Wayside.at: Case Study

March – July 2025

Capital Fringe

Kenilworth Aquatic Gardens - Down to Earth

HyperAllergic Article

1. UX
2. 3D + Animation

1. UX Situation

Artist and stroke survivor Andrew Kastner is developing an immersive augmented reality (AR) app during his residency at Down to Earth. This app transforms park information—from existing waysides and outside research—into post-linguistic interactive AR experiences.

The goal is straightforward: to connect visitors with the park's rich history, current challenges, and future through more sensory and open-ended interactions.

1. UX Situation

Rather than designing for focused, screen-based attention, the experience needed to feel ambient and integrated with the natural environment. Visitors shouldn't feel like they're "using an app" but rather discovering hidden layers of the landscape itself. AR elements should feel native to the space rather than imposed upon it.

2. UX Task

MY ROLE

My role was divided into two foci. I designed the app's experiential framework while simultaneously creating 20+ detailed 3D models that would become the visual heart of the piece.

As we explored how to translate the artist's vision into interactive digital form, I found myself diving deeper into the technical pipeline—collaborating on point cloud conversion and ultimately contributing to the Three.js development to solve the complex challenge of seamless morphing between geometries.

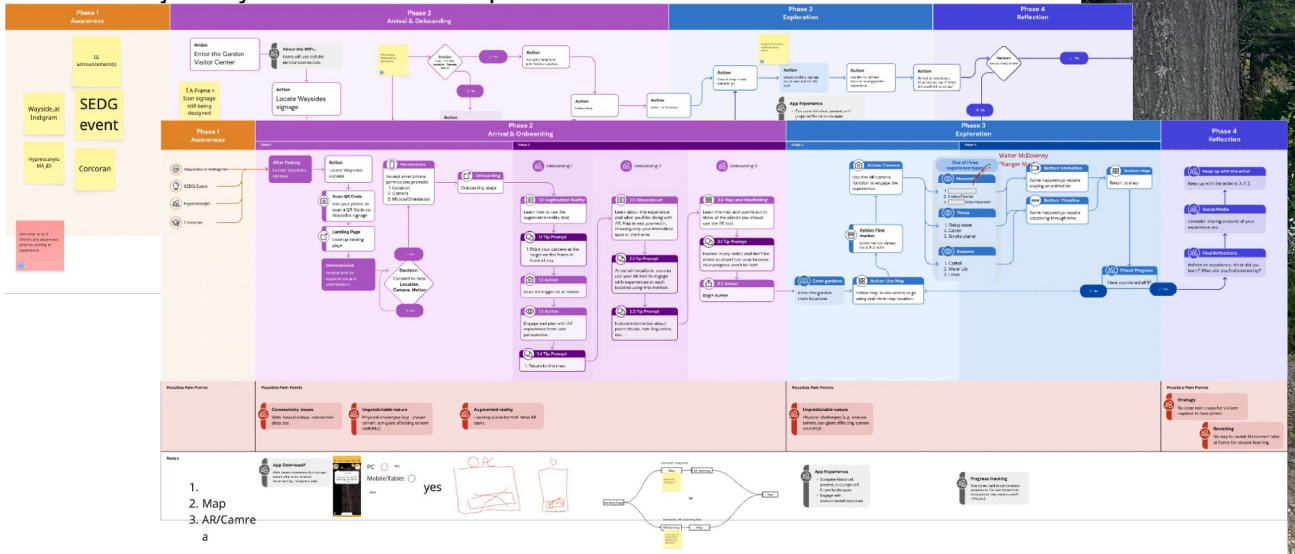
Goals

1. Minimal User Interface
2. Practicing Post-Linguistic
3. Ease of Use for Augmented Reality

3. UX Action

How do you communicate rich, complex information about a place when traditional language-based interpretation fails both the communicator and many potential audiences?

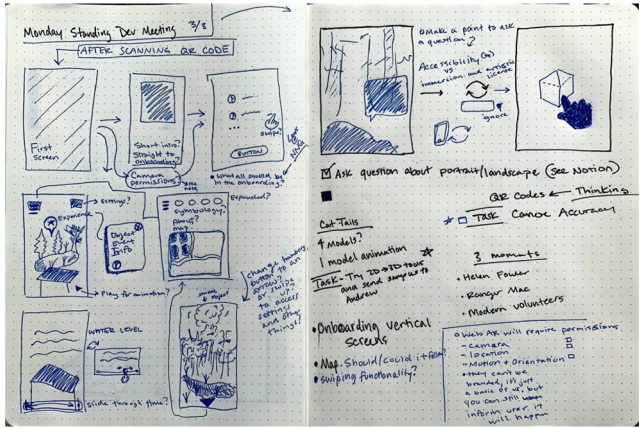
We began by observing park behaviors, and ideating how one might engage with a digital wayside. Multiple collaborative sessions in Miro including outside validation refined the journey into a confident map.

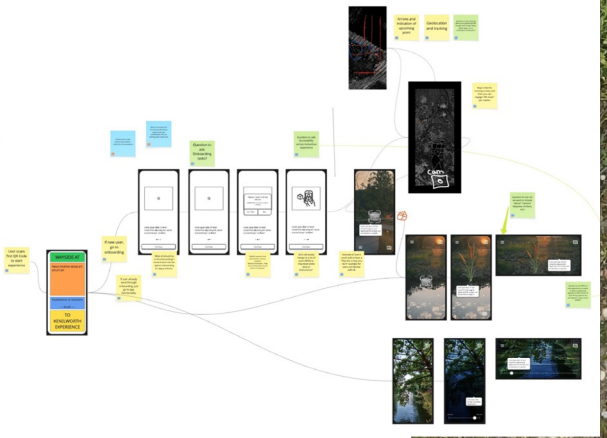


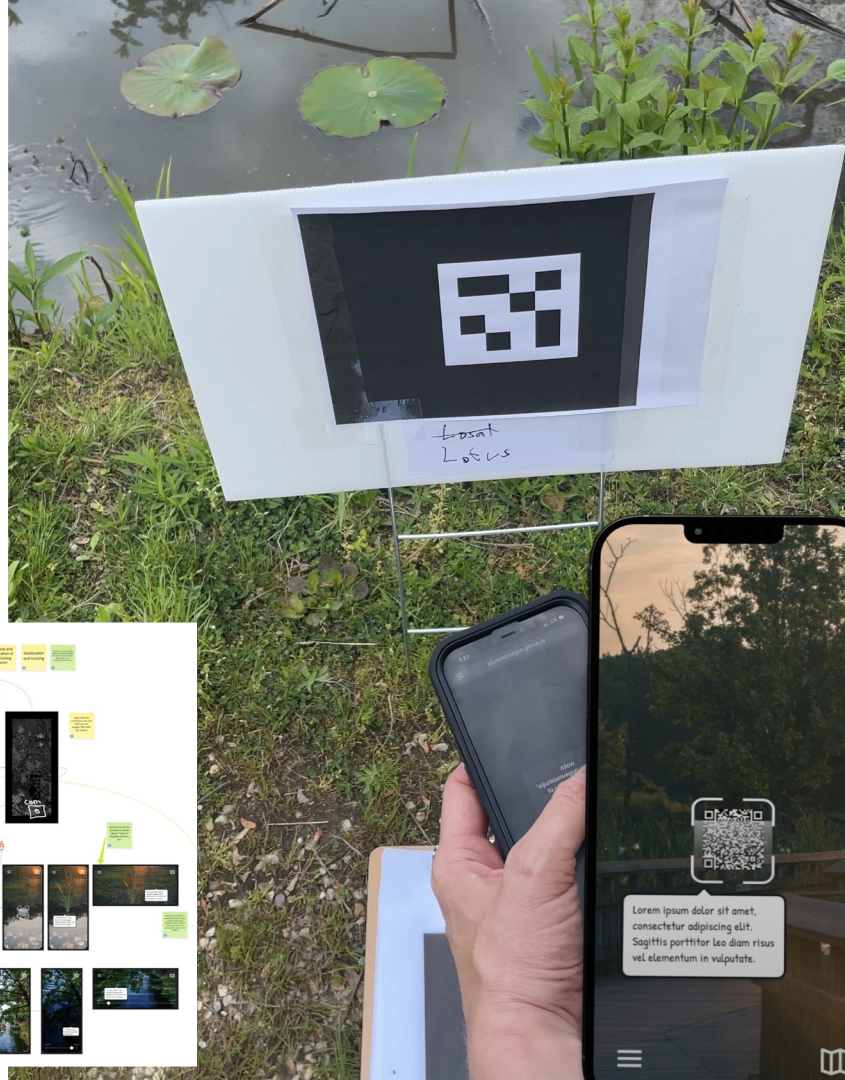
Target-based activation

Rather than presenting static information, I designed a system where visitors could trigger “scenes” by stylized QR code markers. Viable at the time but will present a challenge later.

I began doodling on paper to ideate wireframes before creating the first low fidelity frames in Figma. The interface design prioritized invisibility—minimal UI elements that appeared only when needed, allowing the AR content to feel integrated with the landscape.

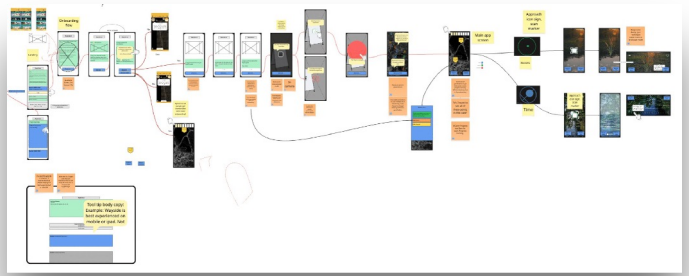


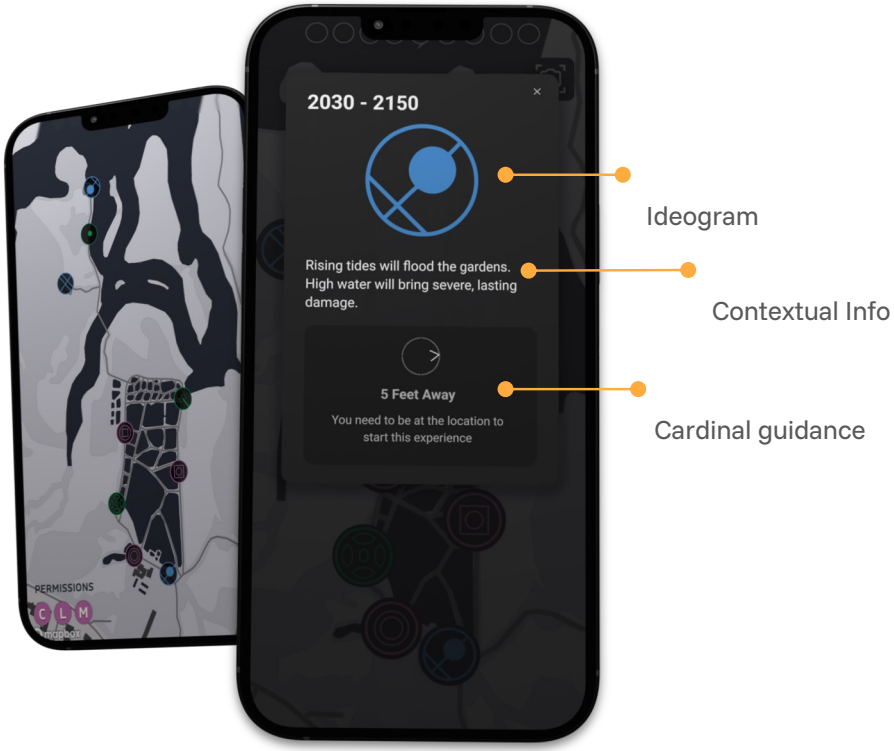




Our first constraint was more abstract: National Park policy updates and park management forbid even the most unobtrusive QR signage to be installed.

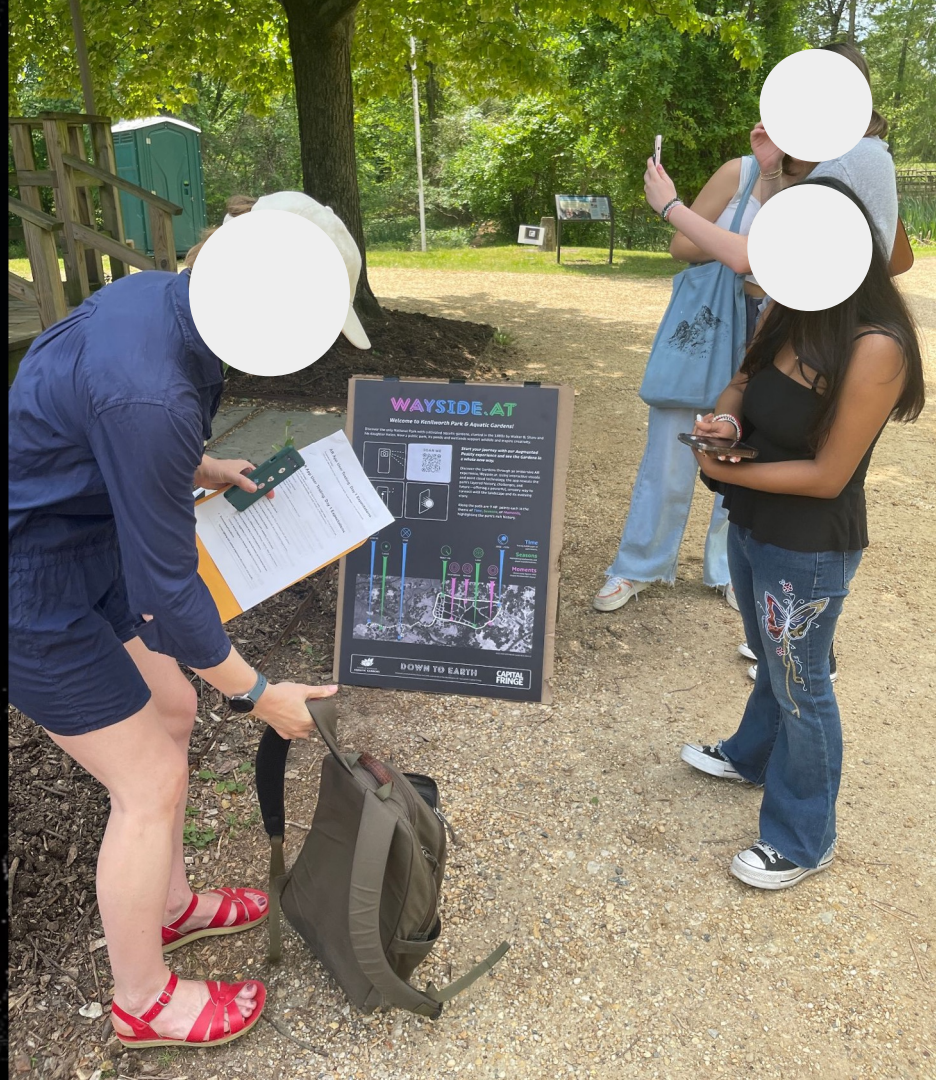
So we dropped the target based model for proximity – a small popup modal when you get close to an experience. The feature is also available manually if you tap any of the map icons.





Testing with Real Visitors

To validate our prototype and test any new ideas (such as the proximity based model we landed on), we conducted informal testing with actual park visitors, observing how people naturally discovered and interacted with the AR elements. Opportunities presented themself over several sessions.

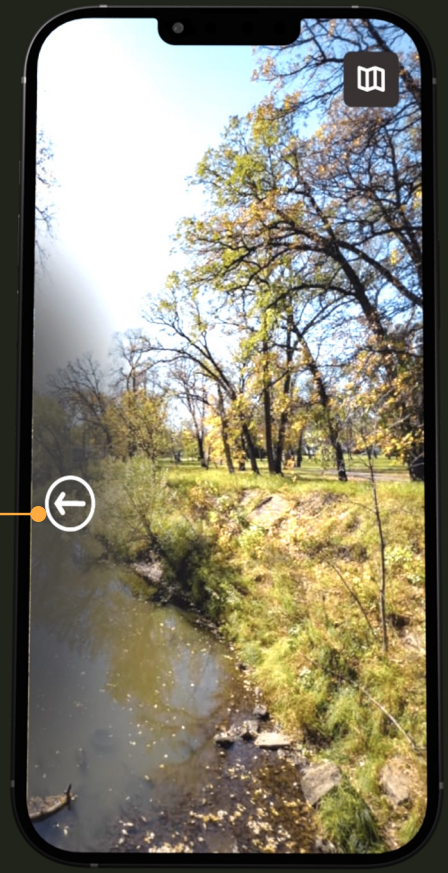


Opportunities from User Testing

Orientation

There are endless angles visitors can view experiences from. Unless they had prior experience with augmented reality, not immediately seeing the model with a nudge to move the phone to discover, visitor retention would drop – believing something was wrong and therefore giving up.

An iconographic pointer would point in the cardinal direction to the point cloud experience. Including a small gradient for visibility, the final version would adhere to style guidelines.



Usability in Natural Environments

Outdoor AR presented their own constraints: tree canopies interfering with tracking, variable lighting affecting interface accessibility, and the need for experiences to work without reliable internet connectivity.
Original dark map; great artistically, not visible in sun.
Simplified light map easier to view in the sunshine

1.15:1 COLOR CONTRAST

7.45:1 COLOR CONTRAST

4. UX Result

In the latest user testing, visitors were able to successfully follow the ideogram-based map and engage with each experience, finding the point cloud objects when they were in their starting viewpoint.

The project not only produced an accessible experience into post-linguistic art, but gave me insights to keep in mind for future design decisions, such as accounting for natural environmental effects like the sun, or

Thanks for reading my design case study for Wayside.at. Keep reading if you're interested in 3D modeling, point clouds, and animation!

2. 3D Task

MY ROLE

My role was divided into two foci. I designed the app's experiential framework while simultaneously creating 20+ detailed 3D models that would become the visual heart of the piece.

As we explored how to translate the artist's vision into interactive digital form, I found myself diving deeper into the technical pipeline—collaborating on point cloud conversion and ultimately contributing to the Three.js development to solve the complex challenge of seamless morphing between geometries.

Goals

1. Optimized models of historical contexts of Kenilworth
2. Renderable point clouds
3. Painless handoff to developer

2. 3D Action

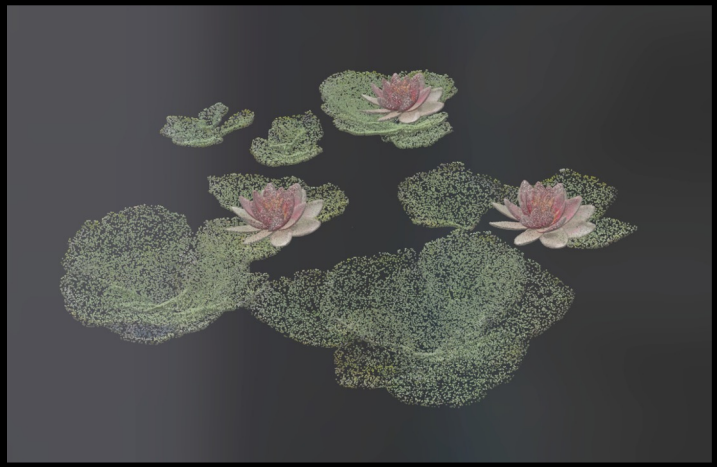
Each of my 20+ 3D models represented different aspects of the park's layered narrative—historical moments, ecological processes, future possibilities.

I learned/used some new techniques in Blender to texture my models with photos – especially paying off when using historical photos.

2. 3D Task

Working with Andrew, we converted these into point clouds that could feel both abstract and familiar, allowing for personal interpretation while maintaining connection to place. The **biggest challenge** was getting to the point cloud milestone in a way that was usable by the developer.

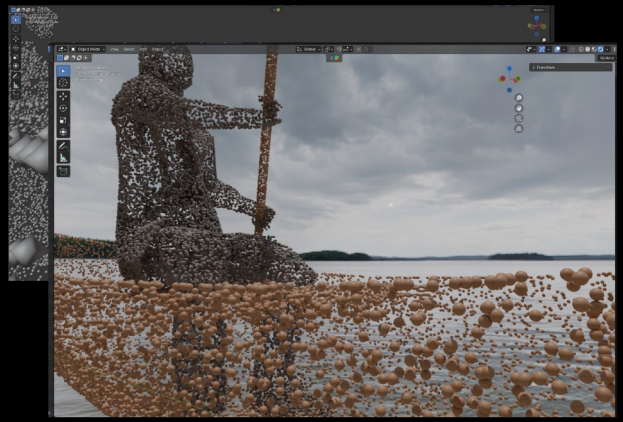
Textured 3D Model

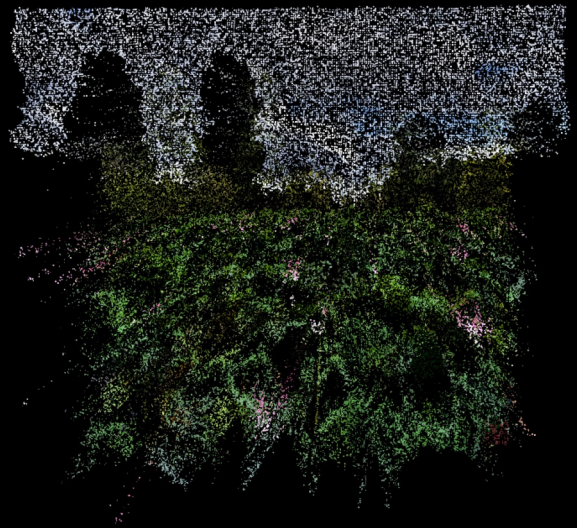


Point Cloud PLY File with embedded texture color information

3D Dead Ends

My first idea was to use Geo-nodes in Blender. It produced an interesting result, like something from Magic School Bus, but it lost texture colors, and would crash when I tried to animate.





Then I tried TouchDesigner, as I was learning it at the time. Again, cool result, but not something readily useable by developers.

I even tried a command line interface tool developed by EA called mesh2splat but ran into technical issues I couldn't dwell on due to time constraints.

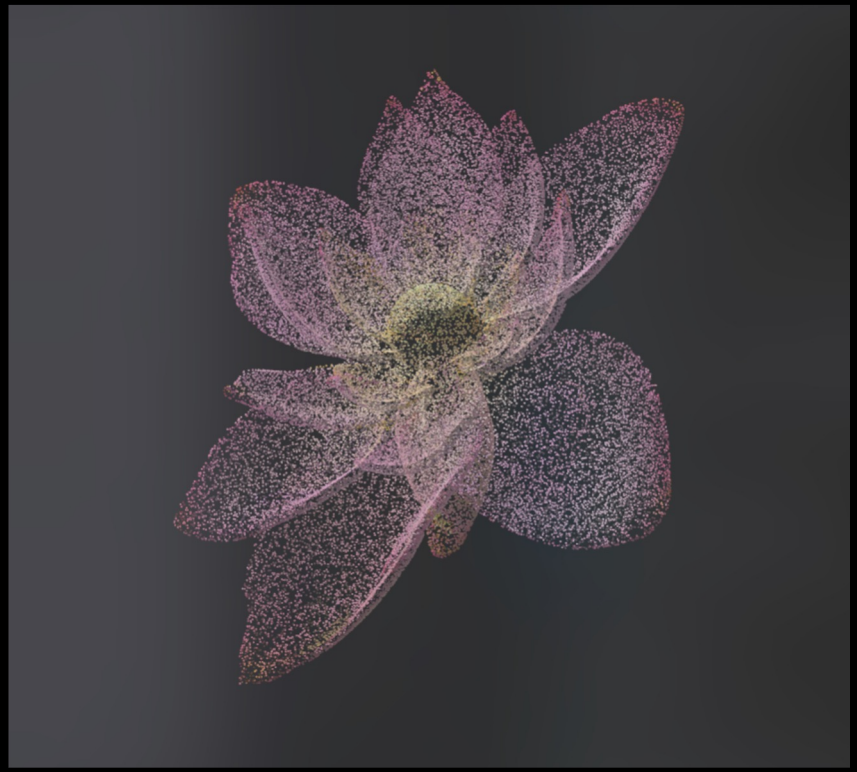
```
Successfully installed pip-25.1.1
➜  ~ gsconverter -h
zsh: command not found: gsconverter
➜  ~ 3dgsconverter -h
3D Gaussian Splatting Converter: 0.2
usage: 3dgsconverter [-h] --input INPUT --output OUTPUT --target_format {3dgs,cc} [--debug] [--about] [--rgb]
                     [--bbox minX minY minZ maxX maxY maxZ] [--density_filter [DENSITY_FILTER ...]]
                     [--remove_flyers [REMOVE_FLYERS ...]]

Convert between standard 3D Gaussian Splat and Cloud Compare formats.

options:
  -h, --help            show this help message and exit
  --input INPUT, -i INPUT
                        Path to the source point cloud file.
  --output OUTPUT, -o OUTPUT
                        Path to save the converted point cloud file.
  --target_format {3dgs,cc}, -f {3dgs,cc}
                        Target point cloud format.
  --debug, -d           Enable debug prints.
  --about               Show copyright and license info
  --rgb                 Add RGB values to the output file based on f_dc values (only applicable when converting to Cloud
                        Compare format).
  --bbox minX minY minZ maxX maxY maxZ
                        Specify the 3D bounding box to crop the point cloud.
  --density_filter [DENSITY_FILTER ...]
                        Filter the points to keep only regions with higher point density. Optionally provide 'voxel_size'
                        and 'threshold_percentage' as two numbers (e.g., --density_filter 0.5 0.25). If no numbers are
                        provided, defaults of 1.0 and 0.32 are used.
  --remove_flyers [REMOVE_FLYERS ...]
                        Remove flyers based on k-nearest neighbors. Requires two numbers: 'k' (number of neighbors) and
                        'threshold_factor'.
➜  ~ 3dgsconverter -i /Users/coreyhowell/Desktop/testtest.ply -o output_test.ply -f 3dgs
3D Gaussian Splatting Converter: 0.2
The provided file is not a recognized 3D Gaussian Splat point cloud format.
➜  ~
```

2. 3D Result

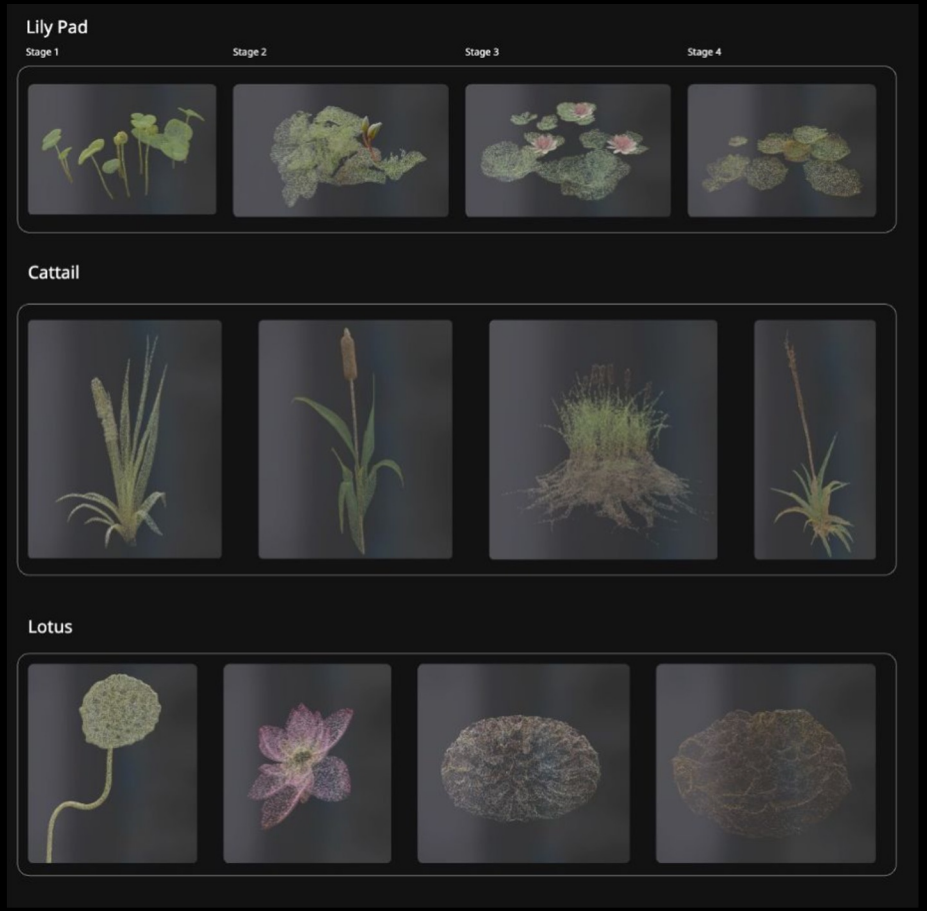
I found success using CloudCompare. Through some back and forth testing with the developer, we built a reasonable production pipeline, creatively approved by Andrew.

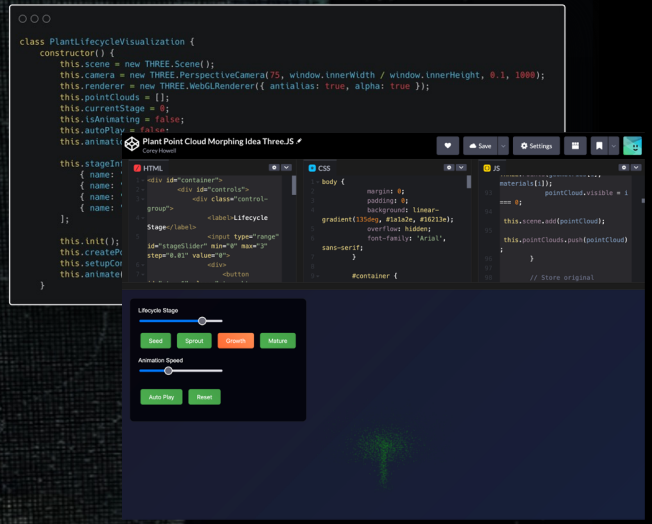


2. 3D Result

One final challenge remained.

How to morph between four seasons' models in-app?





I used Three.JS to prototype a quick function that morphed between four PLY files with smooth bezier interpolation (very quick summary).

Andrew approved the art aspect and Kevin our developer was able to take my code snippet and implement it into the app some additional features. It was the first usable Three.JS i’ve contributed to!


Thank you!

Hey, Corey here.

This case study shows my design process and creative work, which is all my intellectual property. It's here for you to check out my skills, not to copy or share around. The client details and design solutions are confidential, so please respect that and don't use any of this stuff without asking me first. Thanks!

→ coreyhhowell.com