|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Todd Carter | Team | Green Sky Games | TL | 2 | Date | 4-13-2025 | Time |  |

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:  **NPCs combat, NPC dialogue, managing NPC interactions with player, writing storyline\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  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 gantt chart begins with establishing requirements, which did take a while. Then producing the NPC Manager, Game Objects Manager, and NPC superclass all took an extended time and changed definition as time went. These set up for the NPC dialogue system, which took a huge amount of time due to the need to have a branching, choice-based story. Once the combat system and dialogue system were complete, there was only writing dialogue itself left, which came with programming the occasional custom response from an NPC.  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. | /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.)  \_Our TL3 was unable to get the Unity test runner assembly operational for us.\_As a result, we have somewhat ad-hoc tests that are run in-game using buttons. The code for these tests has been attached. These test cases technically met the requirements of the boundary testing and stress testing, but that is all.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  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: \_The Wolf\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  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: \_NPC\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Sub Class: \_setNPC\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Virtual Function: \_setNonHostile\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Choose a dynamically bound method. What method gets called now?  Change the dynamic type. What method gets called now?  Pick a statically bound method. Which one would be called in each of the two previous cases?  Show me an example of reuse in your code where you violate copyright law.  How does it violate copyright?\_\_I took sprites from a source that requested I credit them, and I did not do so.\_  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 utilized the sprites for most of the human and humanoid NPCs. If this was marketed, the creators of the sprites may have grounds to sue. I would argue, however, that their request was not specifically legally binding given the public nature of their request for crediting.  4. One big or two small, well-chosen patterns.  Small Patterns = {Singleton, Private Class Data}  Which patterns did you choose?  1.\_\_Singleton Pattern\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  2.\_\_Decorator Pattern\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Why did you choose each pattern? (Justify your use of it).  I used a singleton pattern to create an instanced version of a DialogueManager that could then be called more easily by objects in the scene, such as the NPCs.  I used a decorator pattern to create the nested canvases which make up the dialogue box that is called by interacting with the NPCs.  Draw the class diagram for your pattern(s).    Would something else have worked as well or better than this pattern? When would be a bad time to use this pattern?  In these circumstances, the singleton pattern was ultimately unnecessary as there is only one instance of the DialogueManager anyway.  As this creates global variables, it is bad to overuse singleton patterns as they can cause conflicts with naming and other problems that arise with global usages. | /4  /3  /3  /4  /4 |