }
            vLast = rightIM.velocity.magnitude;
        }
    }

    private void measureDistance()
    {
        double length =
Mathf.Sqrt(Mathf.Pow(rightIM.leapHand.PalmPosition.x -
cameraTransform.position.x, 2) +
Mathf.Pow(rightIM.leapHand.PalmPosition.z -
cameraTransform.position.z, 2) +
Mathf.Pow(rightIM.leapHand.PalmPosition.z -
cameraTransform.position.z, 2));
        Levels.armLength = length*0.8;
        menuGroup.menus[0].gameObject.SetActive(true);
        wokeUp = false;
    }

...
```

Callibrate:

```
...
void Update()

    {

        if (isReady())

        {

            spawner.callibrated = true;

            Debug.Log("Goodbye World");

            enabled = false;

        }

    }


    public bool isReady()

    {

        bool isReady = this.transform.position.x > previousPosition.x +
0.2 || this.transform.position.y > previousPosition.y + 0.2 ||
this.transform.position.z > previousPosition.z + 0.2;

        previousPosition = this.transform.position;

        return isReady;

    }

...
```

Spawner Parent Class

```
...
[SerializeField]
    protected GameObject target;
    protected float xPos;
    protected float yPos;
    protected float zPos;
    [SerializeField]
    protected Transform cameraTransform;
    protected double armLength;
    protected double difficultyPercentage;
    public bool callibrated = false;

    private void Start()
    {
        armLength = Levels.armLength;
    }

    public abstract void spawn();
...
```

Minigame Spawners Scripts

BasketSpawner:

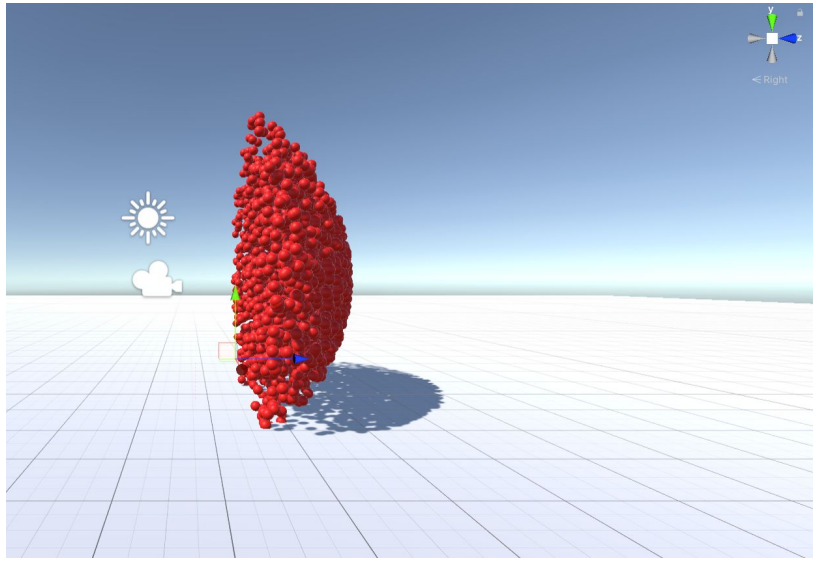
```
...
private void Start()
    {
        armLength = Levels.armLength;
        numberOfTargets = (Levels.levels[Levels.currentIndex] as
basketLevel).numberOfTargets;
        difficultyPercentage = (Levels.levels[Levels.currentIndex] as
basketLevel).difficultyPercentage;
    }

    public void Update(){
        if (basket.getCountdownEnded() && callibrated){
            spawn();
            callibrated = false;
        }
    }

    public override void spawn()
    {
        for (int i = 0; i < numberOfTargets; i++)
        {
            zPos = Random.Range((1-(float)difficultyPercentage)*(float)armLength,
(float)armLength);
            float X2 = Mathf.Pow((float)armLength, 2) - Mathf.Pow(zPos, 2);
            xPos = Random.Range(-Mathf.Sqrt(X2), Mathf.Sqrt(X2));
            yPos = Random.Range(treeHeight, Mathf.Sqrt(X2 - Mathf.Pow(xPos, 2)));
            Instantiate(target, new Vector3(xPos, yPos, zPos) + cameraTransform.position,
Quaternion.identity);
        }
    }
...
```

WAMSpawner:

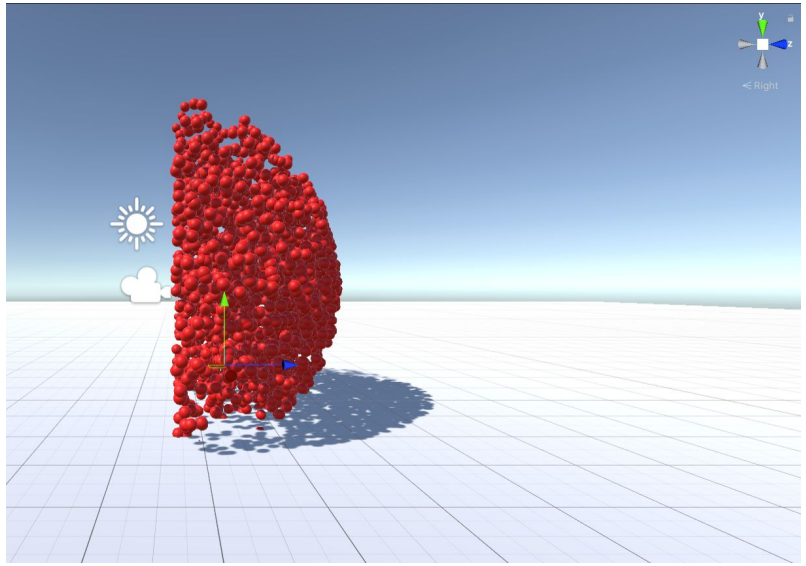
```
...
private void Start()
    {
        armLength = Levels.armLength;
        difficultyPercentage = (Levels.levels[Levels.currentIndex] as
wackLevel).difficultyPercentage;
    }

    private void Update()
    {
        if (callibrated)
        {
            spawn();
            callibrated = false;
        }
    }

    public override void spawn()
    {
        zPos =
Random.Range((1-(float)difficultyPercentage)*(float)armLength,
(float)armLength);
        float X2 = Mathf.Pow((float)armLength, 2) - Mathf.Pow(zPos, 2);
        xPos = Random.Range(-Mathf.Sqrt(X2), Mathf.Sqrt(X2));
        yPos = Random.Range(-Mathf.Sqrt(X2 - Mathf.Pow(xPos, 2)),
Mathf.Sqrt(X2 - Mathf.Pow(xPos, 2)));
        target.transform.position = new Vector3(xPos, yPos, zPos) +
cameraTransform.position;
    }
...
```

Eg. Whack a mole spawner

Test 1: Arm-length=3, Difficulty Radius = 0.5 (1000 Spawned Targets)



Test 2: Arm-length=3, Difficulty Radius = 1.5 (1000 Spawned Targets)



Interactions Scripts

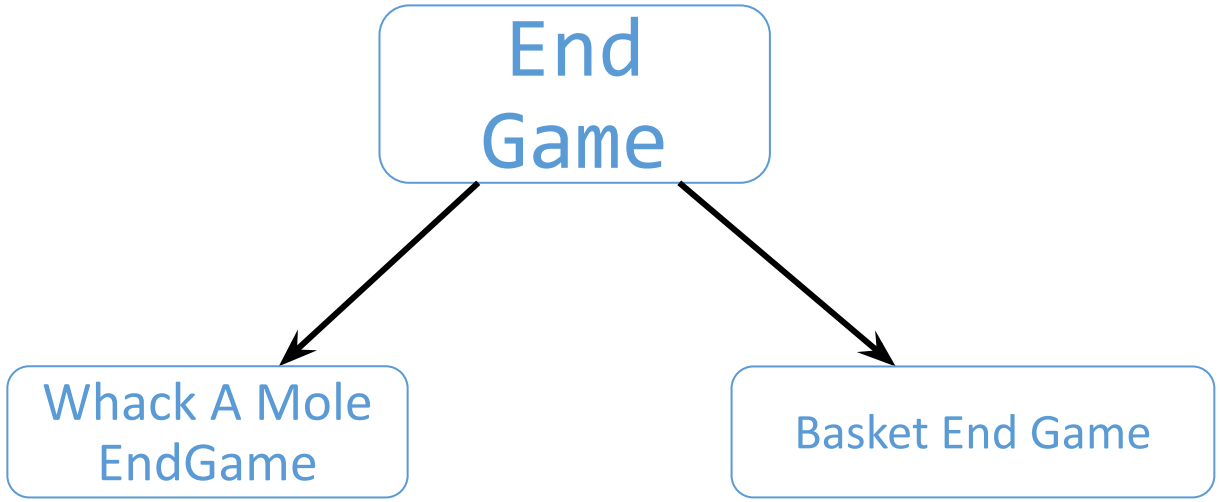
Baskets :

```
private void OnTriggerEnter(Collider collider)
    {
        if (collider.tag == "Target")
        {
            collider.tag = "Counted"; // To avoid
double counting of core
            score++;
            applesCount.text = customText + score;
            GameObject exploder =
((Transform)Instantiate(explosion,
collider.transform.position,
Quaternion.identity)).gameObject;
            Destroy(exploder, 1.0f);
            ender.checkEnd();
        }
    }
private void OnCollisionEnter(Collision collision)
    {
        if (collision.collider.tag == "Target")
        {
            collision.collider.tag = "Counted"; // To
avoid double counting of score
            score++;
            applesCount.text = customText + score;
            GameObject exploder =
((Transform)Instantiate(explosion,
collision.collider.transform.position,
Quaternion.identity)).gameObject;
            Destroy(exploder, 1.0f);
            ender.checkEnd();
        }
    }
```

Whack Target Collision:

```
    void OnTriggerEnter(Collider other)
    {
        if (countdownEnded && IsHand(other))
        {

FindObjectOfType<AudioManager>().Play("BasketScore");
            DateTime currentTime = DateTime.Now;
            motionData.addTo("timeBetweenCollisions",
DateTime.Now.Subtract(lastSpawn).TotalSeconds);
            if (motionData.getFrom("timeBetweenCollisions",
motionData.getLengthOf("timeBetweenCollisions") - 1) >=
0.2)
            {
                GameObject exploder =
((Transform)Instantiate(explosion,
this.transform.position, Quaternion.identity)).gameObject;
                Destroy(exploder, 1.0f);
                scoreCount++;

motionData.logCollision(this.transform.position);
                // Debug.Log("Current Score: " + scoreCount
+ "\nTimer Interval Since Last Collision: " +
motionData.getFrom("timeBetweenCollisions",
motionData.getLengthOf("timeBetweenCollisions") - 1);
                scoreCountText.text = "Score Count: " +
scoreCount;
                lastSpawn = DateTime.Now;
            }
            else
            {

motionData.removeAtEnd("timeBetweenCollisions");
            }
            spawner.spawn();

        }
    }
```

Ending Abstract



Ending Scripts

Wack A Mole Ending:

```
   public override void countScore()
    {
        menuGroup.menus[0].gameObject.SetActive(true);
        menuGroup.menus[0].wait(3);
        molesCollision.showScore(false);
        molesCollision.showTime(false);

        double score = molesCollision.getScoreCount();
        double totalTimeSpent = molesCollision.getAllowedTime();
        bool win = score >= (Levels.levels[Levels.currentIndex]as
wackLevel).targetsToPass;
        if (win)
        {

            if (Levels.currentIndex != Levels.levels.Length -1){

            menuGroup.menus[0].buttons[3].setText("Next Level");
            menuGroup.menus[0].buttons[3].levelIndex =
Levels.currentIndex + 1;
            menuGroup.menus[0].buttons[3].path =
Levels.getNextLevelType();

            }
            else {

menuGroup.menus[0].buttons[3].gameObject.SetActive(false);

            }


            menuGroup.menus[0].buttons[0].setText("Play Again");
            menuGroup.menus[0].buttons[0].levelIndex =
Levels.currentIndex;

            WinLostText.text ="You Won! :)";
            WinLostText.fontSize =50;
        }
        else
        {
            menuGroup.menus[0].buttons[0].setText("Try Again");
            menuGroup.menus[0].buttons[0].levelIndex =
Levels.currentIndex;

menuGroup.menus[0].buttons[3].gameObject.SetActive(false);
            WinLostText.text ="You Lost! :(";
            WinLostText.fontSize =50;
        }
        GameData runData = new GameData(Levels.currentIndex,
molesCollision.getMotionData(), score, totalTimeSpent, win);
        DataManager.gameData.Add(runData);
        DataManager.graphAccelerationData(runData.getRunID(),
accelerationGraph);
        DataManager.graphVelocityData(runData.getRunID(),
velocityGraph);

    }
```

Basket Ending:

```
public override void countScore()
    {
        menuGroup.menus[ 0].gameObject.SetActive( true);
        menuGroup.menus[ 0].wait(3);
        double accuracy = (double)basket.getScore() /
(basket.getScore() + ground.getScore());
        double totalTimeSpent =
DateTime.Now.Subtract(basket.getStartTime()).TotalSeconds;
        bool win = accuracy >=
(Levels.levels[Levels.currentIndex]  as
basketLevel).accuracyToPass;
        if (win)
        {
            menuGroup.menus[ 0].buttons[ 0].setText("Next
Level");
            menuGroup.menus[ 0].buttons[ 0].levelIndex =
Levels.currentIndex + 1;
            menuGroup.menus[ 0].buttons[ 0].path =
Levels.getNextLevelType();
            WinLostText.text =  "You Won! :)";
            WinLostText.fontSize =  50;
        }
        else
        {
            menuGroup.menus[ 0].buttons[ 0].setText("Try
Again");
            menuGroup.menus[ 0].buttons[ 0].levelIndex =
Levels.currentIndex;
            WinLostText.text =  "You Lost! :(";
            WinLostText.fontSize =  50;
        }
        GameData runData = new
GameData(Levels.currentIndex, basket.getMotionData(),
accuracy, totalTimeSpent, win);
        DataManager.gameData.Add(runData);

DataManager.graphAccelerationData(runData.getRunID(),
accelerationGraph);
        DataManager.graphVelocityData(runData.getRunID(),
velocityGraph);
    }
```

Data Handling

Velocity:

```
   public static void graphVelocityData(int run, windowGraph graph)
   {
       List<double> timeBetweenCollisions =
gameData[run].getMotionData().getList("timeBetweenCollisions");
       List<double> leftHandVelocities =
gameData[run].getMotionData().getList("leftHandVelocities");
       List<double> rightHandVelocities =
gameData[run].getMotionData().getList("rightHandVelocities");
       List<float> rightHandVelocitiesFloat =new List<float>();
       List<float> leftHandVelocitiesFloat =new List<float>();
       for (int i = 0; i < rightHandVelocities.Count; i++)
       {


rightHandVelocitiesFloat.Add((float)rightHandVelocities[i]);


       }
       for (int i = 0; i < leftHandVelocities.Count; i++)
       {


leftHandVelocitiesFloat.Add((float)leftHandVelocities[i]);


       }
       float totalTimeSpent = (float)gameData[run].getTimeSpent();
       List<double> timesOfRHCollisions =
gameData[run].getMotionData().getList("timesOfRHCollision");
       List<double> RHVelocitiesAtCollision =
gameData[run].getMotionData().getList("RHVelocitiesAtCollisions");
       List<double> timesOfLHCollision =
gameData[run].getMotionData().getList("timesOfLHCollision");
       List<double> LHVelocitiesAtCollision =
gameData[run].getMotionData().getList("LHVelocitiesAtCollisions");
       double logRate = gameData[run].getMotionData().getLogRate();
        graph.showGraph(rightHandVelocitiesFloat, totalTimeSpent,
timesOfRHCollisions, RHVelocitiesAtCollision,'r', logRate);
        graph.showGraph(leftHandVelocitiesFloat, totalTimeSpent,
timesOfLHCollision, LHVelocitiesAtCollision,'l', logRate);
   }
```

Acceleration:

```
public static void graphAccelerationData(int run, windowGraph
graph)
   {
       List<double> timeBetweenCollisions =
gameData[run].getMotionData().getList("timeBetweenCollisions");
       List<double> leftHandAccelerations =
gameData[run].getMotionData().getList("leftHandAccelerations");
       List<double> rightHandAccelerations =
gameData[run].getMotionData().getList("rightHandAccelerations");
       List<float> rightHandAccelerationsFloat =new List<float>();
       List<float> leftHandAccelerationsFloat =new List<float>();

       for (int i = 0; i < rightHandAccelerations.Count; i++)
       {


rightHandAccelerationsFloat.Add((float)rightHandAccelerations[i]);


       }
       for (int i = 0; i < leftHandAccelerations.Count; i++)
       {


leftHandAccelerationsFloat.Add((float)leftHandAccelerations[i]);


       }
       float totalTimeSpent = (float)gameData[run].getTimeSpent();
       List<double> timesOfRHCollisions =
gameData[run].getMotionData().getList("timeBetweenCollisions");
       List<double> RHVelocitiesAtCollision =
gameData[run].getMotionData().getList("RHVelocitiesAtCollisions");
       List<double> timesOfLHCollision =
gameData[run].getMotionData().getList("timesOfLHCollision");
       List<double> LHVelocitiesAtCollision =
gameData[run].getMotionData().getList("LHVelocitiesAtCollisions");
       double logRate = gameData[run].getMotionData().getLogRate();
        graph.showGraph(rightHandAccelerationsFloat, totalTimeSpent,
timesOfRHCollisions, RHVelocitiesAtCollision,'r', logRate);
        graph.showGraph(leftHandAccelerationsFloat, totalTimeSpent,
timesOfLHCollision, LHVelocitiesAtCollision,'l', logRate);
   }
```

Scaling the graph to the window

```
 public void showGraph(List<float> velocities, float totalTime, List<double> CollisionTimes, List<double> velocitiesAtCollisions, char handedness, double lr)
    {
        logRate = lr;
        int current = 0;
        graphContainer = transform.GetComponentInChildren<RectTransform>();

        float graphHeight = graphContainer.sizeDelta.y;

        float graphWidth = graphContainer.sizeDelta.x;

        float yMaximum = Mathf.Max(velocities.ToArray());
        float xSize = graphWidth / velocities.Count;

        GameObject lastCircleGameObject = null;

        for (int i = 0; i < velocities.Count; i++)
           {

            float xPosition = i * xSize;
            float yPosition = (velocities[i] * (graphHeight / yMaximum));

            GameObject circleGameObject = new GameObject();

            if (current < CollisionTimes.Count)
            {
                if (((xPosition * totalTime) / graphWidth - CollisionTimes[current]) >= 0)
                {
                    current++;
                    circleGameObject = CreateCircle(new Vector2(xPosition, yPosition), true, handedness);

                }
                else
                {
                    circleGameObject = CreateCircle(new Vector2(xPosition, yPosition), false, handedness);
                }
            }
            else
            {
                circleGameObject = CreateCircle(new Vector2(xPosition, yPosition), false, handedness);
            }


if (lastCircleGameObject != null)
            {
                CreateDotConnection(lastCircleGameObject.GetComponent<RectTransform>().anchoredPosition, circleGameObject.GetComponent<RectTransform>().anchoredPosition, handedness);
            }

            lastCircleGameObject = circleGameObject;
        }

    }
```

Graphing Scripts

Plotting Velocities/Accelerations:

```
 private GameObject CreateCircle(Vector2 anchoredPosition,  bool show, char
handedness)
    {
        GameObject point =  new GameObject("point", typeof(Image));
        RectTransform rectTransform =  new RectTransform();

        if (handedness ==  'l')
        {
            point.transform.SetParent(graphContainer,  false);
            point.GetComponent<Image>().sprite = leftSprite;
            rectTransform = point.GetComponent<RectTransform>();
            rectTransform.anchoredPosition = anchoredPosition;
            if (show)
            {
                rectTransform.sizeDelta =  new Vector2(40, 40);
            }
            else
            {
                rectTransform.sizeDelta =  new Vector2(0, 0);
            }
        }
        else
        {
            point.transform.SetParent(graphContainer,  false);
            point.GetComponent<Image>().sprite = rightSprite;
            rectTransform = point.GetComponent<RectTransform>();
            rectTransform.anchoredPosition = anchoredPosition;
            if (show)
            {
                rectTransform.sizeDelta =  new Vector2(40, 40);
            }
            else
            {
                rectTransform.sizeDelta =  new Vector2(0, 0);
            }
        }
        rectTransform.anchorMin =  new Vector2(0, 0);
        rectTransform.anchorMax =  new Vector2(0, 0);
        return point;
    }
```

Creating Connections

```
 private void CreateDotConnection(Vector2 dotPositionA,
Vector2 dotPositionB,  char hand)
    {
        GameObject gameObject =  new
GameObject("dotConnection", typeof(Image));
        gameObject.transform.SetParent(graphContainer,
false);
        if (hand ==  'l')
        {
            gameObject.GetComponent<Image>().color =  new
Color(1,  0,  0,  1f);
        }
        else if (hand ==  'r')
        {
            gameObject.GetComponent<Image>().color =  new
Color(1,  1,  1,  1f);
        }
        RectTransform rectTransform =
gameObject.GetComponent<RectTransform>();
        Vector2 dir =  (dotPositionB -
dotPositionA).normalized;
        float distance = Vector2.Distance(dotPositionA,
dotPositionB);
        rectTransform.anchorMin =  new Vector2(0,  0);
        rectTransform.anchorMax =  new Vector2(0,  0);
        rectTransform.sizeDelta =  new Vector2(distance,
3f);
        rectTransform.anchoredPosition = dotPositionA +
dir * distance * 0.5f;
        rectTransform.localEulerAngles =  new Vector3(0,  0,
(Mathf.Atan2(dir.y,  dir.x)  * 180 / Mathf.PI));
    }
```

Future Development

- Personalizing levels by analyzing the user progress. This could be implemented in our solution by integrating the Levels and DataManager scripts into a LevelSpawner.
- Adding more minigames:
  - Air-hockey
  - Placing objects in precise positions
  - Pat-a-cake
  - Fishing

Lessons Learned:

- Patience
- “The Programmer’s Mindset”
  - Modularity
  - Testing and Debugging
  - Refactoring old code
  - Always being willing to throw away code you worked hard on in favor of better written-code
- Working with few resources (Using documentation and premade examples)
- Learning to operate new devices
- Listening to user feedback:
  - The W-A-M minigame was always timed, but people kept mentioning how stressful that would be as a first level.

References

[1] http://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html#:~:text=Annually%2C%2015%20million%20people%20worldwide%20suffer%20a%20stroke.

[2] https://www.khaleejtimes.com/uae/every-hour-one-person-gets-a-stroke-in-uae

[3] https://www.ahajournals.org/doi/full/10.1161/01.STR.0000105386.05173.5E

[4] https://www.mayoclinic.org/diseases-conditions/stroke/in-depth/stroke-rehabilitation/art-20045172

[5] https://pubmed.ncbi.nlm.nih.gov/27334684/

[6] https://pubmed.ncbi.nlm.nih.gov/27334684/

[7] https://www.youtube.com/channel/UCTruhuRf8ODu6O4qVppA3Cw

[8] https://www.rewellio.com/vr-virtual-reality/

[9] https://www.realsystem.com/

Demonstration