Code Testing Plan

Executive Summary

This document features a list of all conditions that were tested throughout the development of ArachnoTherapy VR. It is extremely difficult to run instantaneous automated testing for Unity projects with a linear progression of mechanics and logic and this difficulty only compounds when certain modules can only be invoked by a user's physical actions. As a result, the testing that we conducted on our project's code base usually involved either gradual observation of modules in real-time (*passive testing*), or physically donning the VR headset and interacting with new components ourselves (*invasive testing*).

Modules that operated cyclically or were capable of self-advancement were generally tested passively. Examples included the spider behaviour module in MVP1, which was tested by simply placing pre-compiled objects of each spider type in an enclosed virtual space and observing their behaviours. Similarly, the therapist character audio clips that play in succession at the beginning of the experience could be tested by simply running the application and listening. However, all modules that listen for interactions (i.e. collisions or button presses from the user's Oculus Touch controllers) directly, or are dependent on previous interactions having occurred, needed to be tested invasively. As will become clear upon more closely examining the application's conditions for validation, some invasive tests were highly focused, while others could only be judged qualitatively. For example, in testing the Add One Spider button before options for random diversity and intensity were added, there were 12 clear test cases: one for each combination of diversity and intensity. However, it would have been infeasible to test that collisions between the controllers and every possible point on an interactable object's mesh would cause the object to highlight. These types of tests were purely qualitatively observational: one of us would simply move our virtual hands around each grabbable object, and verify that the highlight material would activate and deactivate in a reasonable manner.

All of the conditions for validation listed in the sections below passed prior to the application's final deployment.

ArachnoTherapy VR

ENSE 400 / 477 Capstone - Team Edentata

MVP 1

- **Spider Behaviour (passive)**

  - Scripts involved:
    - AbstractSpider and all child classes
  - Conditions for validation:
    - Periodic decision-making
      - BasicSpiders could choose to idle or move
      - IntermediateSpiders could choose to idle, move, or jump
      - ComplexSpider could choose to idle, move, jump, or attack
      - ExtremeSpiders could choose to move, jump, or attack
      - New actions would only be chosen after the specified time interval had passed (5 s for BasicSpiders, -0.5 s for each increase in complexity)
    - Animation transitions
      - Appropriate animation would play for each chosen action
      - *Idle* and *Move* animations would loop
      - *Jump* and *Attack* animations would automatically transition to a looping animation
    - Collision logic
      - Appropriate action(s) would be taken upon collision with another spider
        - BasicSpiders would run from everything
        - IntermediateSpiders would do nothing to BasicSpiders or other IntermediateSpiders, and run from ComplexSpiders and ExtremeSpiders
        - ComplexSpiders would attack BasicSpiders and IntermediateSpiders, and run from other ComplexSpiders and ExtremeSpiders
        - ExtremeSpiders would attack everything
      - Appropriate action(s) would be taken upon collision with the player
        - BasicSpiders would always run
        - IntermediateSpiders could run or do nothing
        - ComplexSpiders could do nothing, run, jump, or attack
        - ExtremeSpiders would always attack
      - Movement direction would change upon collision with a wall

- **Oculus Controller Input & Object Manipulation (invasive)**

  - Scripts involved:
    - Movement, Controller, Grabber, AbstractGrabbable and all child classes, AbstractButton and all child classes
  - Conditions for validation:
    - Left joystick translation
      - The player's vertical position would remain constant
      - The player could not pass through walls or tall objects
      - All movement would stop if the user released the joystick
    - Grabbable object manipulation
      - Grabbable objects could only be grabbed if the Grip button was pressed on either controller

        - Grabbed objects would map to the correct hand (i.e. the controller whose Grip button was pressed), which required correctness of the Hand enumeration and Controller data class
        - Thrown objects would travel at realistic translational and rotational velocities
    - Button object manipulation
        - Button objects could only be “pressed” if the Trigger button was pressed on either controller
        - Button objects would move to their pressed position after being interacted with
        - Button objects would return to their unpressed position when the user released the Trigger

- **Door Physics (invasive)**

- Scripts involved:
    - Handle, HandleProxy
- Conditions for validation:
    - Door logic
        - Doors could neither be translated nor rotated along the pitch and roll axes
        - Collisions between the player’s head and the door would not affect the player’s position
        - Doors would not continue to rotate if its handle was released by the player
        - The angle of a door, with the closed rotation treated as 0 degrees, could not exceed 135 degrees
    - Handle logic
        - The visible handle would not snap to the user’s hand position as other grabbable objects do
        - The visible handle would follow the position of the invisible “grab box handle” within the door’s existing rotational constraints

- **Button Events (invasive)**

- Scripts involved:
    - AbstractButtonEvent and all child classes, AbstractButton and all child classes
- Conditions for validation:
    - Audio button events
        - Audio would play if a Button object with an associated AudioButtonEvent reference was interacted with
        - Audio would not loop
    - Text button events (later removed from the design space)
        - A Text element’s visibility would be toggled if a Button object with an associated TextButtonEvent reference was interacted with
    - Video button events
        - A video would begin playing if a Button object with an associated VideoButtonEvent was interacted with for the first time
        - Subsequent interactions with the same Button object would toggle the play status of the Video
        - Videos would loop

- **Spider Creation Interface (invasive)**

  - Scripts involved:
    - SpiderCreatorModel, SpiderCreatorSingleton, SpiderAmountController, SpiderDiversityController, SpiderIntensityController, SingleCreationController, SingleDeletionController, MassCreationController, MassDeletionController, DefaultState
  - Conditions for validation:
    - Spider amount
      - User-controlled minimum amount of spiders could not decrease below 0
      - User-controlled maximum amount of spiders could not exceed 20
    - Spider diversity
      - The diversity would be set to Common House Spider by default
      - Only one diversity setting could ever be active at a given time
    - Spider intensity
      - The intensity would be set to Low by default
      - Only one intensity setting could ever be active at a given time
    - Spider creation
      - The correct spider(s) would be created based on the user-specified diversity and intensity settings (i.e. Basic Tarantula, Complex Common House Spider, Intermediate Wolf Spider, Random Extreme Spider, Random Tarantula, etc.)
      - The Add One Spider button would always create one spider if the number of existing spiders was less than the user-controlled maximum amount
      - The Add One Spider button would not create a spider if the number of existing spiders was equal to the user-controlled maximum amount
      - The Add All Spiders button would continue creating spiders until the user-specified maximum amount was reached
      - The Add One Spider button would not create any spiders if the number of existing spiders was equal to the user-controlled maximum amount
    - Spider deletion
      - The Remove One Spider button would always remove one spider if the number of existing spiders was greater than 0
      - The Remove One Spider button would do nothing if no spiders were present
      - The Remove All Spiders button would always remove all spiders

MVP 2

- **Conditional Room Display (invasive)**

  - Scripts involved:
    - RoomActivator, RoomColliderArea
  - Conditions for validation:
    - Player presence in each room
      - Each room would be aware of whether or not the player was inside of it
      - Each room could communicate its player status to a central RoomActivator object
    - Room visibility
      - Only the room in which the player was currently located, and any rooms adjacent to said room, would be visible at any given time

- **Preliminary Therapist Character Audio (passive)**

  - Scripts involved:
    - No backend scripting (AudioSource settings in Unity project)
  - Conditions for validation:
    - Preliminary introduction audio would play on awake (i.e. when the application was launched)

- **Spider Haptics (invasive)**

  - Scripts involved:
    - HapticGenerator, HapticReceiver
  - Conditions for validation:
    - Haptic feedback generation
      - Each spider type would be initialized with specific amplitude and frequency values
    - Haptic feedback execution
      - A vibration function (with amplitude and frequency dictated by the spider type) would be applied to the controller that “touched” a spider object
      - The vibration would cease when the user’s virtual hand lost contact with the spider
      - If the user’s virtual hand remained on the spider, then the vibration would automatically cease after 1.5 seconds

MVP 3

- **Room Condition Logic (invasive)**

  - Scripts involved:
    - RoomController, RoomParameterGrabbable, ChildButton, TransitionButton
  - Conditions for validation:
    - Transition button logic
      - Door would be locked on awake
      - Button press would cause all room conditions to be checked
      - Door would not unlock on button press if any tasks were unfulfilled
      - Door would unlock on button press if all tasks had been fulfilled
      - Window glass would remain tinted if any tasks were unfulfilled
      - Window glass would become fully transparent if all tasks had been fulfilled
    - Room task fulfillment
      - Picking up a room parameter grabbable would update the room controller's list of Boolean conditions
      - Pressing a room parameter button would update the room controller's list of Boolean conditions
    - Therapist character audio
      - The "Room X transition" clip would play if a transition button was pressed and all tasks were fulfilled

- **Highlight Required Interactables on Hover (invasive)**

  - Scripts involved:
    - RoomParameterGrabbable, Grabber
  - Conditions for validation:
    - Applying the highlight material
      - Yellow highlight material would be applied on hand hover
    - Removing the highlight material
      - The room parameter grabbable's initial material would be re-applied on termination of hover
      - The room parameter grabbable's initial material would be re-applied on grab

- **Exclusive Spawner for Each Spider (mixed)**

  - Scripts involved:
    - MassCreationController, SpiderCreatorModel
  - Conditions for validation:
    - Initialization (passive)
      - All SpiderCreatorModel objects required a list of 20 spawners
    - Execution (invasive)
      - The Add All Spiders Button would create no more than one spider per spawner for any added amount between 2 and 20

MVP 4

- **Revamped Interactable Highlight Logic (invasive)**

  - Scripts involved:
    - Grabber, AbstractButton and all child classes, AbstractGrabbable and all child classes
  - Conditions for validation:
    - Optional grabbable objects
      - Yellow highlight material applied to object on hand hover
      - Object's initial material re-applied on grab or termination of hover
    - Room parameter grabbables
      - Yellow highlight material replaced by pink highlight material
    - Button objects
      - Blue highlight material applied to object on hand hover
      - Object's initial material re-applied on press or termination of hover

- **Extra Room Condition Logic (invasive)**

  - Scripts involved:
    - RoomController, TransitionButton
  - Conditions for validation:
    - Permanent transition button colour change
      - Base material colour would change from red to green on button press if all room conditions had been fulfilled
    - Therapist character audio
      - The "incomplete room" audio clip would play if a transition button was pressed and tasks were unfulfilled

- **Room 3 Educational Info (invasive)**

  - Scripts involved:
    - AudioButtonEvent
  - Conditions for validation:
    - Toggleable audio parameter
      - AudioButtonEvents could be specified as having toggleable audio or not
    - Random audio selection
      - AudioButtonEvents with toggleable audio would randomly select a clip from a pre-constructed array before playing any audio

MVP 5

- **Room 0 Auditory Instruction Sequence (mixed)**

  - Scripts involved:
    - ManyToOneAudioSource, PlayAudioOnDelay, PlayAudioOnDisplacement, PlayAudioOnHeadCollision, AudioAndVideoButtonEvent, ActivateObjectRenderer
  - Conditions for validation:
    - Introductory audio clip (passive)
      - Audio would play on awake
      - Audio would not loop
    - Controller audio clip (passive)
      - Audio would play after a specified delay time
      - Audio would not loop
    - Object interaction audio clip (invasive)
      - Audio would play upon the first collision between player head and field surrounding bulletin board
      - Audio would not loop
      - Audio would not play upon any future collisions between the aforementioned objects
    - Video preview audio clip (invasive)
      - Audio would play upon the player grabbing a piece of paper from the bulletin board for the first time
      - Audio would not loop
      - Audio would not play upon the player grabbing additional objects
    - Diaphragmatic breathing video & audio (invasive)
      - Audio and video would play upon button press
      - Further button presses would toggle video play status, but not affect audio
      - Video would loop, but audio would not
    - Diaphragmatic breathing screen (invasive)
      - Screen object would appear upon playing of video preview audio clip
    - Room 0 transition button (invasive)
      - Button and text panel would appear upon completion of diaphragmatic breathing audio playthrough

MVP 6

- **Button Press to Finish Play Session**

  - Scripts involved:
    - FinishButton
  - Conditions for validation:
    - Play session stops when button is hit during play session in Unity Editor
    - Application is exited when button is hit during play session in built application

- **New Passive Jar Mechanics**

  - Scripts involved:
    - UpdateRoomConditionsOnCollision
  - Conditions for validation:
    - Spider does not constantly “collide” with the event trigger space and change direction
    - Room condition is fulfilled upon “collision” with the player’s head