|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Jeffrey Zhang | Team | Spaghetti Studio | TL | 4 | Date | 4/13/2025 | Time | 0331 |

Fill in the underlined areas (and the boxes above), now but don’t write on the remainder of this form.

|  |  |
| --- | --- |
| **Contribution:** Briefly describe what your feature(s) is/are:  **\_\_\_\_\_\_**I did the Sound Design and Background Assets, as well as all the ingredient  **\_\_\_\_\_\_\_\_\_\_\_\_**sprites.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Walk me through your Gantt chart. How long did this take? How long did you estimate it would take? What did you learn about your skill as an estimator?  *My skill as an estimator is quite poor, in addition it doesn’t help that I’ve never worked with Gantt and Pert charts before attending this course. Being a TL4, the Gantt Chart itself took several hours to make, and making sure that everyone’s matched up initially was a little difficult.*  Run your game and point out places where your code is called and run. (I will cycle through asking you this question and the next one until you either run out of interesting things to talk about or it is clear that you have made an above average contribution.)  Show the C++/C# code that was run. Walk me through the methods called from the time it enters your section of code.  This wasn’t required but I made a “Game Flow Call Graph” for ease of interpretation:    For the UI Button Clicking: | /10 |
| **Technical:**  Walk me through your test plan. Give an example where a test case later found a bug in your code by things a teammate added later. (Or explain why you chose a test case specifically because you wanted to ensure that a teammate would know if they broke your code.)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Pick a Prefab you have created that is documented well in a separate readme file.  (I will point to several places in your code documentation and ask) What question where you trying to answer here? Who do you anticipate would be asking that question? What other questions might this person need the answers to?  Prefab Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Show me a class in your code where there could be either static or dynamic binding. Write some mock code on this paper showing how you would set the static type and dynamic type of a variable.  Super Class: \_\_\_\_\_\_\_\_\_\_\_\_\_AudioControlBase\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Sub Class: \_\_\_\_\_\_\_\_\_PlayBackgroundMusic, PauseWithSFX\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Virtual Function: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_TogglePause()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *Mock Code:*  *AudioControlBase controller; // Static type*  *controller = new PlayBackgroundMusic(); // Dynamic type*  *controller.TogglePause(); // Dynamically bound (if virtual)*  Choose a dynamically bound method. What method gets called now?   * *‘TogglePause() in PlayBackgroundMusic*   Change the dynamic type. What method gets called now?  *controller = new PauseWithSFX();*  *controller.TogglePause();*   * *TogglePause() in PauseWithSFX*   Pick a statically bound method. Which one would be called in each of the two previous cases?  *If TogglePause() were* ***not virtual****, the method called would always be the one defined in AudioControlBase,* ***regardless*** *of the dynamic type.*  Show me an example of reuse in your code where you violate copyright law.  How does it violate copyright?\_\_\_\_\_\_\_\_\_ You didn’t obtain a license or fair-use permission to use the audio in a distributed or commercial product.\_\_\_\_\_\_\_\_\_  What did you have to do to integrate it with the code you wrote? What are the legal implications if you market your code with the re-used portion? Use fair use argue that you can use this anyway.  *I didn’t obtain a license or fair-use permission to use an audio such as ‘tape rewind.mp3’ for the pause menu in a distributed or commercial product.*  *pauseSFX.clip = Resources.Load<AudioClip>("pause"); // Assume "pause" is copyrighted*  *pauseSFX.Play();*  *If I distribute or sell our game, the owner of the audio clip could issue a takedown, sue for damages, or force licensing fees. However, an argument for fair-use permission is that the mp3 was used for educational/demo purposes, and part of a “Transformative, original gameplay context”.*  4. One big or two small, well-chosen patterns.  Small Patterns = {Singleton, Private Class Data}  Which patterns did you choose?  1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Singleton\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Private Class Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Why did you choose each pattern? (Justify your use of it).  ***Singleton:*** *To ensure only one global background music controller exists and persists across scenes.*  ***Private Class Data:*** *To encapsulate audio state flags like isPaused, so external scripts don’t accidentally modify it.*  Draw the class diagram for your pattern(s).  +----------------------------------+  | PlayBackgroundMusic |  |----------------------------------|  | - backgroundAudio |  | - isPaused |  |----------------------------------|  | + Start() |  | + Update() |  | + TogglePause() |  +----------------------------------+  ▲  |  +----------------------------------+  | Singleton (Static Ref) |  +----------------------------------+  Would something else have worked as well or better than this pattern? When would be a bad time to use this pattern?  ***Singletons*** *are generally bad in large projects where global access creates hidden dependencies. In addition, Singletons could be replaced with* ***Dependency Injection*** *for testable and scalable systems.*  ***Private Class Data*** *is fine here in my opinion, however if overused, it might make internal logic too rigid.* | /4  /3  /3  /4  /4 |