Note Taker: All

- Find out how the classes relate
- Ask: Can we remove classes without arrows?
- Figure out the root nodes' relationship with one another (excluding the children?)
    - Identify useful stuff

Important classes defined as:

Classes in the link (direct classes + classes under the module)
https://matplotlib.org/3.1.1/api/index.html#usage-patterns

Tasks:
[Chris, Harrison, Vili]

- Fix inheritance for important classes (big nodes+classes in link)
- Place them into one of 3 layers
- Interaction between the important classes
- Also fix the giant uml

[Faris, Winston]

- Discuss process model + aspects of it and modifications
- Discuss pros/cons of our model and other models and why we chose ours

Plan driven-->waterfall
Agile→Kanban+scrum
Incremental agile & plan-driven

We are going with **waterfall**

Incremental:
Don't need it, since we won't have changing requirements. Don't need to release/build in increments

Plan driven:
Closer match to our needs, since given the short time frame, we would spend some time at the start planning, then implementation, then tests maybe, then release

Agile:
Do not need the tools that agile offers, they don't help us

Reuse-oriented:
We don't have anything to reuse since we will be fixing bugs initially.

Waterfall:

Requirements won't change, will not be releasing in increments, not enough time for multiple sprints

Mods: don't need maintenance phase, requirements phase not needed for bugs, could be needed for feature, verification phase to ensure code quality so it gets merged

Must fix inheritance arrows for the +5 marks (third bullet point), *at the very least, fix arrows for important classes*

Important classes

- AbstractMovieWriter
- AbstractPathEffect
- Node
- TransformNode
- _GeoTransform
- OffsetBox
- _Base
- Animation
- Patch
- Collection
- DateLocator
- TickHelper
- _ImageBase
- Widget
- ToolBase
- Fonts
- _Base
- Container
- BaseFilter
- BlockingInput
- Normalize
- PdfPages
- RendererPDFPSBase
- HatchPatternBase
- Event
- ColorbarBase
- FontConstantsBase
- TriInterpolator
- _Backend*very big
- ScaleBase
- HandlerBase
- MathtestBackend
- PolarAxes
- _DOF_estimator
- Artist

Useful links

https://helpmanual.io/help/pyreverse/

https://creately.com/diagram/example/hqz3w6j12/matplotlib%203-tiered%20architecture

https://matplotlib.org/3.1.1/api/index.html#usage-patterns