Onboarding Manual

**Agile**

* Agile is a software methodology where instead of gathering all the requirements upfront like waterfall, we set a few requirements in the beginning and add/remove/adjust them as we go along. We'll be using this style due to the fact that we ourselves are unsure of what we want in the game, so Agile exists for us to adjust to that. Essentially, we will be breaking our goal up into a series of sub-goals, and each sprint(2-3 week time periods) will be devoted towards accomplishing a small sub-set of those subgoals. Advantages of doing it like this is that we can adjust sub-goals to changing requirements if it turns out that we want to change the game up to do things differently, it's also more productive to have a series of sub-goals rather than have one large goal.
* Would be beneficial to have a demo for each significant milestone accomplish, to ensure everyone is aware and satisfied with the current product and discuss if there needs to be any changes before we move on.
* Weekly meetings should also be held regularly, to check in on everyone's progress for a sprint as well as hold discussion for important matters that otherwise, cannot be done through messaging.
* During Sprint Planning, we will look ahead into the next few weeks and set goals for ourselves, making estimates on what we can accomplish in that time with the time available to each of us. One should also factor in what their amount of available time is. It is okay to over-commit slightly as if we end up over-estimating the amount of time it takes to finish our assigned work, we still have other tasks we can do in preparation for next sprint.
* Sprints are essentially 2-3 week time frames where we spend to try and accomplish the goals we set out to do. A sprint will have a series of tasks assigned to it. All members should primarily focus on the tasks we have left to do and only those tasks. Exceptions are permitted in the situation of blockers. If a member's only work is currently depends on another member's work that is currently progress, they are permitted to take on other tasks not committed to the Sprint on their own but ONLY if they have no other work committed in the sprint to perform.
* At the end of each sprint, we should have a retrospective to reflect things we did well, things we did wrong, and how we should improve. The time should also be spent talking about what changes we should do. Re
* Talk about self-responsibility and initiative. There will be no leaders in the group. Have people take up their own tasks rather than have other people assign work to them. More motivating and makes people more involved. Ask questions if unsure of what to do. You can ask for help from others if you need it, talk to anyone who's work might correlate with what you're doing and collaborate with each other if needed but you alone are responsible to get the things you decided to do done. It is your responsibility to find something to do.
* Modified Scrum meetings: Using skype/facebook messages. Doesn't have to be daily. Only when people are actually doing work on that day do they report. . Report status regularly. Anything related to the project, no matter how small it is, should be notified to everyone involved. Things that should be reported includes:
  + Things you have done so far- It's good to know what your progress is so far so we know what's been accomplished.
  + Things you plan to do- So we know what to expect and no one's work overlaps each other.
  + Questions/Concerns you have: This could be about the stuff you're working on or the project as a whole but it's best to report any questions or concerns you have about it so we know and hopefully, someone can answer that.
  + Estimated time of next scrum update. Does not have to be when you're likely to finish your assigned work but rather when you can notify the group of your progress.

The contents don't matter. It can even be as small as simple brainstorming. So long as we know what's going on, that's all that really matters. It's better than being left in the dark and knowing about your involvement may motivate some others as well.

**Group Organization**

* Define the primary roles of each member of the group. Roles may shift depending on what tasks are needed to be done.
  + Programmers: Eddie, Johnson, Jerry
  + Writers: Gary, Johnson
  + Art: Wan
* Member Collaboration. While the group roles may seem separate by themselves, there will be times where people will be required to work with others with different roles. Examples include:
  + Programmers may need to work with artists/writers to ensure the scene transitions are good
  + Artists may work with writers to ensure the art matches the characters and scenes and vice versa
* Peer Reviews. Everyone in the group should have at least 2 people review their work. People are responsible for keeping track of their finished work until they found someone who officially agreed to review it for them, which then responsibility passes to the reviewer. Do not be half-hearted when conducting the reviews. If you have even the slightest critique, make it known but try to be constructive about it. Review Guidelines, organized by role.
  + Programmer: For Code Reviews, have it reviewed only by programmers. Quality Assurance, ensuring things like UI or scenes are in order, can be done by anyone.
  + Writer: Have one writer and one non-writer review your work. This will produce a wider range of feedback and having a non-writer review will help point it out any out-of-character moments as two writers may have two vastly different opinions so a third opinion will be beneficial.
  + Artist: Generally, similar to Writer.
* Quality Assurance. While the programmer's job is to implement the features of the game, everyone is responsible to ensuring the work is at the quality we want. Therefore, do not be half-hearted during reviews or during manual testing of the VN. Most of the testing may be done manually; therefore, anyone can take up the task. Be sure to take note of everything you observe during the test and report anything that may seem out of place.
* Ask Questions. It's extremely important to ask questions if there is something that you're even the slightest unsure about. Failing to do so could lead to a complete misunderstanding of what needs to be done and what is required, leading to a completely different design from what is desired. Worst case scenario, you'll be doing twice as much work than required by having to re-do all the work done as a result of misunderstood requirements when you could've gotten it right the first time through.
* Cooperation. Cooperation is important as some of our work may depend on the input of others. Everyone should always be willing to assist others when available and everyone should be aware when you're not available. Despite these seemingly disjointed roles, there will be many occasions where writers will have to work with artists, or artists work with programmers to get things done. We should also be able to contact each other regularly in case of emergencies. There is a taskboard which people can use to update their progress on their work so that others are aware. The task board should be checked regularly so that members understand what everyone else's work involves. Also, in the event that members will be offline or unable to work on the project for a significant amount of time, they should notify all the other members on the facebook group and if possible, provide a means of contact in case other members need them in a time of emergency. In Agile, it is not about how much you can do in your group compared to others, but how much can you accomplish together.
* Organization: Documentation and task management. Regardless of roles, everyone should pitch in and put in the effort to ensure the group stays organized and all important information gets documented so that it can be looked back on as a reference. Task management should also be a group effort, with each members taking the time to update the Product Backlog with tasks that we have yet to accomplish. All new tasks should be added to the Product Backlog, leaving it until further discussion and task breakdown is required. Everyone has their own specialties and as such, have their own understanding of what needs to be done for the project. We should take advantage of everyone's varying backgrounds to cover all the bases and add in tasks that we may or may not need to do. This applies to documentation