RC-Gamepad commit-structure README

Bitte versucht mir zuzuhören und mitzudenken, sonst dauert das alles ewig...

Vorher

1. Paul committed die folder structure und alle wichtigen Vorraussetzungen
2. Alle pullen sich das Projekt lokal in einen Ordner und öffnen diesen (im root folder) in VSCode
3. Jeder legt sich einen anderen Ordner an, in den er die heruntergeladenen files verwahrt
4. Jeder konfiguriert die GitLab credentials im VSCode Git plugin
5. Jeder konfiguriert sich eine sinnvolle Ansicht: Am besten zwei file explorer nebeneinander und in einem anderen workspace das vscode-fenster und den text editor öffnen

Sprint Ablauf

1. Eigenen Ordner aus Sprint 1 Ordner herunterladen, in den anderen Ordner kopieren
2. jede neue datei im Texteditor öffnen und den Inhalt in kleinen Schritten committen (mit sinnvoller Commit-Message: z. B. "added class messenger" oder "added this_is_a_function()", vllt. mittels auto commit message chatgpt) in folgender Reiheinfolge:
    1. Erst alle freistehenden Funktionen
    2. dann den Grundaufbau (z. B. klassen mit ihren Constructors)
    3. dann alle member functions der klassen für jede einen commit, währenddessen einen Error einbauen (z. B. eine Zeile weglassen oder einen Variablenname sinnvoll verändern)
    4. Error verbessern (commit message "fixed wrong variable name" oder so)
        1. EINMAL PULLEN, dann einen Bug-Report alegen nach dem Schema, das schon im GIT drin ist
    5. am ende einmal strg+a und strg+x im Editorfenster und strg+v im vscode Fenster und noch ein letztes mal commiten (wenn es noch changes gibt), die Commit message sollte hier sowas sein wie "beautifyed" oder "refactored"
3. Nachdem alle Ordner von allen heruntergeladen und committed wurden wird das Gesamtprojekt getestet
    1. Nachdem es wie zu erwarten nicht funktioniert, wird es so lange von Paul gedebugged bis es funktioniert, dafür werden alle Änderungen aufgeschrieben um sie später wieder rückgängig machen zu können (mit git revert)

4. Dokumentation des Sprints wird committed

Sprint nr.	Was?	Funktion am Ende
1	Simulation, Vehicle	Simulationsfenster öffnet sich (noch kein controller zum kontrollieren)
2	RC stuff	GUI funktioniert, Controller output ist da (debug print)
3	Messaging, encoding	Networking, logging, alles andere

Danach

1. Paul stellt Kompilierbarkeit nach den Wünschen der API Leute her, TODOs und FIXMEs werden entfernt / bearbeitet
2. Allgemeine Dokumentation, Anleitung wird erstellt, comitted (vorher darf Paul probelesen)
3. Evtl. auf Clara zum Laufen bringen und mit Bild / Video dokumentieren