My idea

Use this to summarize your idea, plan it using sketches, notes and pseudocode as needed

|  |
| --- |
| My GitHub Username: EternalNyaki  Idea: Rhythm game built for the arcade cabinets, with notes coming from the edges of the screen and moving towards the center. Notes must be hit when they reach a ring at the center of the screen. Notes are hit by pointing the joystick towards them and pressing any button.  A paper with writing on it  Description automatically generatedA white paper with writing on it  Description automatically generatedA white paper with writing on it  Description automatically generated  A diagram of a game  Description automatically generatedA grid paper with black text  Description automatically generated with medium confidenceA screen shot of a screen  Description automatically generatedA paper with lines and a circle in center  Description automatically generatedA graph paper with a letter and numbers  Description automatically generated  Pseudocode:  Declare variables and objects (10, 28, 29, 30, 33, 34)  Setup (3, 4):  Load chart JSONs  Draw loop:  Clear screen (5)  If on start menu (12, 15):   * Draw start menu (1, 2, 20) * Update start menu (6, 11, 17)   If on options menu:   * Draw options menu * Update options menu   If on song select screen:   * Draw song select screen * Update song select menu   If playing song:   * Update notes based on chart (21, 23, 24) * Update joystick position indicator based on joystick input * If any button pressed:   + If joystick pointing towards note and note is close to indicator (12, 14, 40):     - Update score     - Update accuracy     - Delete note     - Play hit sfx   + If note passed indicator:     - Update accuracy     - Delete note     - Play miss sfx * Draw score * Draw song title and artist * Draw background gradient   If on results screen:   * Draw rank icon based on accuracy * Draw score * Draw accuracy * Create confetti if score is above S (39, 41) * Draw background gradient |

Where will the inventory skills be demonstrated? List every one to be sure you’ve included them.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Primitive shapes will be used to draw most things in the game 2. Will be used pretty much whenever I’m drawing something to the screen 3. rectMode will be set to CENTER and ellipseMode will be set to RADIUS in setup() 4. I don’t think it’s possible to create a functional game in processing without using these 5. background() will be used to clear the screen every frame 6. constrain() will be used to limit the game state and limit button selection for menu navigation 7. keyPressed() will be used in menus to activate buttons and in game to hit notes 8. I’ll be using these all over the place, especially in for loops 9. In every single one of my classes 10. I’ll be using variables for damn near everything 11. I’ll need to debug stuff at some point, I’ll leave it in when I use it 12. I’ll be using these everywhere 13. … and these 14. …… aaaaand these 15. For handling game states 16. Will also be using for loops everywhere 17. For handling menus, as buttons will be stored in a 2D array that will need to be navigated with two nested for loops 18. To only handle the notes that would be on screen 19. A for loop is just a while loop that integrates an incrementor directly into its structure, making it more efficient but less versatile 20. Used for drawing objects 21. I will definitely be using my custom modulo function 22. Parameters are the values the function requires, whereas arguments are the actual values passed into the function when it’s run 23. Will at minimum be in my custom modulo function 24. The function for parsing the chart JSONs into chart objects will take a JSONObject as a parameter 25. A class is like the “blueprint”, with an object being an actual instance and implementation of the class 26. A constructor function creates an object based off a class with the given values as arguments, and is run when an object is created 27. Keeps things organized 28. I’m gonna have a lot of these, don’t worry 29. Also gonna have a lot of these 30. And a lot of these 31. The size of an array is set when it is initialized, and it can only hold one type of object, whereas the size of an ArrayList can be changed at any time during the program’s runtime 32. When you want to shift everything in an ArrayList forward to add something new at the first index 33. Will be used for the buttons in a menu 34. Will be used for the currently on-screen notes when playing a song 35. Will be used for the currently on-screen notes when playing a song 36. Will be used for the currently on-screen notes when playing a song 37. Whenever you’re working with something that can be represented as a point or a vector 38. Will be used for the positions of damn near everything 39. Will be used for the confetti effect that displays on getting an S rank or higher on a song 40. Will be used for checking if a note was hit on time 41. Will be used for the confetti effect 42. Normalized vectors are vectors with a magnitude of 1, which makes them helpful when you only care about their direction 43. I’ll use fromAngle() for the velocities of the confetti | | | |
| **Milestone 1** | **Milestone 2** | **Milestone 3** | **Milestone 4** |
| What will I deliver?   * Base gameplay with a simple chart * Basic start and results screens | * Charting tool (made in unity, I know I won’t be getting any marks for it) * Chart files for every song | * Menus * Confetti * Polish |  |
| Which inventory skills will this demonstrate? List them. |  |  |  |
| line, ellipse, rect, triangle, quad, arc, curve (1) |  | constrain(), dist() (6) |  |
| fill, stroke, strokeWeight, noFill, noStroke, color (2) |  | println(), stop() (11) |  |
| Modes: CORNER, CORNERS, CENTER, RADIUS (3) |  | switch statement (15) |  |
| setup(), draw() (4) |  | A nested loop (17) |  |
| background(), random(), noise() (5) |  | Do some basic physics: use position, velocity, and acceleration (due to gravity) vectors (39) |  |
| keyPressed(), keyReleased(), keyPressed, mousePressed(), mousePressed (7) |  | Create a random 2D vector (41) |  |
| Increment operators: ++, +=, --, -=, \*=, /= (8) |  | Using the Processing documentation look up a method in the PVector class that’s new to you and use it in your code (43) |  |
| Declare and use a local variable (9) |  |  |  |
| Declare and use a global variable (10) |  |  |  |
| Conditional statements: if, if else, else (12) |  |  |  |
| Boolean expressions: ==, >=, <=, >, <, != (13) |  |  |  |
| Logical operators: &&, || (14) |  |  |  |
| for loop, while loop (16) |  |  |  |
| break() (18) |  |  |  |
| Declare and call a function with no parameters and no return type (20) |  |  |  |
| Declare and call a function with a return type (21) |  |  |  |
| Pass by copy (value): declare and use a function that takes int,  float, char, etc as an argument (23) |  |  |  |
| Pass by reference (objects): declare and use a function that  takes an object as an argument (24) |  |  |  |
| Write a class with a constructor function (28) |  |  |  |
| Use the keyword new to instantiate an object (29) |  |  |  |
| Write a constructor function with parameters (30) |  |  |  |
| Initialize and populate an array (33) |  |  |  |
| Initialize and populate an ArrayList (34) |  |  |  |
| Manage a set of objects with an array or ArrayList (35) |  |  |  |
| Use an ArrayList method: size(), get(), remove(), contains() (36) |  |  |  |
| Use the PVector class (38) |  |  |  |
| Find the direction and distance between two points (40) |  |  |  |
| You should deliver approx. 10 skills at this milestone | You should deliver approx. 10 skills at this milestone | **You must deliver 30 inventory skills by this milestone.** |  |