# Progressive Render Checkpoint Workflows in Production

Alex Harvill[*] Andrew Kensler[†] David Laur
Pixar Animation Studios

## Abstract

Render checkpoints typically serve only as recovery protection for long running renders. New tool and workflow changes leverage full-frame progressive refinement checkpoints to provide valuable artist feedback and improve productivity in Pixar film production.

## 1 Checkpoints as Artist Feedback

Film production at Pixar relies on an iterative design process across all of our departments. Feedback on the full effect of any given change requires lengthy test renders. Small mistakes are common and often result in discarding many hours or days of rendering. As a result, we have been forced to be conservative with production tests that may affect render times or induce another rendering cycle. Furthermore, long or unpredictable render times can require manual intervention in farm scheduling in order to ensure that particular frames are ready for a review on time.

Our work integrating progressive render checkpoints into our pipeline has neatly solved these real production problems. Rough feedback on the effects of a change are available quickly, making additional creative iterations and experimentation possible and less risky. Mistakes can be spotted early and fixed for resubmission while time is still available, reducing waste and frustration. Useable checkpoints can be taken to reviews at a fixed time of day, while progressive refinement of the same render continues to run.

## 2 Checkpoint Support in RenderMan RIS

Our production renderer, Pixar's RenderMan ("prman") running in RIS mode, can produce final frames by progressively casting sampling rays across the whole frame until it converges on a given level of refinement. Thus the entire frame proceeds from noisy to final quality in a series of passes. This is different than modes that emit final quality tiles or scanlines sequentially without displaying intermediate results, and leaving interesting portions of the image blank, possibly for hours at a time.

The RIS renderer can periodically write these intermediate full frames to disk, at user-defined elapsed time or sample count intervals. The usual purpose of these checkpoint images is to enable *recovery* – if the current render crashes or is killed, a new prman process can pick up again from the last checkpoint. These images are written using a special prman OpenEXR *display driver* that adds metadata and extra channels, using standard EXR format extension tags. This additional data is sufficient for recovery; no auxiliary files are needed. Standard image viewers and other tools will ignore the extra data and see only a regular image file, or the extra data can stripped entirely if the recovery capability is no longer needed.

[*]e-mail:aharvill@pixar.com
[†]e-mail:aek@pixar.com

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
Copyright is held by the owner/author(s).
SIGGRAPH 2015 Talks, August 09 – 13, 2015, Los Angeles, CA.
ACM 978-1-4503-3636-9/15/08.
http://dx.doi.org/10.1145/2775280.2792573

## 3 The Post-Checkpoint Hook

In practice, RIS production renders on the farm run in full-frame progressive mode and are set to write a checkpoint image every 20 minutes. Furthermore, the custom EXR driver also provides one critical additional function. Whenever a checkpoint image is written, the driver also causes an external post-process to be run on the new checkpoint file, while the renderer itself returns to refining the in-memory frame. At Pixar, this post-process does several things. It copies the recoverable checkpoint to a central fileserver where it is safe and can be resumed later if the current farm node goes down. It also records a few basic statistics about the current checkpoint into the image itself and into a production database. It also creates a compressed, playback friendly copy of the image. Artists are then free to use these images for review and sanity checking purposes.

## 4 Changes in Day to Day Behavior

These new refinement, checkpoint, and recover capabilities have led to several important workflow and cultural changes at the studio. Reviews scheduled for 9:00AM are now canceled less often due to lack of finished renders; instead partially refined checkpoints are available and are often suitable for review. The improved feedback loop from these early checkpoint results on long batch renders and the interactive RIS live re-rendering technology both contributed to people being more comfortable with the new renderer during early adoption. Early feedback on overnight renders allow artists to go home confident that there will be useful results in the morning.

## 5 Unexpected Benefits

"Done" is now a fuzzy concept – people can proceed with work while being unaware whether a render has really finished or not. People have an intuitive sense of the how the image is progressing and whether it is "right" or not, even from noisy early iterations. The stored recovery image allows us to tolerate more machine faults on the farm, and we can make use of less robust servers. The recovery capability can also be used to migrate memory-intensive renders to machines with more available resources – simply by killing the process and rescheduling the resumed render elsewhere.

## 6 Challenges

This new workflow presents some challenges as well. Multiple intermediate copies of frames create more I/O on the network and fileservers, the checkpoints are larger than final images due to the additional data, and they may not compress well due to being inherently more noisy. It is not practical to ask prman for full statistics with every checkpoint, and in practice the renderer waits until end of frame to generate some of the summary statistics. Furthermore, it isn't clear how statistics should be combined across restarts, especially when resuming on a different class of machine.

Render farm job scheduling and dependency constraints are also complicated by the presence of useful partial results that themselves need additional processing, such as compositing across rendered sub-elements or creating an animation loop. Experiments with reentrant and checkpoint-aware job structures in our Tractor queueing system have suggested ways that some of these issues may be addressed in the future.