crafty apes

Rez TSC Meeting Presentation

Crafty Apes Timeline

2011: Founded in Los Angeles

2014: Expansion to Atlanta

2016: Expansion to NYC

2020: Expansion to Baton Rouge, Albuquerque, Vancouver (acquisition of CVDVFX) - All right before COVID

2021: Expansion to Montreal

2022: Acquisition of Molecule VFX in NYC

2023: Expansion to London

Crafty Apes pipeline early 2022

- No package management / versioning
- Fully Python 2
- Less than 10 devs and TDs in the pipeline team
- ~ 700 Artists across locations
- Pipeline built on SGTK
- Mostly Windows, with a sprinkle of Linux
- In the process of deciding the future of the Pipeline

Why Rez?

- Open-source
- Industry-tested
- Windows compatibility
- Package management
- Environment management
- Dev workflows (build/install/release)
- In-house solution would have been more resources than we had
- Not many alternatives

Challenges with Rez

Learning curve

- On the dev side:
  - Devs not used to work with packages
  - Need to learn new ways to organize code and think about dependencies
  - Need to get used to building for testing
- On the user side
  - Users used to launch from a GUI (SG Desktop) and barely ever using command line interfaces.
  - Lack of understanding/education about environments

Amount of work needed to switch

- Years of pipeline to organize into packages:
  - De-tangling of dependencies
  - Categorizing code based on responsibilities to decide if it should be part of Package A or Package B
  - Have to define a certain structure to code that didn’t have any
- Hundreds of third party libraries to find and install as packages
  - Lots of third party python packages were used and installed in different ways
  - Had to first identify those as being third-party and finding their source
  - Most were re-installed via rez-pip (although we are not connected to the internet, which required pulling all these from a separate machine, then ingesting them)

Unanswered questions after reading the docs

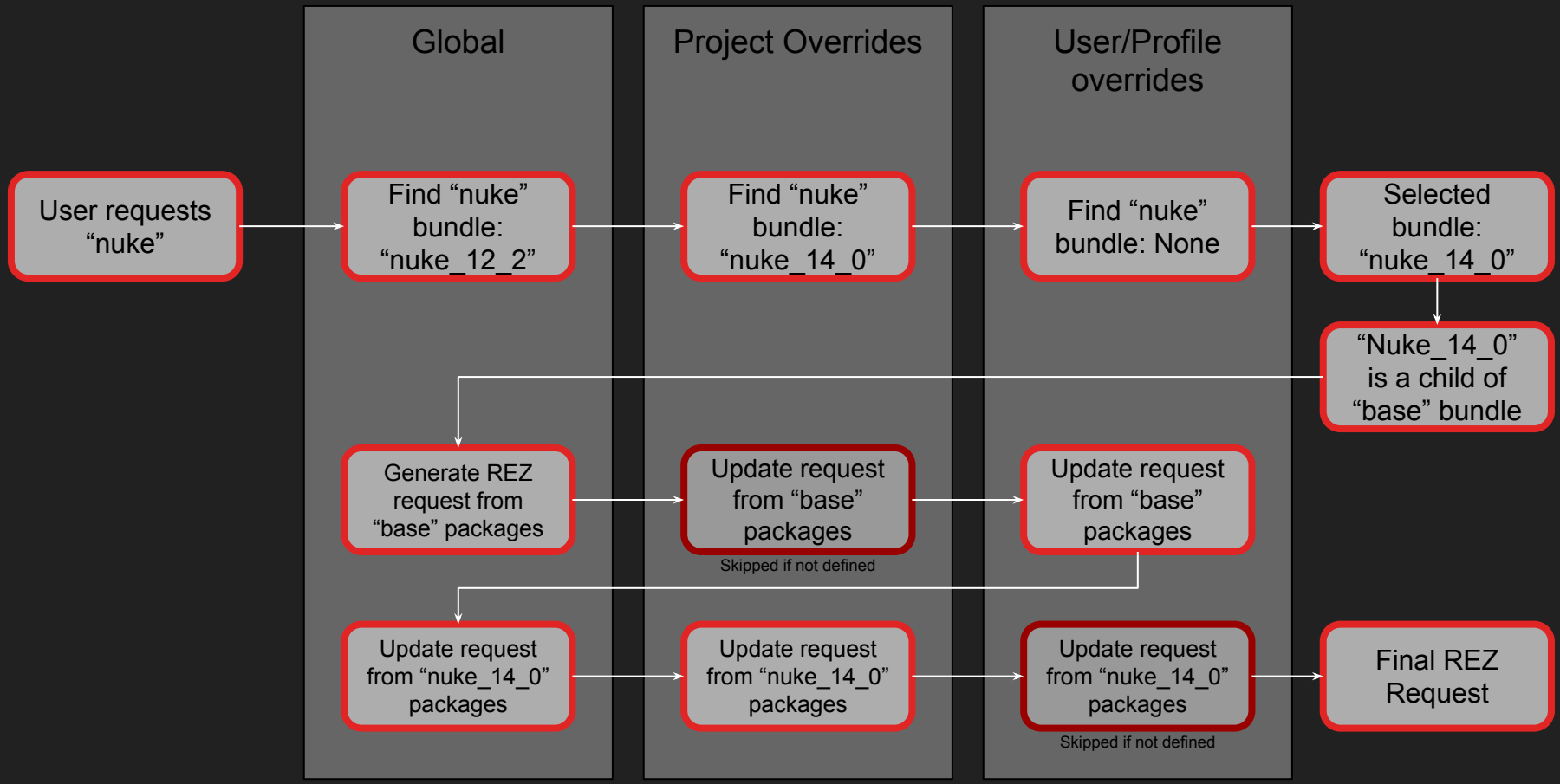
- A few pitfalls to avoid:
  - How to define dependencies without shooting yourself in the foot?
  - How to do “shallow” packages the right way?
  - Weirdness using the CMD shell plugin
- How to manage built packages that should be accessible to multiple users but not load by default?
- How to manage environments per-project, while providing context-aware variables, and loading full suites dynamically?
- How to manage packages that are already vendored within other packages?!
- Some things possibly already built-in that we just didn’t find…

What we did

Dynamic suites

- Multi-layer system used to compose Rez Requests:
  - We create what we unfortunately named “Bundles” (not the same as Rez bundles)
  - Bundles can be composed of other bundles (Base Bundle in Nuke bundle)
  - Bundles can be overwritten / modified at multiple levels: Global/Project/Sequence/…/User
  - The bundle is then “flattened” to a single list of requests to be fed to Rez.
- Only critical packages get a version specified, while the rest is left to resolve live.
- Packages get added to Bundles using one of 3 modes: set/remove/constrain
  - Set: If a package request for this package already in the request, replace with this
  - Remove: If package already in the request: remove
  - Constrain: If a package request for this package is already in the list, set the version range to the intersection of both ranges
- Required to have some code before rez-env to generate request

Bundle resolution



Context-aware env variables

Following years of SGTK and its Tk-Context, we needed an alternative

- Add environment variables to keep track of project/shot/etc..
- Use that same context to define which Bundle to resolve

Was implemented as a ResolvedContext.execute_shell() post_action_callback:

- Re-applies the variables in the correct order onto the rex executor

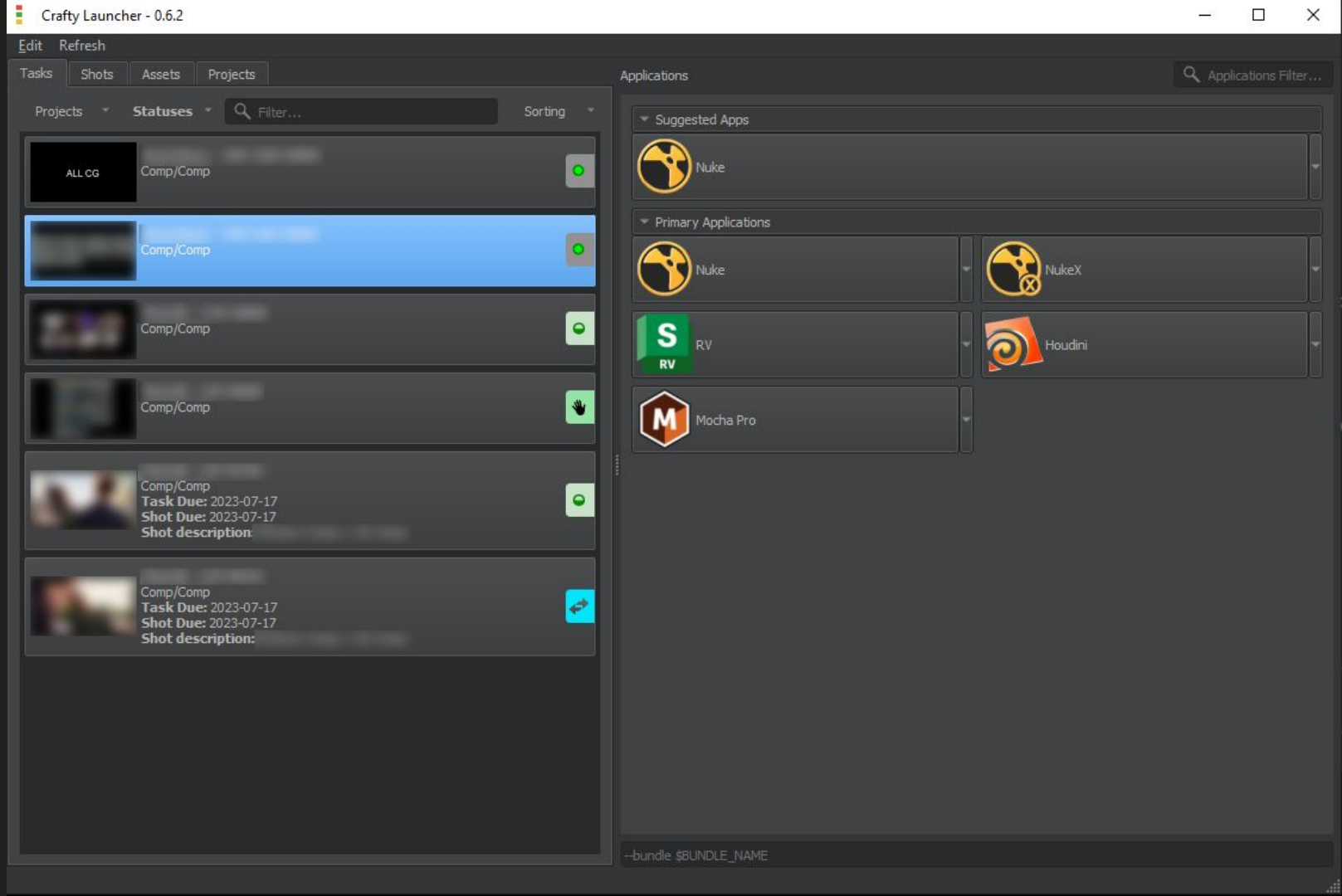
Had to also be implemented for ResolvedContext.get_environ() so we can obtain the environment as a dict with our custom env variables.

Shared non-released packages

We needed a quick way to build a package and let multiple users test it, without doing a full release which would cause the package to get automatically picked up by our dynamic Resolves.

- Set a global pre-process function to insert a token into version number that are built for sharing
- Set a filter to ignore these packages
- Add logic to our ResolveContext to ignore that filter for any package explicitly requested (==)

Making a GUI


Crafty Launcher - 0.6.2
Edit
Refresh
Tasks
Shots
Assets
Projects
Projects
Statuses
Filter...
Sorting
ALL CG
Comp/Comp
Comp/Comp
Comp/Comp
Comp/Comp
Comp/Comp
Task Due: 2023-07-17
Shot Due: 2023-07-17
Shot description:
Comp/Comp
Task Due: 2023-07-17
Shot Due: 2023-07-17
Shot description:
Applications
Applications Filter...
Suggested Apps
Nuke
Primary Applications
Nuke
NukeX
RV
Houdini
Mocha Pro
--bundle $BUNDLE_NAME