AcuteStrokeSimulatorVR

By: Trajan Salinas, Richard Kelley, Joshua Hardy
Client: Dr. Askiel Bruno

Department of Neurology

Purpose

- Provide crucial missing hand-on training to medical professionals

- Provide a realistic environment to prepare users for real world situations

- Enhance diagnostic and decision-making skills

Approaching the Problem

- Making a Simulator game in VR

- HTC Vive headset and SteamVR framework

- Early and frequent concept builds to test functional requirements

Challenges

- Learning Unity
- VR world still has many conflicts
  - This made it hard to determine path for user input
- Starting from nothing

Functionality Overview

- Main menu -> Simulation room -> Summary scene
- Movement within simulation
- Interactable buttons and menus
- Clock and vitals

Technical Implementation
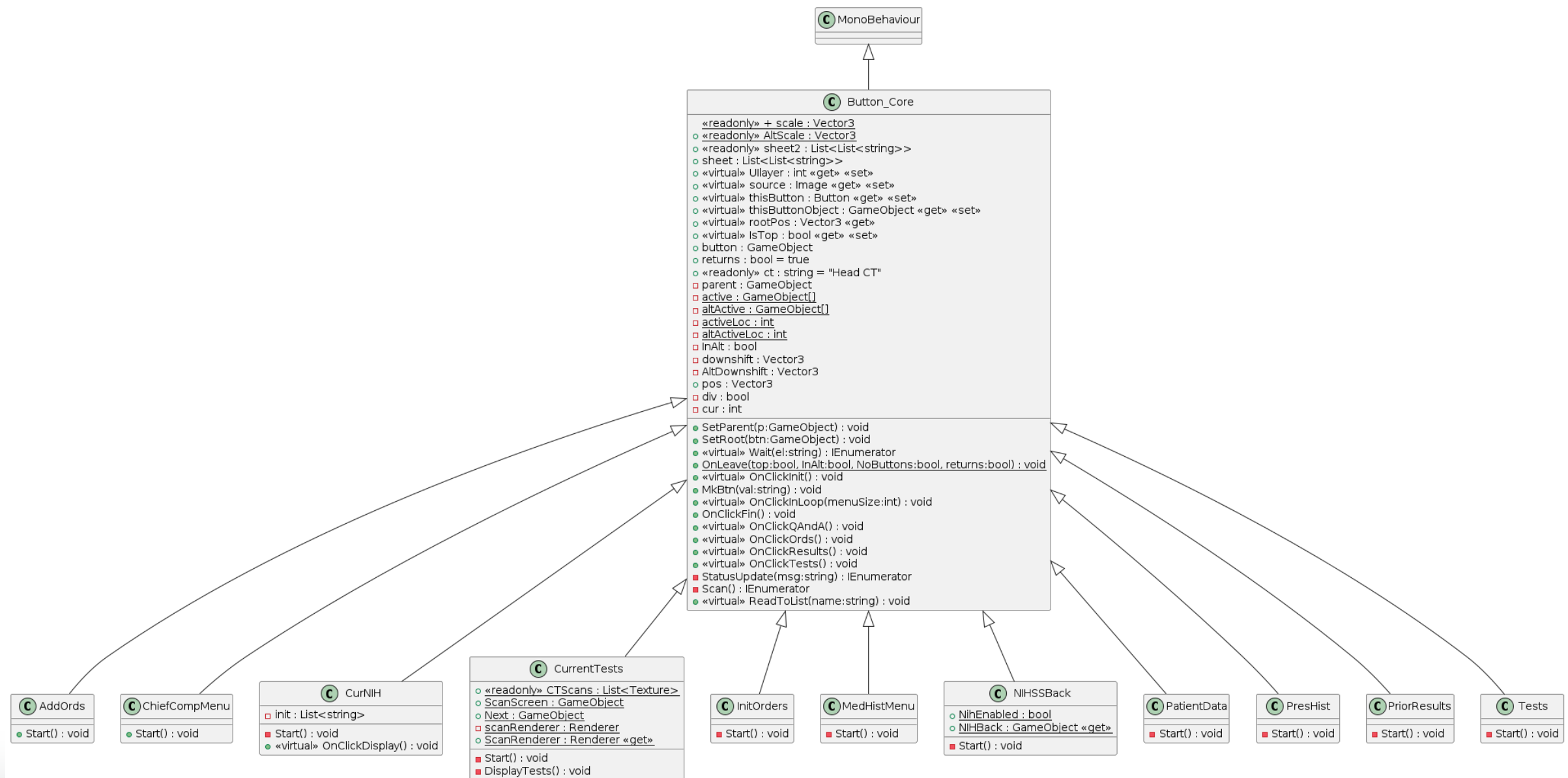
```csharp
5 references
public virtual void OnClickInit() //sets up initial menus
{

    pos = rootPos;
    div = false;
    if (!InAlt)
    {
        Core.subMenus.Clear();
    }
    else {
        Core.AltSubMenus.Clear();
    }

    if (!IsTop && !InAlt)
    {
        foreach (var el in Core.Menus.Peek())
        {
            el.SetActive(false); //deactivates the current menu
        }
        Core.Menus.Push(Core.Menus.Peek());
    }
    else if (!InAlt)
    {
        Transform[] children = transform.parent.GetComponentsInChildren<Transform>();
        List<GameObject> tmp = new();
        foreach(var child in children)
        {
            tmp.Add(child.gameObject);
            child.gameObject.SetActive(false);
        }
        Core.Menus.Push(tmp); Core.Menus.Push(tmp);
    }
    else if (!IsTop)
    {
        foreach (var el in Core.AltMenus.Peek())
        {
            el.SetActive(false); //deactivates the current menu
        }
```

```csharp
public static IEnumerator Motor(string limbName)
{

    bool up, down, hold;
    Transform limb = null;
    Vector3 start, end;

    float timePassed = 0f;
    float holdTime = 0f;
    up = true; down = false; hold = false;
    start = new Vector3();
    end = new Vector3();
    switch (limbName)
    {
        case "LA":
            limb = GameObject.Find("LArm_target").GetComponent<Transform>();
            holdTime = HoldT("/5bMotorLA", holdTime);
            start = limb.position;
            end = GameObject.Find("lATarget").GetComponent<Transform>().position;
            break;
        case "RA":
            limb = GameObject.Find("RArm_target").GetComponent<Transform>();
            holdTime = HoldT("/5aMotorRA", holdTime);
            start = limb.position;
            end = GameObject.Find("rATarget").GetComponent<Transform>().position;
            break;
        case "LL":
            limb = GameObject.Find("LLeg_target").GetComponent<Transform>();
            holdTime = HoldT("/6bMotorLL", holdTime);
            start = limb.position;
            end = GameObject.Find("lLTarget").GetComponent<Transform>().position;
            break;
        case "RL":
            limb = GameObject.Find("RLeg_target").GetComponent<Transform>();
            holdTime = HoldT("/6aMotorRL", holdTime);
            start = limb.position;
            end = GameObject.Find("rLTarget").GetComponent<Transform>().position;
            break;
        default:
            up = false; hold = false; down = false;
            break;
```

```csharp
1 reference
private Quaternion CalculateOrientation()
{
    float rotation = Mathf.Atan2(moveValue.axis.x, moveValue.axis.y);
    rotation *= Mathf.Rad2Deg;

    Vector3 orientationEuler = new Vector3(0, head.eulerAngles.y + rotation, 0);
    return Quaternion.Euler(orientationEuler);
}

1 reference
private void SnapRotation()
{
    float snapValue = 0.0f;

    if (rotatePressLeft.GetStateDown(SteamVR_Input_Sources.LeftHand))
    {
        snapValue = -Mathf.Abs(rotateIncrement);
    }

    if (rotatePressRight.GetStateDown(SteamVR_Input_Sources.RightHand)) //changed from right hand
    {
        snapValue = Mathf.Abs(rotateIncrement);
    }

    transform.RotateAround(head.position, Vector3.up, snapValue);
}
}
```

Video Demo

https://www.youtube.com/watch?v=luI7_ts0Jdg

Differences from Original Requirements

- Three scenarios -> One scenario
- Tests of 15-30 minutes -> 3-15 minutes
- Scan of 25-40 minutes -> 10 minutes
- BP fluctuate after 15 minutes -> after 7
- Summary on panel -> separate scene
- Added: persistent data, CT scans, notifications, movement

Questions?