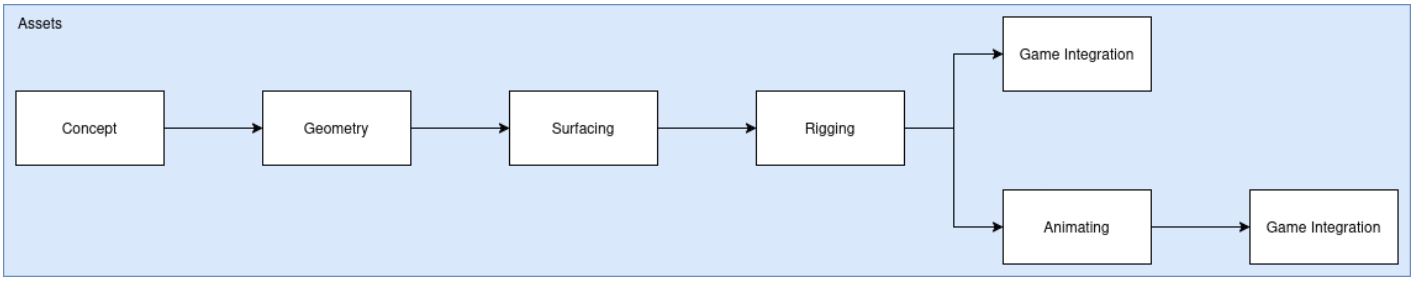
Design

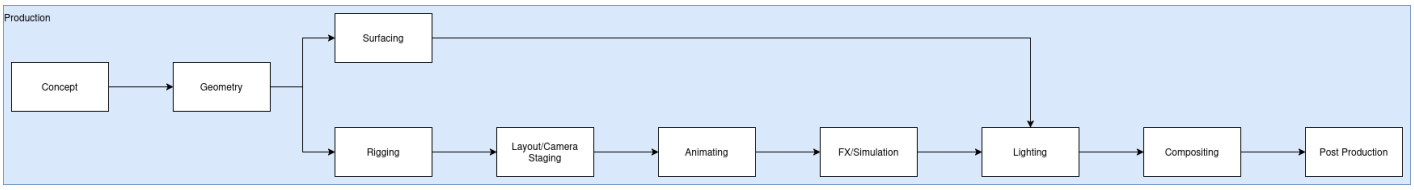
The goal of the pipeline is to support the production team working on the BUGJam. While we will not be able to have a full pipeline from the start, we can at least start off with a small pipeline and grow it and this document. All contributions to the proposed long term designs are highly welcome.

Production Flows

Games



Film/Episodic



Infrastructure

Each phase represents a different BUGJam project.

Phase 1: Minimal Game Pipeline

- Deploy: We need to be able to deploy the pipeline to each jam participant.
  - Possible solutions

        - Python wheels and upload to the Forgejo package repository, then creating a virtual environment for each of the applications and installing the packages from the Forgejo repository and PyPI.
        - Rez
- Bootstrap: We need to be able to launch the applications in our environment with the proper environment variables, application flags, resources, etc.
    - Possible solutions
        - Shell scripts
        - Launcher application
- Asset Workspace: We need to be able to create a location for an asset to save to.
    - Possible solutions
        - A small utility in Blender to create asset directories based on a given name
- Publish: We need to be able to publish and sync work from production members.
    - Possible solutions
        - Plain Git commands

Phase 2

- Entities tracking: We need to be able to track entities and start to know more about what everything is/what needs to be done.
    - Possible solutions
        - Kitsu
        - Roll our own entities database
- Context management: We should keep track of the current context someone is in when using their DCC so we can easily manage the environment (for example, have the asset library default to show only assets that are part of the shot).
    - Possible solutions
        - Roll our own context manager
- Publish: The publish tool will need to become more production friendly and handle tracking
    - Possible solutions
        - Pyblish
        - Roll our own
- Validations: We should have validations happen at every step of the pipeline.
    - Possible solutions
        - Pyblish
        - Scott's checks framework
        - Roll our own
- Workspace: We should support creating loading scenes via a workspace loader
    - Possible solutions
        - Roll our own

Phase 3

- Metadata: We should start tracking more information about assets, shots, etc. This may include sidecar files for all of the publishes
  - Possible solutions
    - Universal Scene Description
    - Roll our own (sqlite, json, etc)
- Metrics/Error tracking: While we need to make sure we're extremely careful about collecting potentially personal identifying information, being able to track errors and bottlenecks is extremely useful. This would need to be a big discussion with everyone about who gets access to the data (assuming that people are comfortable with the information being stored), and how to do this in a way that'll make people feel comfortable (show them what information would be transmitted, opt into data collection, etc).
- Shot/Map assembler: We should have some automation that could create a new shot/map from scratch in our DCCs using assets that are assigned to the shot.
- Asset library: While Blender does have an asset library, we can add some logic that connects to the asset database, and handle different cases (loading assets to work on or work with).

Revision #1
Created 2025-12-26 18:02:21 PST by propersquid
Updated 2025-12-26 19:40:16 PST by propersquid