JENGAGE - GAME DESIGN DOCUMENT

Author: Oguz Guvenc
Date: 9/30/2023

1. Game Information

- Jenga Types
    - Glass: Not yet learned
    - Wood: Not yet tested
    - Stone: Mastered
- Game Modes
    - "Test my Stack"
        - Eliminates all Glass blocks
        - Activates the laws of physics
        - Causes the stack to topple over if they have many gaps.
    - "Strengthen my Stack"
        - Test the student on a random Wood block in the stack
        - Check if they really know it by assigning them a quiz.
    - "Earthquake"
        - Activate physics
        - Shake the ground to determine if the stack can withstand it.
    - “Build my stack”

        - Predicts how high their stack can be, when we start adding blocks from future grade levels on a shaky foundation.
    - “Challenge”
        - A multiplayer game that allows to challenge other students on pieces in their stack.
- Unity Version
    - 2021 or 2022

Requirements

Build a 3D Jenga game based on the above concepts and the following requirements:

1. Use this API to fetch data about the Stacks (there are 3 stacks for 3 different grade levels)
2. Put 3 stacks on a table in a row
    - Use data from the API response to determine the block type:
        - mastery = 0 → Glass
        - mastery = 1 → Wood
        - mastery = 2 → Stone
    - Put a label in front of each stack showing the grade associated with it (e.g., 8th Grade)
    - Order the blocks in the stack starting from the bottom up, by domain name ascending, then by cluster name ascending, then by standard ID ascending
3. Enable orbit controls on Mouse 1 hold so that the user can rotate the camera around the stack
    - One of the 3 stacks should always be in focus by default
    - Add some control to switch the view between the 3 stacks
4. Add an option to get more details about a specific block (e.g., on right-click)
    - Show the following details available in the API response
        - [Grade level]: [Domain]
        - [Cluster]
        - [Standard ID]: [Standard Description]
5. Implement the “Test my Stack” game mode described above, where you remove Glass blocks in the selected stack and enable physics

Please, feel free to check the below screenshot to get a better understanding of how this might look:

You will be evaluated based on your ability to create a proper architecture and provide a working solution that's pleasing to the eye that meets requirements.

2. Planning

My Questions

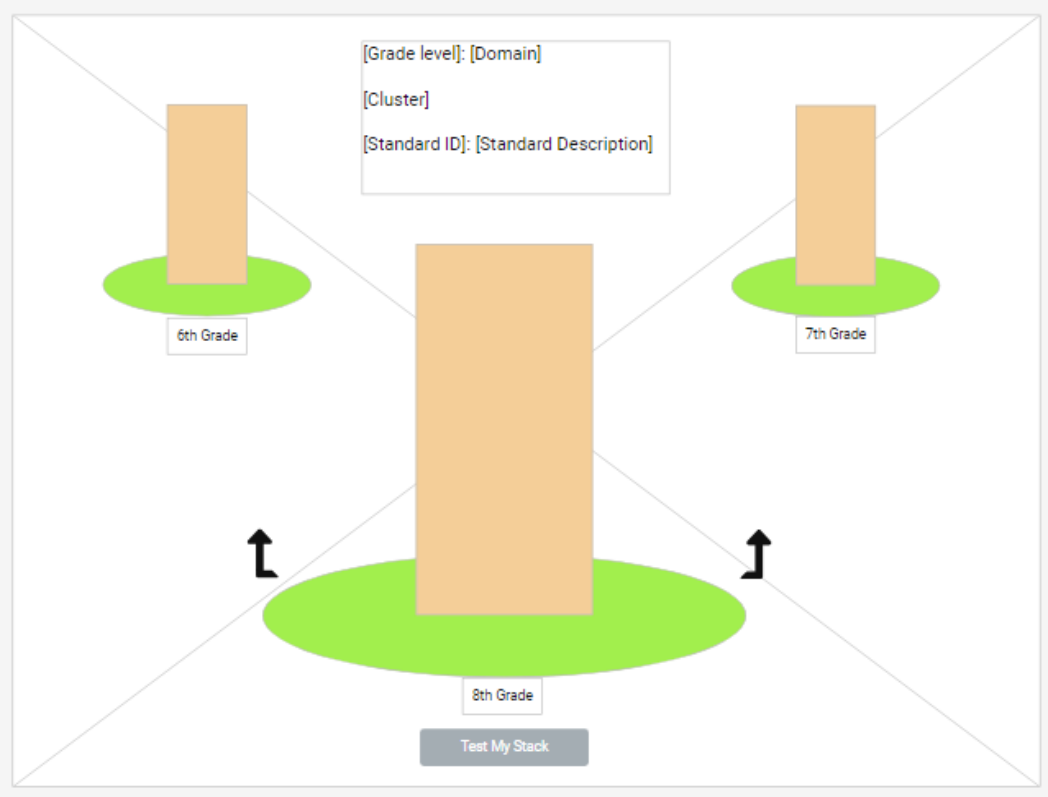
1. What happens if a stack collapses when the glasses are removed?
    a. Does that show the student that their knowledge is insufficient?
    b. If the three stacks are next to each other, will they affect each other?
2. Is there an deterministic way to structure which combination of missing jenga pieces cause collapse? (center of mass vs base of missing layers?)
3. Aren't 6,7,8 stacks supposed to be stacked together for an 8th grade student?
4. Is all the information in the API related to one student's current skill representation?
5. Should "Test My Stack" test all three stacks at the same time? Or should I have a separate button for each one?

Tasks Overview

1. Get the data from the REMOTE API
2. Write the data to local storage, as a dictionary
3. Reorgane the information based on the ranking requirements
4. Build the stacks
    a. Create a block template that will hold information on the following faces
        i. Small face = domain name
        ii. Middle face = cluster name
        iii. Large face = standard ID

    b. Run a BuildStack method for each of the three stacks that will
        i. Use the ordered list of blocks to fill each layer with three blocks, if available
        ii. Determine if a block will use glass, wood or stone block based on mastery
5. Test the physics
6. Reset the board
7. Information modal pop up when block is right clicked

UI Overview

- StudentName
- "Test my Stack" Button
    - If pressed, it will show:
        - A 5 second timer for the test
        - If success:
            - "You got this! Keep building" text shown
        - If failure
            - "Time to strengthen your base!"
        - "Reset Stacks" button
- "Exit Game" Button
- Mockup
    - 

Assets

- Glass, wood, stone PBR materials
- Classroom environment
- Buttons for UI

3D Design & Art Style

- Add three floating islands, each one holding a grade level stack, and the user can rotate between the three
-

Bonus

- Add a mode to add all three stacks on top of each other, or mix & match consecutive grades (i.e. 6&7, 7&8, or 6&7&8)
- When "Test My Stack" is pressed, the glass should be shattered

3. Architecture

Big PictureOverview

- MVCS
- Frontend
    - Unity build for Web
    - A method to request information from the server based on a specific student
    - When a student passes a quiz, send a PUT to the backend to change student record in the database
- Database
    - Hold information about students, and their related masteries in different standards
- Backend
    - A "GetUserData" service that will fetch the data from the database and pass to the frontend

Frontend (Unity) Design

- Classes
    - LevelManager
        - Function: Handles the start, exit and level change (if any) logic

        - Methods:
            - LoadLevel()
            - ExitGame()
    - DataManager
        - Function: Fetches data from backend and organizes data for use
        - Methods:
            - FetchData()
            - OrganizeData()
                - Order data based on domain>cluster>standard
        - Variables
            - *private dictionary* RawUserData
            - *private dictionary* OrganizedUserData (set/get)
            - *private dictionary* UserActionHistory
    - StackManager
        - Function: Facilitates the building and re-building of the stacks
        - Methods:
            - GetUserInformation()
            - BuildBlock()
            - BuildStack()
            - RebuildStack()
        - Variables:
            - int[] BlockSize
                - x, y, z sizes of each block
            - material[] BlockMaterial
                - Materials for glass, wood, stone blocks
            - physicsMaterial BlockPhysicsMaterial
                - Determine drag properties of the blocks
            - int RowSize
                - Number of blocks to be added per each layer
            - int AngleVariation
                - The amount of change in angle between layers (90 by default)
            - bool FavorStability
                - Arrange blocks to have glasses in center if possible
    - TestManager
        - Function: Handles the "Test My Tower" button functionality
        - Methods:
            - DestroyGlassBlocks()
            - ActivateGravity()
            - StartTestCountdown()
            - CheckSuccess()
                - Checks if any blocks touched the floor
    - AudioManager
        - Function: Facilitates all audio events

- InputManager
  - Function: Controls the movement of the camera around a block, and the rotation of the blocks
  - Methods
    - MouseRotate()
    - SwitchStackView()
      - Moves between the grade level stacks
- Tooltip Manager
  - Function: Provides information related to the selected block
  - Methods:
    - CheckForBlock()
      - Raycast to the mouse point, if ray hits a block, then call FetchBlockInformation
    - FetchBlockInformation()
      - Fetch information regarding the chosen block and call DisplayBlockInformation
    - DisplayBlockInformation()
      - Show the fetched information in UI
  - Variables:
    - string BlockInformation
- ResolutionManager
  - Function: Adjusts the scene resolution based on Web interface specs
  - Methods:
    - UpdateResolution()