# **CSD2451 – INDIVIDUAL MILESTONE REPORT – V01**

**Every student** must submit a report at each milestone detailing all the work they have done for the project module, during, **only, the current milestone**. This report is submitted through a submission link in Moodle (https://distance3.sg.digipen.edu).

**Subject Line**

The file name must be in the following format:

“**Mx\_TeamName\_LastName \_FirstName \_IMR.docx**”. Replace **x** with a number between [1 … 6]

* Incorrectly formatted files will be rejected, with a high penalty!
* You can only replace the default **blue** text(s) below. Penalties apply otherwise.
* The text(s) in **blue** color below is a sample for your reference, and you should remove it. You must change it based on your position and contribution. If you have nothing to add in a section, you must write “None”.

**Contents**

The contents of this report must start with the following (in this order, with all caps for the section titles). Anything in italics here must be changed to whatever is appropriate for you (but should not be in italics in the actual submission you send).

**MILESTONE**: *Alpha*

**TEAM NAME**: *Anarchy Forge*

**GAME NAME**: *RiccoShaman*

**JOBS/CHAMPIONING**: *Product Manager, Game Designer, Gameplay programmer*

The next section contains a detailed listing of all the code you wrote this milestone. For every source file you wrote code in, state the number of lines of real code you wrote (within +/- 10 lines, you can use LOC to assist with this), the file name, and then a description of what the code you wrote does (this can be much longer than what is listed in the example below). Note that “lines of code” does not include comments, empty lines, code broken into multiple lines to inflate the count, repetitive cut-and-paste code, etc. Quality and difficulty of code counts more than quantity, so don’t just focus on writing lots of lines. Give us the TOTAL LOC at the end. Use a tool like **CLOC** to count the lines of code: “http://cloc.sourceforge.net/”.

Add the scripted files you’ve worked on. Example: shader files, .bat files, .lua files or C# files…

***CODE:***

|  |  |  |  |
| --- | --- | --- | --- |
| **File** | **Lines of code** | **Describe your current work added/ updated in the file** | **Reference (if any)** |
| DataCenter.h/cpp | 108 + 1178 | Remove and refactor Serialization & Deserialization from Scenemanager.cpp and into its own fille. Also updated system to accommodate for all new components created |  |
| *Dialogue.h* | 27 | *Created dialogue component for dialogue scene* |  |
| DialogueSystem.h | 80 | *Created dialogue system for dialogue scene behavior* |  |
| Cutscene.h | 26 | *Created* Cutscene *component for* Cutscene *scene* |  |
| CutsceneManager.h | 83 | *Created* Cutscene *system for* Cutscene *scene behavior* |  |
| Editor\_Inspector.h/cpp | 400 | Added ImGUI inspector stats for dialogue and cutscene. Also updated inspector for player and baseenemy component |  |
| Random.cpp | 8 | Added math function(will change name of script later. Sorry) |  |
| CameraSystem.h | 16 | Added camera shake function. |  |

***TOTAL LOC: 1916***

The next section contains a detailed listing of all the prototypes you made this milestone. Prototypes can be for gameplay, content, level design, or systems. Your prototypes can be made in any engine, although it makes sense to use your game's engine once it is available. For every prototype, state how many different versions you made and a description of the different versions. You must also have tested your prototypes, at the very least among your team members. Quality and difficulty of your prototypes counts more than quantity, so don’t just focus on making lots of very simple prototypes.

**Custom Engine – Cutscene:**

*Added dialogue and cutscene manager that is used to create the Intro scene, Intro dialogue scene and the*

*outro scene*

This section contains everything you did for this milestone that was not actual making of gameplay. This includes debugging, optimizing, testing, running playtest sessions, planning, meetings, designing, creating, or finding art, creating, or finding audio, preparing presentations, helping teammates, helping other teams, helping the instructors, buying food for the team, etc. Don’t worry if it “counts” or not—just include everything.

**Extras:**

*Help voice act in the game. I was the cat(meow)*

*Constantly communicate among designer, programmers, and artist to ensure no miscommunication or*

*misunderstandings occur.*

*Delegate and reorganize tasks to ensure everyone has something do. If a teammate is done with all his task but*

*another has multiple tasks. Some of his tasks will be for him to fasten production.*

*Help introduce the new member of the team on how the engine work and get him comfortable with the team.*

*Help balance gameplay.*

Finally, throw in anything else you think is relevant, including comments on how you think your teammates are doing (positive or negative), explanations for poor performance or absences, explanation of team changes, how the milestone/semester went, etc.

**Notes:**

*They were some choices I regretted making early in development in terms of delegation of task. If it was different, it could have made our production much smoother. However, I try not to dwell too much about the past and try to make the most of the current situation the team is in and accept it as part of the learning process.*