Laura Lantz Technical Interview

Pre-interview exercise for **VR Design Engineer** role | March 2017

The challenge

Your team is looking at introducing a spell-casting mechanic into a Unity-based game using a VR controller. You have been asked to develop both concepts and proposed implementations for several spells, including a spell in which a creature is summoned, a projectile is thrown, a character is healed, and a magically locked door is opened. Assume that this game is using either the Oculus Touch or Vive controller, depending on which you are more familiar with. Develop a concept for the interface and gesture that would be used to cast this spell, including interaction with any necessary props, such as a wand, crystal ball, or whatever other magical paraphernalia you deem appropriate. Propose pseudo code for how the game can detect that each spell is being cast. Consider differentiation between spells based solely on gesture (ie, no button to choose which spell you're casting), and think about issues such as accessibility, different levels of mobility, and learnability. Your solution should consist of proposed designs, some amount of pseudo-code, and a list of questions for discussion. You may also present and discuss existing libraries for gesture recognition. If you do, be prepared to explain them in some detail. This problem is open-ended, so if you have a grand idea that is plausible and within the spirit of the original problem, feel free to include it - your team expects you to be willing to improvise a little. Note also, that as this is an open-ended creative problem, there really is no ideal solution.

Additional info

Experience goals/design pillars

The experience should induce wonder and delight and create a world into which players can escape from their day to day lives. The experience should focus on exploration, discovery and a sense of being somewhere else. Danger is not a big element of this experience, but some mild threats may exist (though threats would only ever inconvenience players, not harm them). The world is light fantasy, with elements of modernity but design that appeals to a simpler, more magical time. The interface should be low in complexity but give a feeling of expressiveness. We want players to feel like the interface is a natural way of achieving the effects that are occurring.

Intended audience

Nothing about the experience would be designed to preclude any normally abled adult. That said, the experience should be designed to particularly appeal to adult women, particularly older women with families.

Interaction mode

The experience should support seated players with normal adult flexibility.

Notes on spell-casting system

Spell casting system is wand-based.

- Wands have clear and immediate associations with magic (good fit for audience)
- Wands are hand-held and don't involve a lot of full-body movement (good for seated experience)
- Wands extend the player's reach and let players exaggerate small motions into larger ones (good fit for gesture-based system, also helps avoid controller/headset collisions with some gestures)
- Associating all magical actions with a specific tool can be helpful to player and can also provide contextual information for gesture recognition -- no need for the game to attempt to detect magic-casting gestures when player is not holding wand
    - Wand can be kept on desk when not in use (other possibilities: up sleeve, or in sheath/holster on back)
- Consider incorporating aspects of the actual game controller into wand design (ex: an enlarged "wand grip" with trigger, buttons, etc -- using a fantasy-steampunk style of wands could work well for this.)
    - When running demos, I've noticed that older players are often thrown off by not being able to see their hand on the control (even when there are other forms of visual feedback when they press a button). Showing a hand on a controller-like wand grip can help reinforce correct hand position and provide a kind of visual feedback that these players seem to crave.
- Incorporate player-relative movements and trigger presses in distinctive ways
    - This further clarifies player intent, provides more prerequisites to gesture detection and helps avoid "false positives"
    - We can give immediate feedback in the form of slight controller vibration, audio feedback (faint magical sound of twinkling or chiming), and visual feedback (wand takes on a slight glow, tip leaves slight trail)

Real-time continuous detection & feedback

- Provide feedback to the player *while* they are performing magical actions.
    - This can instill player confidence by communicating how their action is being interpreted, reinforcing interface knowledge, and encouraging action completion (or intentional discontinuation).

Notes on spells

Design spells to make the player feel competent fast. Spells accomplish this by:

- Being easy to perform
- Being easy to remember
    - Spells should feel clearly differentiated from one another (this helps avoid player embarrassment, because spells are not easily confused or misfired)
    - Spells should feel more concrete/functional than abstract/emblematic (think of the difference between learning a tool/technique vs learning a language)
- Providing lots of real-time feedback during performance

Design spells to give players a sense of autonomy

- Support different play styles/self-expression—let players "make it their own"

Spell concepts

Summoning portal

...a spell in which a creature is summoned...

1. Player can hold down controller trigger while drawing a single-stroke closed shape (within a given size range; any plane orientation in 3D space) to form portal
2. Holding down the trigger while drawing a second stroke that passes (more-or-less) perpendicularly through portal determines portal directionality and “pulls” summoned creature out

    - Portal orientation and directionality could have minor effects (ex: whenever a roly-poly creature with tiny wings is summoned, it flaps heavily through the air to hover at eye level. When summoned from a downward-facing portal, it first tumbles out and drops a short distance before catching itself with rapid wing-flapping.)
    - Size of portal could influence size of creature
    - Possible extension: Could enable summoning of different types of creatures through...
        - ...use of differently-shaped portals (ex: circle, square, triangle, etc.)
        - ...positioning of portal (ex: portals below eye level vs. portals above eye level)
        - ...orientation/direction of portal (ex: pulling toward you through portal generates defensive creature, pushing outwards generates offensive creature)
    - Alternative paradigm: instead of portals being drawn in space directly by the tip of the wand, the portal edge is projected to the nearest surface. This would enable players to easily draw much larger portals if summoned creatures need to be larger.

Magic missile

...a spell in which a projectile is thrown...

1. Player moves tip of wand around quickly to gather projectile energy (visual, audio, and haptic feedback indicates amount of energy gathered. Projectile energy can be stored indefinitely (similar to Samus’s Charge Shot in Smash Bros.)
2. Player presses and holds trigger, draws tip of wand back in straight line (slingshot-style), and releases trigger to send projectile energy flying as a magic missile (as wand is drawn back, vague glowing line appears and updates continuously in real-time to show direction and distance/force with which projectile will be released)

    - Projectile-energy-gathering can be accomplished through short rapid “shaking” motions or large sweeping motions of the controller -- player has freedom in making movements that are easiest and most satisfying to them
    - Both the amount of energy gathered to form the magic missile and the force with which it is slingshot modulate its effects
    - No “friendly fire” -- if magic missile isn’t fired a minimum range it simply dissipates (for example, if the player clicks the trigger instead of pressing and holding it)
    - Should incorporate some amount of aim-assist/target-lock (test & tune to get a good feel)
    - Could simplify by removing slingshot component and simply have sight-line extend from wand to indicate path along which missile will travel when trigger is clicked -- missile explodes upon contact with any object
    - Players should be able fire magic missiles fairly rapidly -- start initial testing with a goal of 2 seconds of wand-shaking for optimal-energy-gather and 1 second drawback for optimal fired distance

- Experiment with maximum energy-gather cap -- could be some interesting tradeoffs between firepower and rate of fire
- There is danger of players throwing or dropping controllers while casting this spell -- use of wrist straps is advised
- Alternative possibility: Player must make a fast wand-tip movement such as a wand flick (meeting a high speed threshold) to enter energy-gathering mode, but once this mode is activated they can keep the wand in continuous motion at a much lower minimum speed to gather energy.

Breath of life

...a spell in which a character is healed...

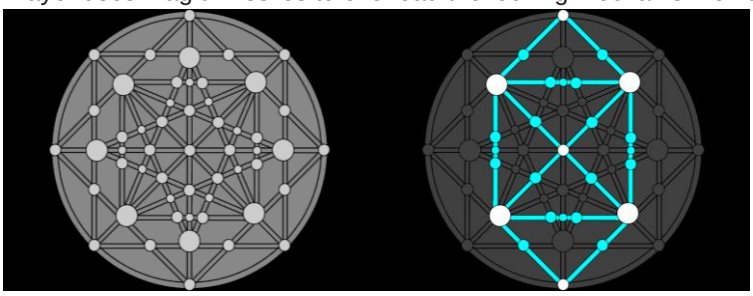
1. Player can raise the tip of their wand near their face to generate a pale green spark that hovers at the end of the wand
2. Blowing onto the spark gathers healing energy (spark glows brighter and greener to indicate amount of energy gathered). Gathered healing energy fades and can dissipate entirely if not used within a short time.
3. As long as some healing energy remains, pointing the wand near an injured character causes faint green glow trails to extend toward that character. Holding down the controller trigger while pointing the wand at an injured character directs green healing energy into that character.

- The same breath of life can be used to heal multiple characters in succession if only minor healing is needed for each
- Multiple breaths may be necessary to heal very large or grievously-injured characters
- Need to test microphone volume sensitivity -- does a blowing breath create enough sound to register?
- Because my proposed implementation simply tests for a sustained minimum level of microphone volume, there's nothing to stop players from using other noises such as humming, talking, singing, yelling, etc to perform this spell
- Good idea to test early iterations to see whether concept plays well with target audience
- Alternative posibility: instead of having two different types of energy (projectile energy gathered from fast wand movements and healing energy gathered from the player's own breath) there could be a single type of energy that, once gathered, can be used either as a projectile or for healing depending on whether it is slingshot or directed into an injured character.

Runic break-in

...a spell in which a magically locked door is opened...

1. Player uses magic missiles to overload the locking mechanism on the door with magical energy



2. When mechanism is overloaded, a set of nodes and edges lights up
3. Tracing the illuminated graph using an Eulerian path (a trail which visits each edge exactly once -- no retracing a line) unlocks the door.

- Different doors present different Eulerian puzzles
- Can ramp up difficulty with easier/harder puzzles -- the most challenging doors could require completion of multiple puzzles in succession
- Many puzzles have multiple solutions
- No speed pressure
- Could provide hint system -- player can summon a helpful creature for advice or to reduce puzzle difficulty

Spell priority/disambiguation

Projectile-throwing is highest-priority spell— when the player is holding a wand the game is constantly monitoring wand-tip speed, and shows projectile-energy-gathering feedback whenever and as soon as the wand tip moves quickly enough to meet the projectile-energy-gathering speed threshold. If a player attempts to perform healing, door-unlocking, or summoning actions at a speed that exceeds the projectile-energy-gathering threshold, they will gather projectile energy instead of their intended action (the threshold should be set high enough that this possibility would be extremely rare). If the wand currently has projectile energy stored, this energy must be released before another magical action can be performed. (Projectile energy can be quickly released by clicking the wand-controller trigger button)

Second-highest priority is healing—when the player is holding a wand and the wand has no projectile energy gathered, the game is constantly monitoring whether the wand tip collides with the invisible solid positioned in front of the player's mouth. As soon as it does, a hovering green spark is shown, and any microphone input that exceeds the volume threshold while the wand tip stays in collision results in immediate healing-energy-gathering feedback. If a player attempts to initiate a door-unlocking or summoning action while their wand tip is in collision with the invisible solid, the action will not work (the collision solid should be sized such that this possibility would be extremely rare). If the wand currently has healing energy stored, this energy must be released before another magical action can be performed. (Healing energy can be quickly released by holding down the wand-hand trigger button while the wand is not pointed at an injured character.)

Third-highest priority is door-unlocking—when the player is holding a wand, the wand has no projectile or healing energy gathered, and the wand is pointed at a door with an overloaded locking mechanism, the game is constantly monitoring for collisions with gemstone graph nodes, and relevant feedback is immediately shown. If a player attempts to draw a portal while the wand is pointing in the direction of an overloaded locking mechanism, the movements will be interpreted as door-unlocking movements instead.

Fourth-highest priority is summoning—when the player is holding a wand, the wand has no projectile or healing energy gathered, and the wand is not pointed at an overloaded locking mechanism, the game is constantly monitoring whether the wand-controller trigger button is being held down. Whenever the trigger is held down and the wand tip is moved through space, portal-edge-creating feedback is immediately shown.

Proposed implementations

Summoning portal pseudocode

IF player holding wand, wand has no projectile or healing energy stored, wand is not pointed at an overloaded locking mechanism, and player is holding down wand-controller trigger, then…

- DRAW PORTAL EDGE
    - COLLECTPOINTSAMPLE
    - GUESS/UPDATE INNER SIDE (using center of mass of sampled points)

    - DISPLAY VISUAL TREATMENT
    - ABORT (if trigger is released before completion)
    - DETECT COMPLETION (as soon as line overlaps)
    - FORM PORTAL if closed shape is a plane shape (determine by standard deviation of points from least squares planar fit) & is within size range (determine by minimum radius – distance from center of mass) … otherwise…
    - COLLAPSE PORTAL
- DRAW OUT CREATURE (based on intersection, angle & direction of second stroke)

Magic missile pseudocode

Variables
storedProjectileEnergy
minimumThresholdSpeed
gatherRate
projectedDistance
minFiringDistance

WHILE *storedProjectileEnergy* is not zero
    Display visual/audio/haptic feedback based on current amount of *storedProjectileEnergy*
Get wand-tip speed
IF wand-tip speed exceeds *minimumThresholdSpeed* && wand-controller trigger is not pressed
    Increase *storedProjectileEnergy* by *gatherRate*
IF wand-controller trigger is pressed // probably actually an onEvent and not an if
    Display aiming feedback
    IF trigger is released // probably actually an onEvent and not an if
        IF *projectedDistance* is greater or equal to *minFiringDistance*
            Fire projectile
        ELSE
            Set *storedProjectileEnergy to 0*

Breath of life pseudocode

Variables
storedHealingEnergy
minimumThresholdVolume
decaying
gatherRate
decayRate
healingTarget = null
healRate
consumeRate
discardRate

IF wand-tip enters mouth collision area (must not have stored projectile energy or be exceeding speed threshold)
    Display green spark
    Check HMD microphone for *minimumThresholdVolume*
        IF microphone sound exceeds *minimumThresholdVolume*
            Set *decaying* to false

Increase *storedHealingEnergy* by *gatherRate*
ELSE set *decaying* to true
WHILE *storedHealingEnergy* is not zero and *decaying* is true
Display visual/audio/haptic feedback based on current amount of *storedHealingEnergy*
Decrease *storedHealingEnergy* by *decayRate*
IF wand is pointed near an injured creature
Display appropriate feedback
Let *healingTarget* be the creature the wand is pointed near
ELSE
Let *healingTarget* be null
WHILE wand-controller trigger is pressed
IF *healingTarget* is not null
Increase creature HP by *healRate*
IF creature HP is equal to creature's max HP, set *healingTarget* to null
Decrease *storedHealingEnergy* by *consumeRate*
ELSE
Decrease *storedHealingEnergy by discardRate*

Runic break-in pseudocode

Variables:
energyAbsorbed
energyOverloadThreshold
doorPuzzle
allPuzzleNodes is a list
allPuzzleEdges is a list
lastNodeTouched = null
currentNode
touchedEdges is a list
currentEdge

Door detects the amount of energy it has absorbed from projectile hits
With each projectile hit, check whether *energyAbsorbed* is greater or equal to *energyOverloadThreshold*
When this happens, load & display the puzzle for that door (illuminate and activate trigger detection for each node in *allPuzzleNodes*)

Detect onTriggerEnter between wand-tip and nodes
IF wand-tip contacts a node
Set *currentNode* to the node that was just contacted
Change *currentNode*'s illumination color to *touchedColor*
IF *lastNodeTouched* is not null
Let *currentEdge* be the edge between *currentNode* and *lastNodeTouched*
IF *currentEdge* is already in the *touchedEdges* list
Change all nodes and edges back to *untouchedColor*
Empty the *touchedEdges* list
ELSE
Add *currentEdge* to *touchedEdges* list
Change *currentEdge*'s illumination color to *touchedColor*
IF *touchedEdges* contains every edge in *allPuzzleEdges*
Puzzle is solved! Display feedback and open the door
ELSE Set *lastNodeTouched* to *currentNode*

Questions for discussion

1. How are the four spells differentiated/prioritized?
2. How has the intended audience been taken into account? (Can specifically consider issues of accessibility, different levels of mobility, and learnability)
3. How do these spell concepts reflect the goals/design pillars of the experience?
4. Does the pseudocode match the level of detail you were hoping for? (If not, let's discuss the proposed implentation of the spells in more depth.)
5. What potential issues do we anticipate with these spells, and how might they be mitigated?
6. How might we go about prototyping and playtesting these spells with our target audience?
7. Do these spell concepts and proposed implementations meet with your expectations? Did anything surprise you?

Outside resources

Works consulted

Gesture detection

- VR Infinite Gesture plug-in on Unity Asset store
- Adaptive Gesture Recognition with Variation Estimation for Interactive Systems
- $1 Unistroke Recognizer (and family)

Make shape without retracing line

- Math.stackexchange.com -- Is it possible to draw this picture without lifting the pen?
- CSE 326: Data Structures lecture 17 Really, Really Hard Problems
- Eulerian Path - Wikipedia

Pseudocode

- wikiHow – How to Write Pseudocode
- $P Point-Cloud Gesture Recognizer Pseudocode
- Protractor Pseudocode
- A Brush Stroke Synthesis Toolbox

Other inspirations & references

- Black & White
- The World Ends With You
- WarioWare: Touched!
- Fantastic Beasts VR app (haven't played personally but read Stephanie Hawn's UX Breakdown – Google Daydream VR – Fantastic Beasts not too long ago)
- VRGONAUTS gestural hotkeys