****

**APOLLO Art Plan Version 2.0**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Revisions | By Whom | Date |
| 1.0 | Draft Document Created | Sherry Floyd | 08.03.09 |
| 2.0 | Draft Document Updated | Sherry Floyd | 10.20.09 |

Contents

[Contents 3](#_Toc243817848)

[Art Department Goals 4](#_Toc243817849)

[Pre-Production to Production 4](#_Toc243817850)

[Implementation 4](#_Toc243817851)

[Risk Mitigation 5](#_Toc243817852)

[Process: How we do it 5](#_Toc243817853)

[Culture, Growth and Accountability 6](#_Toc243817854)

[Production 6](#_Toc243817855)

[Tools/Tech 11](#_Toc243817856)

[Staffing/Insourcing/Outsourcing 18](#_Toc243817857)

# Art Department Goals

As an art group, our primary focus is to develop and create a Nolan-esque world supporting Game Design and Art Direction goals resulting in a profitable AAA end product.

By fostering a culture wherein *whole team contribution* provides *creative solutions* we will meet project goals, mitigate risks and support transparency; in turn building a successful team and product.

This goal is not met by leads and directors alone but in accountability by all. We are building a process which supports this contribution.

# Pre-Production to Production

As an art team we will always be considering how to best visually reflect and support design and how to best interpret Nolan’s world in ours.

Completion of these features has been identified as core to our success in production.

* Player Movement
* Combat
* Gadgets
* Talent Tree
* Interrogation
* Stealth
* Fear
* City
* Cape
* Vehicles

# Implementation

Timing is everything. Implementation comes in several stages covering the “what and whens”. As we are validating through pre-production we will be adapting weekly via Strike Teams. The strike teams will cover each game feature and focus on review, solutions and possible redirects based on findings.

Project Management:  
It is important for a team in pre-production to remain flexible to the changing needs of the product, tools development and iteration cycles for proof of concepts/systems as well as rolling into the production phase. To meet this need the process must not only be used by the producers but by the whole team, providing valid feedback and organization for all roles. We will use a blend of agile management techniques, strike teams and project/performance metrics reporting.

# Risk Mitigation

Issues and risks will be assessed by producers and directors through art discipline dailies and weekly strike team meetings. These risks will be visibly noted through the risk assessment doc, prioritized and have plans applied to daily work satisfying end requirements and project goals. Details here…

[Risk Assessment PPM3](../Production/Apollo%20PPM3%20Risk%20Assessment.docx)

# Process: How we do it

**Docs used to plan the milestones**  
Owners: Art Development Director/Producers/Leads

* **Apollo Vertical Slice Schedule** - this doc contains the high level feature schedule for vertical slice
* **MS Deliverables Planning** - this doc contains a breakdown of deliverables by feature with the following *Acceptance Criteria*
  + Fidelity definition
  + Teams accountable
  + Sign off (assigned)
  + Requirements for pass/fail
* **Apollo Risk Assessment** - this doc contains all risks associated with production and is maintained and reported weekly

**Management and Tracking**

Owners: Art Development Director/Producers/Leads

* **Dailies and reports**
* **Milestone post mortems**
* **Milestone Redirect/Planning**
* **Product Backlog defined**
* **Jira workflow implementation**
* **Individual Accountability**
* **Staggered Scheduling**

**Dailies**: Daily meeting between the ADD and the art discipline leads to define and resolve blockages and issues as well as work status; report of findings is sent out to team

**Milestone post mortems**: At the end of each milestone a post mortem is held to discuss the following and form action items

* What went right/wrong… what worked… what didn’t?
* What got done and what didn’t?
* Review backlog for implementation of features/reprioritization.
* Resource assessment… anything gained/lost?
* Risk assessment update/recap

**Milestone Redirect and Planning**: This is a full day of planning occurring at the beginning of the milestone to update and deliver the above docs to the team as well as build out tasks. This meeting consists of the following:

* Review/Establish the list of deliverables/goals for the next milestone
* Establish Departments/Strike Teams/Individuals involved in accomplishing those goals
* Establish requirements of the deliverables
* Get together with Leads/Individuals, break down tasks into weekly goals, and enter everything into JIRA for the next milestone

**Product Backlog defined**: This is a recurring assessment and update of the product backlog based on new findings.

**JIRA workflow implementation**: The producers enter tasks for the strike teams and adjust for any discrepancies in time estimates of work; this tool is used for daily and milestone tracking.

**Individual Accountability**: The tasks are derived and agreed to by the assigned; they are merely tracked by the producers.

**Staggered scheduling**: In many cases we have resources which contribute to multiple disciplines; this form of scheduling is necessary to ensure the right resources are available to cover the needs of the project in both pre-production and production.

# Culture, Growth and Accountability

To help aid in the goal of whole team contribution and in support of consistency in art and support of art direction we are:

* Providing expectations, mentoring and training to the Leads and contributors
* Promoting weekly peer review within each discipline
* Bi-weekly team meeting Art and Design presentations for full team understanding of product
* Dailies with Leads
* Redesign of Art Team Confluence page for ease of use and communication

Living docs outlining these efforts can be viewed upon request

# Production

**Leadership:**  
Cory Allemeier, Art Director   
Sherry Floyd, Art Development Director  
Courtney Evans, Technical Art Director

*Apollo Art Team Pipelines* **Apollo Character Team**How a character is created

**Concept** (2-5 MD)(2) Resources: Cory Allemeier, Joon Park

A concept is created (via Cory Allemeier or designated artist)  
This may consist of the following:

* Image reference
* Illustration (Ortho preferably)
* Background info is helpful, i.e. written description, story, design needs, world placement

**Model**   
(3) Resources: Maegan Walling, Matt Radunz, Arnold Ayala (15-20 MD)

**Base mesh:** This may consist of parts borrowed from previously built models, then using Maya/Max the shape and topology of the character is defined. Clean edge loops are ensured in this stage for ZBrush work to begin.

* Review by Cory Allemeier (AD)

**High Poly Mesh:**Defined/Created in ZBrush

* Review by Cory Allemeier (AD)

**Phase 2 (5 MD)**

**Low Poly Mesh:**This is the actual geo represented in game and used by Animation

* Review by Cory Allemeier (AD)

**Phase 3 (5 MD)**

**UV’s and Burns are generated:**These could be partially or fully generated by software yet to be explored. Time for R&D is needed.From this step the following is achieved:

* Normal maps
* Ambient Occlusion

**Phase 4 (5 MD)**

**Texture:**

* Specular maps
* Painting
* Diffuse Maps
* Review by Cory Allemeier (AD)

**Model is reviewed in engine: (Process to be documented and inserted)**

**Shadows are built:** This step can be processed at any time from this point on

Apollo check-in alias needed to report hand off

**Rigging/Weighting**

(1) Resources: Ronald Kury (3-5 MD)

**RIG:**

* Line up bones
* Skin Mesh
* Set up Animation Controllers
* Set up Physics objects
* Set up Sockets
* Create Resource File- model, materials, animations
* Creates Database entry- resource file, textures, AI behavior, animations
* Review by Animation- Josh Paraventi

**Apollo Animation Team**

(4) Resources: Patrick Watje, John Piel, Josh Paraventi, Sean Fitzsimmons (tbd MD)

**Animation:** Import available characters through the Tabbed Panel or Model Import button located on Anim\_Apollo shelf.

* Scale of character is checked
  + Provide feedback to Character setup with problems or suggestions on how to improve functionality.

**Import animations (MoCap/Traditional)**

* Clean animations
* Review in- engine for systems proof of concept
* Assess what’s needed or missing, implement.
* Review by Cory Allemeier (AD)

***Proofing repeats, should be noted in planning along with MoCap time***

***LOD systems doc to be inserted here***

***Cinematic doc to be inserted here***

***Facial animation doc to be inserted here***

***Audio/Animation pipeline doc to be inserted here***

**Apollo World Team**

**Concept (1-2 MD)**Resources: Cory Allemeier, Joon Park, and WA

A concept is created or reference photos are agreed upon (via Cory Allemeier or designated artist)

This may consist of the following:

* Image reference
* Illustration (Ortho preferably)
* Background info is helpful, i.e. written description, story, design needs, world placement

**Model (HIGH POLY MODEL FOR NORMAL MAPPING)**

**High Poly Mesh: (10 MD)**

* Review by Dan Thibadeau (LWA)

**Model (HIGH POLY MODEL FOR ENGINE)**

Resources: Casey Burpee, Anthony Lampa, Andrew Griffin, Brandon Nobbs

**Low Poly Mesh (HIGH LOD): (5 MD)**

This is the actual geo represented in game and used by the LOD system as the HIGH DETAIL version

* Review by Dan Thibadeau (LWA)

**Model (MEDIUM POLY MODEL FOR ENGINE)**

**Low Poly MESH (MEDIUM LOD): (2.5 MD)**

This is the actual geo represented in game and used by the LOD system as the MEDIUM DETAIL version

* Review by Dan Thibadeau (LWA)

**MATERIALS (Ideally HIGH and MEDIUM LODS should use the same material)**

**Texture: (5 MD)**

* Specular maps
* Painting
* Diffuse Maps
* Review by Dan Thibadeau (LWA)

**Setup**: (In engine)

**Create prefab: (2.5 MD)**

* Get assets in engine
* Ensure LODS are functioning properly
* Review by Dan Thibadeau (LWA)

**Model is reviewed in engine: Have content review meeting with Cory Allemeier before final check in**

Apollo check-in alias needed to report hand off

**Apollo FX Team**(2) Resources: Mark Wood, TBD **(0.5 - 20 MD)**

**Types of FX: Small, Medium, Large, Cinematic**Environment  
Weapon related  
Character related  
Vehicle

FX may include any or all of the following:

* Painting FX textures in Photoshop.
* Modeling and Texturing in Maya.
* Searching for reference images and video.
* Working in FXEdit.

**FX Integration (1-6 MD)**The FX team either integrates the finished FX into the game (via the Game Database Editor or adding the FX to Model Keystrings, etc) or passes the finished FX back to the requesting party for them to integrate.

* Review by Cory Allemeier (AD) and requester

# Tools/Tech

This is an outline of all documentation Apollo will produce through the project cycle. Deadlines will be applied. This list can also be viewed through Confluence.

<https://confluence/display/monolith/Content+Creation+Guide+Feedback>

* **Documentation style requirements:**  
  -Must be portable to LithEdit once that framework takes over  
  -Should be holistic  
  -Should be consistent  
  -Should be written from the user's perspective  
  -Should be maintainable and expandable

General format for new articles:

* Table of contents/Sample image
* Guidelines
  + Basics (super basic summary of what the thing is)
  + Depends On/Used By (what other resources use or are used by this thing)
  + Workflow Overview (graphic overview of what files this thing uses for source/intermediate/runtime)
  + Naming, Tagging, and Storing (where to keep this thing, naming conventions, etc.)
  + Sizes (practical size limitations for this thing)
* Source format
  + Ways of creating textures (external programs, methods, techniques, etc.)
  + Other topics relevant to the given thing
* Intermediate format
  + Monolith's editor for that format (textureEdit, modelEdit, etc.)
* Compiling and Rebinding (how to compile this thing, how long should it take, can it rebind automatically)
  + Any special tools that work on this topic (trayClient, etc.)
* Performance, Memory, and Visualization
  + Performance (how to prevent performance problems with this thing)
  + Memory (how to check on this thing's memory usage)
  + Console Commands (how to look at this thing different ways in-game)
* Tips
* Content Creation Library Navbar

**Content Creator Library - Navigation**

**Basics:** Introduction & Concepts - Perforce - Setting Up - Tools Introduction - Runtime Introduction - Creating and running a test level  
**Content Creation Guides:** [*Introduction to Shaders, Materials, and Textures*](https://confluence/display/monolith/Introduction+to+Shaders%2C+Materials%2C+and+Textures) - [*Textures*](https://confluence/display/monolith/Textures) - Texture Animations - Materials - Models - Animations - FX - Prefabs - Worlds  
**Polycount/Memory Quick Reference:** Characters - Vehicles - Prefabs - Buildings - Worlds  
**Tool Quick Reference:** RemoteView - AssetBuilder - ModelEdit - FXEdit - WorldEdit - Perforce  
**Other Topics:** Tagging - Lighting - Fast Volumetrics - Building Tool - Level of Detail - Keyframing - Streaming and Memory - Performance - Dependencies - [*Photography*](https://confluence/display/monolith/Content+Creator+Library+-+Photography)

**Introduction & Concepts**

* General workflow overview (tools, assets, perforce, compilers, runtime)
* Asset creation pipeline overview (source->intermediate->compiler->runtimes)
* Naming conventions, directory structures (just that these exist)
* Tagging and searching (an intro)
* Source assets (overview of different kinds and what creates them)
* Intermediate assets (overview of different kinds, what they're used for, and what tool uses them)
* Runtime assets (how compiled assets go into different trees and onto different machines; how the pc compiled assets are sometimes needed)
* Source control - Perforce (source assets go here, compiled go here, don't check in compiled runtime assets)
* Compiling (locally vs. build machine) (when and how to do it yourself)
* Rebinding (what kind of assets can be rebound, how that works, what works automatically, what needs a trayclient)
* Memory (how we get things in and out of memory, quick overview of how we see how much memory we use)
* Performance (how to see basic things like framerate, basic considerations like draw calls, shader cost, etc.)

**Perforce**

* Setting up Perforce (your client space, client spec, etc.)
* Source assets (what to check in; what to look out for)
* Intermediate assets (what to check in; get everything; when to compile things you get)
* Runtime assets (don't check these in; only get; mask out platforms you don't use)
* Build machine (what it does and when)

**Setting Up**

* Workstation (vista 64; do a get from Perforce)
* Consoles (plug into network, give 360 a host name, throw your ps3 into the path of an oncoming 255 bus)
* 3rd Party Tools (install what you need; version and 64-bit considerations; setting options in external tools)
* Monolith Tools (run installer; should install things you need for; setting remoteView to talk to your console; setup trayclients)
* Game Runtime (how to sync consoles; how to launch the game)

**Tools - Quick Introduction**

* RemoteView
* AssetBuilder & the compilers
* Trayclients
* WorldEdit
* ModelEdit
* FXEdit
* MaterialEdit & TextureEdit
* GDBEdit
* AssetView

**Runtime Quick Introduction**

* Launching from tools
* Launching standalone
* Cheats (in game, vs. in remoteview things)
* Basic console commands (wireframe, drawX, light\_X, light\_addambient, etc.)

**Creating and running a test level**

* How to include your assets

**Content Creation Guides**

**Introduction to Shaders, Materials, and Textures**

* How materials, shaders, and textures work together
* A guide to various shaders, how they look, what textures they need, and what application they're good for
* Solid vs. translucent
* Note on shader creation

**Textures**

* General guidelines - Depends on - Used by
  + Sizes vary by application and platform
  + Understanding a texture's role in materials
* Naming, directory, and tagging standards
* Source format
  + Photoshop DDS plugin and settings
  + Working with mipmaps
* Intermediate format
  + Texture role
  + Compression
  + Mipmapping
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Texture Animations**

* General guidelines
* Naming, directory, and tagging standards
* Intermediate format
  + Working with TextureAnimEdit
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Materials**

* General guidelines
* Naming, directory, and tagging standards
* No source format
* Intermediate format
  + Material flags
  + Alpha channels
  + Solid colors
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Models**

* General guidelines
  + Skeletal vs. rigid models
  + Animation considerations
  + Physics considerations
  + Pieces
  + Bones
  + Sockets
  + LOD
  + Polycounts
* Naming, directory, and tagging standards
* Source format
  + Exporting from maya
* Intermediate format
  + Using ModelEdit
  + Using Havok
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Animations**

* General guidelines
* Naming, directory, and tagging standards
* Using motion capture
* Source format
  + Exporting from maya
* Intermediate format
  + Grouping animations and child models
  + Using modelEdit
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**FX**

* General guidelines
  + FX key type overview
  + Shaders, materials, and their peculiarities for FX
  + Paths for FX to exist in the game
* Naming, directory, and tagging standards
* Intermediate format
  + Using FXEdit
  + Prefab FX
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Prefabs**

* General guidelines
  + LOD considerations
  + Physics considerations
* Naming, directory, and tagging standards
* Source format
  + Using max panel
* Intermediate format
  + Using WorldEdit
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Worlds**

* General guidelines
* Overview of systems in a world
* Naming, directory, and tagging standards
* Intermediate format
  + Using worldEdit
  + Using layers and the node tree
* Source format
  + Using 3dsmax panel where necessary
* World Special Topics
  + LOD
  + Visibility
  + Objects
  + Volumes
  + SpecialFX
  + Scripting
* Compiling & Rebinding
* Runtime performance, memory, and visualization

**Polycount/Memory Quick Reference**

* Characters (Hero, cinematic, ordinary, background)
* Prefabs (Complexity, size, importance)
* Buildings
* World shell

**Specialist Topics**

**Lighting**

* Light Types, uses, requirements
* Baking cubic projectors
* Static lighting - approaches and techniques
* Static Light Image Data
* Fast Volumetric Textures
* Working in realtime
* Dynamic changes and other special lighting tricks
* Performance and diagnosing lighting problems

**Building Tool**

* Basic concepts
* Creating building tiles
* Creating buildings

**Keyframed Rigid Geometry**

* Use cases
* Animating in Max
* Curve editor
* Physics simulation in Max
* Exporting, compiling, and running

**Technical Topics**

**Streaming and Memory**

* Setting up streaming regions
* Using WorldEdit's streaming region builder dialog
* Using exported streaming data
* Finding problem assets

**Performance**

* Introduction to performance
* Diagnosing CPU bottlenecks
* Diagnosing GPU bottlenecks

**Dependencies**

* Using AssetView to examine dependencies
* Scripting AssetView to mine data

**Pre-production Goals for Tech Art can be found here:**  
https://confluence/display/batman/LOD+systems+in+depth

Documents are living and subject to change

**Character Variety tools:** this could provide more in-engine variety using variants and a “mix and match” approach to generic geo. Full details here: <https://confluence/display/batman/Character+Variety+Tools>

**Building Generator :** The building generator is a tool for rapidly creating buildings by taking footprint outlines drawn by an artist or designer and assembling the geometry of that building out of a set of prefabbed tiles. These buildings take advantage of level-of-detail, so that geometric detail and material count drops away as the player gets further away from a given building. Full details here:  
<https://confluence/display/batman/An+Introduction+to+Buildings+in+Apollo>

**Building LOD’s**: Full details on LOD system here: <https://confluence/display/batman/LOD+systems+in+depth>

# Staffing/Insourcing/Outsourcing

* Collect companies to evaluate
* Gather information on each company
  + Securities
  + QA/ Production support
  + Resources, A,B,C team- scalability
  + Best practices for production and communication
  + Strengths and weaknesses in products, e.g. characters vs. props

* Transparent space for info display/sharing
  + Evaluate best solution
    - Review of models/product and feedback
    - Review of point list of above mentioned information
    - How and where info can/should be displayed
    - Who is the audience?
* What should be outsourced?
  + Define what’s appropriate and why
    - Cinematics
    - Audio
    - Characters
    - Props
    - Buildings
    - Levels
    - Animation
    - FX
  + Gather pipelines, tools and process for External handoff
* When should Outsourcing happen?
  + Complete schedule through product of when each contract needs to be initiated
    - How long does a contract take to complete through legal and signing?
    - What are the signing limits? Who are the signoffs per limit?
* Why are we Outsourcing?
  + Resource shortage?
  + Time Shortage?
  + Budget requirements?
* Where does the outsourcing initiate and implement?
  + How long does the work take?
  + To what stage do we expect work to be completed?
  + Who implements “completed” work? Resources available vs. needed
    - Team possible for implementation?
    - Shared resource group for all studios?
    - Each studio responsible for own outsourcing effort?

When should/can an outsourcing plan be rolled out?

Who is building/rolling out the plan?

Who is responsible for its implementation?

How does/will Outsourcing affect the studios/projects/cultures?