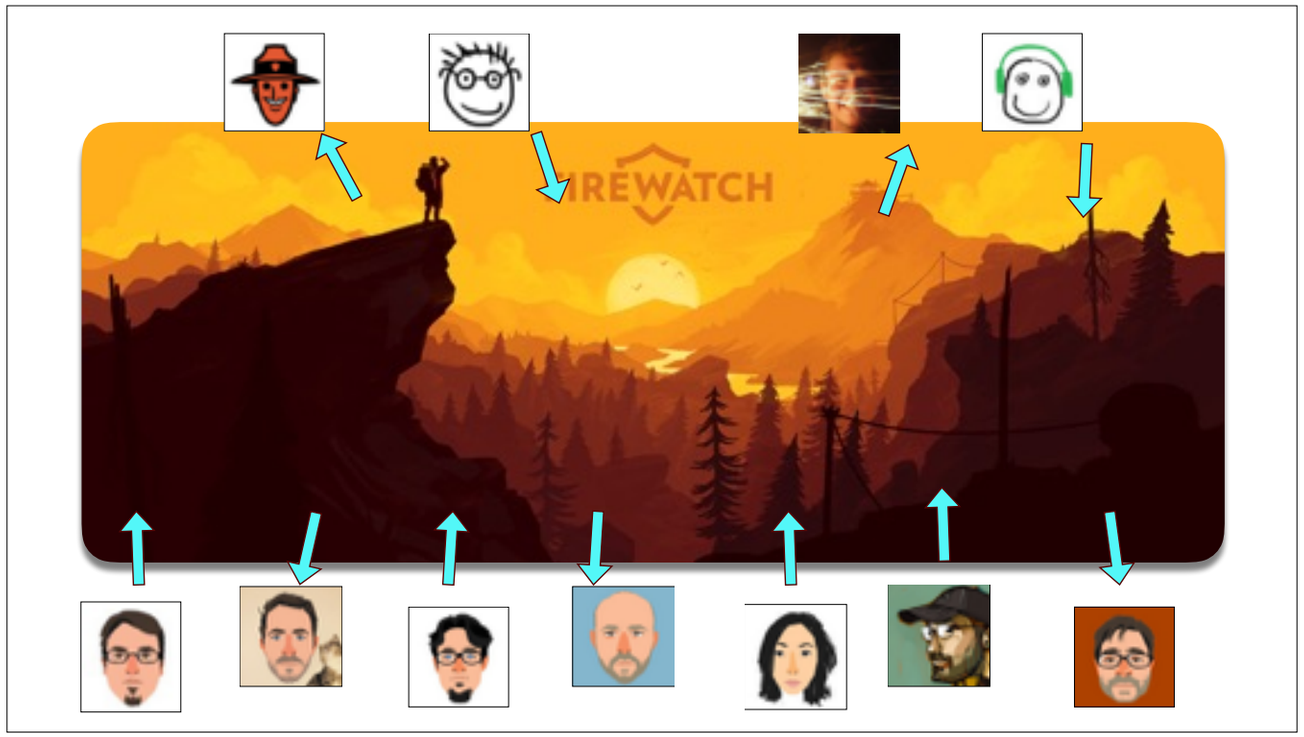
Hello everyone, thank you for coming to this session!

My name is Jane Ng and I’m the lead artist at Campo Santo.

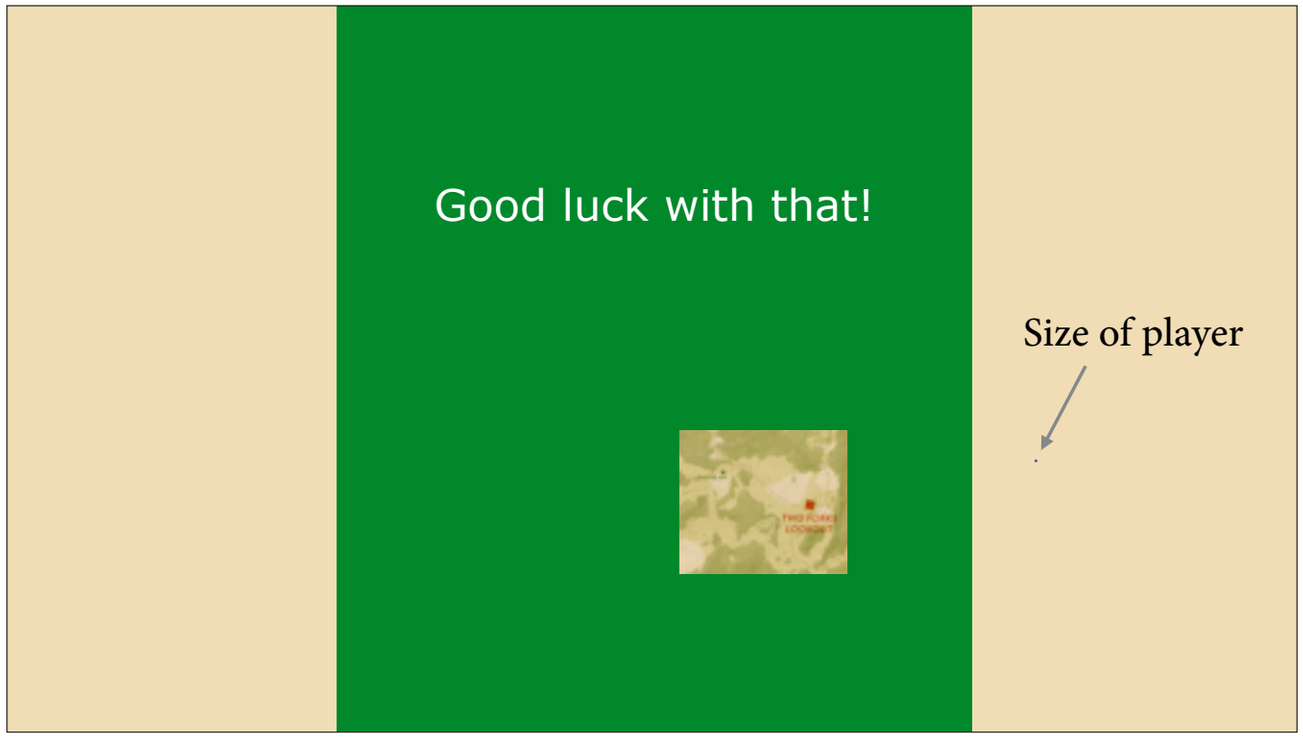
Today we will be talking about Making the World of Firewatch!

We at Campo Santo spent the last 2 years making this game, and we knew from the beginning that it was kind of a crazy idea.



We knew that it was going to be tough, workflow wise, for a team of over 10 people to make any open-world-ish game in Unity.

Let alone one that features a very stylized forest, with tons of trees.



That is supposed to be THIS big when we can only realistically probably only draw this much of the game at full res any given time.

Now I know all this is crazy ambitious because I have been doing environment art professionally for over 13 years. I worked on Godfather, which was EA's first open world game. That team had 200+ ppl and a custom engine.

When I joined Double Fine Productions, I also worked on yet another open world title, Brutal Legend, this time with only about 60 people but also a custom engine.

So I knew when I joined Campo Santo in the fall of 2013 to make Firewatch. I was signing up for a serious challenge.

In game screenshots taken by players.

As I said we had about 11-12 people, 8 of us are in San Francisco, 3 are overseas.

In game screenshots taken by players.

We all wear many hats, but about 3 of us were full time directly involved with creating the world art.

Olly Moss
Concept / 2D / Lighting

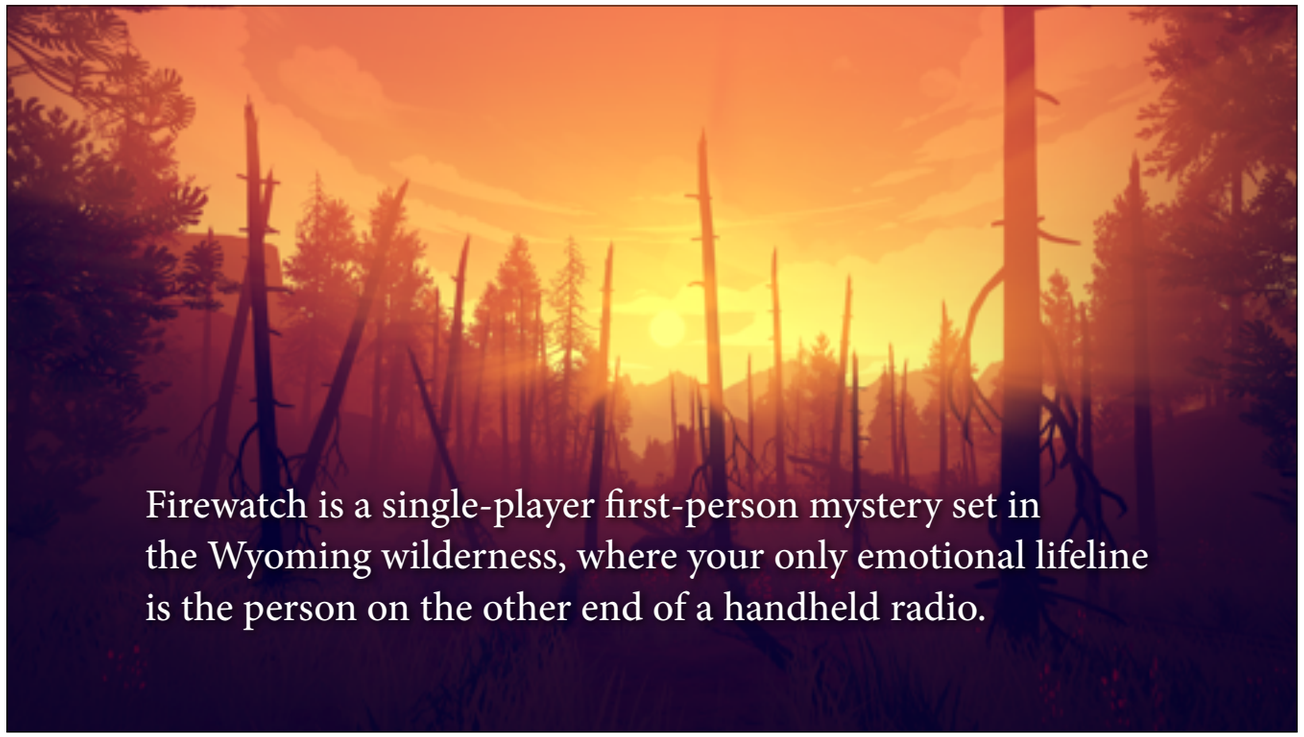
Jake Rodkin
Greybox / UI

Jane Ng (me!)
3D art / World Design

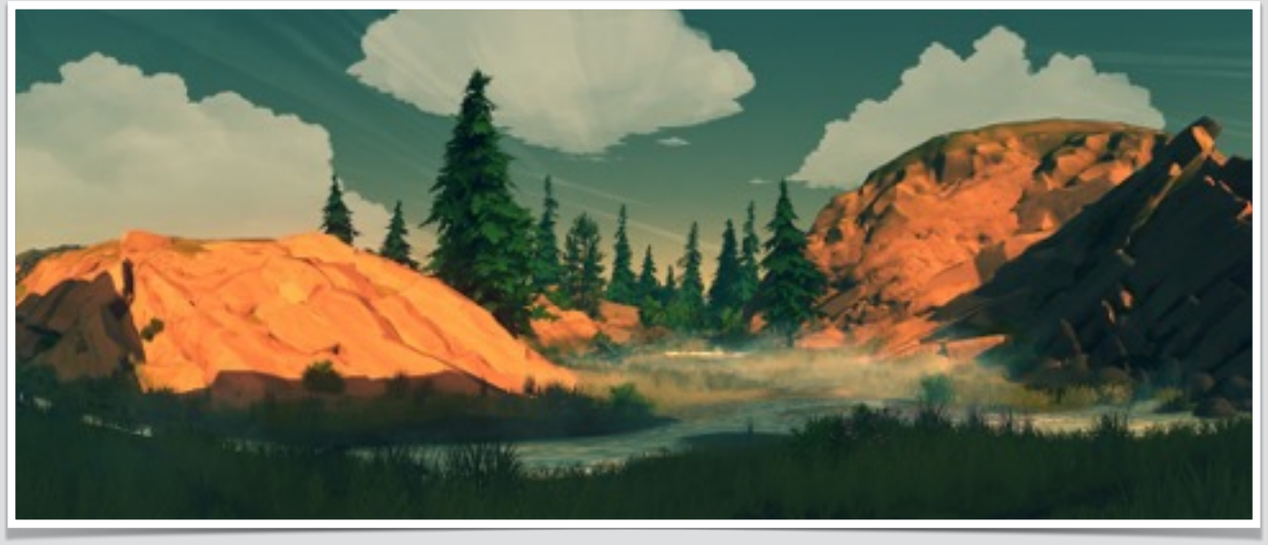
Will Nunes
contract prop artist

Olly Moss did production design, the concepts, a lot of 2D work, and also lighting. Jake Rodkin was in charge of greyboxing and level design. Though he was also 100% in charge of UI. We hired a contract prop artist for 6 months. But I'm the one full-time 3D artist on the team.

Not only would I be responsible for the 3D environment art in the game, I would have to make sure our art production pipeline is as efficient as possible, and that all the world art ties into level design that helps tell a story.



Firewatch is a single-player first-person mystery set in the Wyoming wilderness in 1989, where your only emotional lifeline is the person on the other end of a handheld radio. It is a character driven adventure game with lots of exploration and interactive dialogue.



In game screenshots taken by players.

So I'd say overall we are very happy with how the visuals turned out in the game. These shots I'm showing you are not marketing shots, they were just taken by random players.

In game screenshots taken by players.

The game ended up being aesthetically very stylized in the way we wanted, but it also felt very real in the way we wanted.

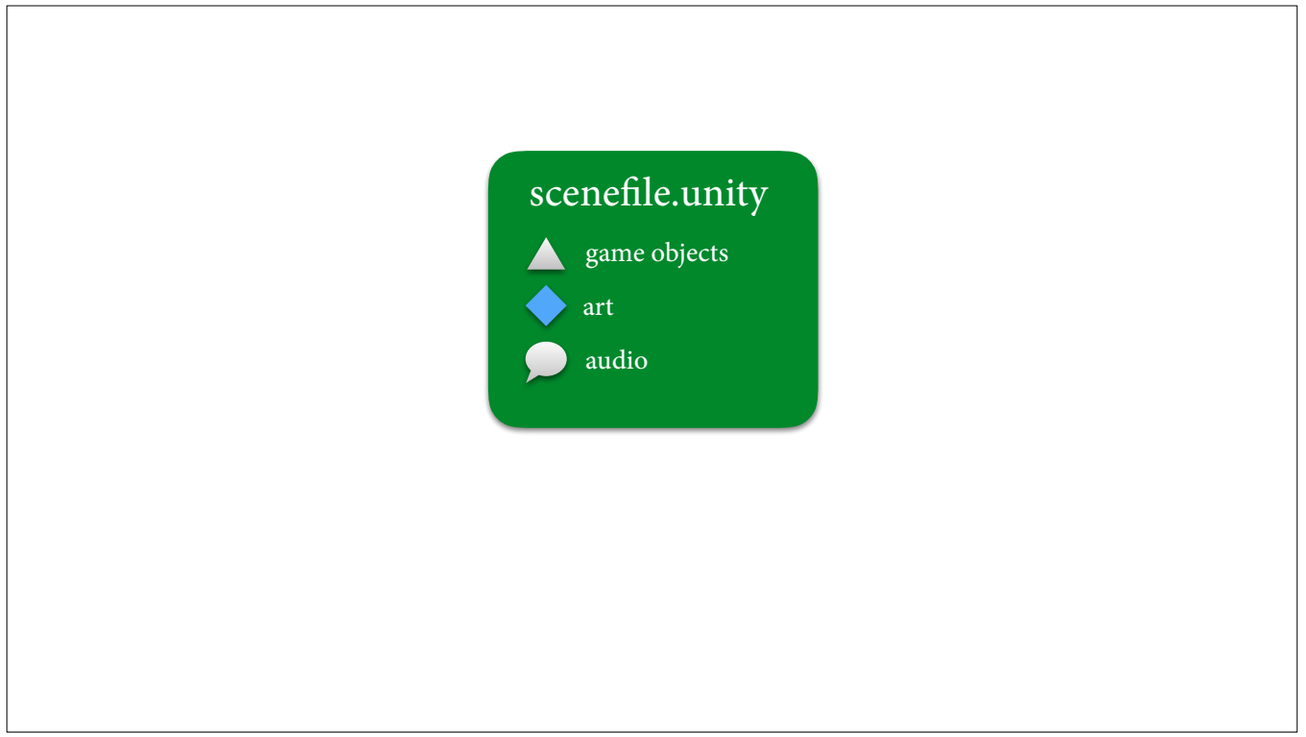
Not an open world but . . .

. . . Lots of open world-ish issues

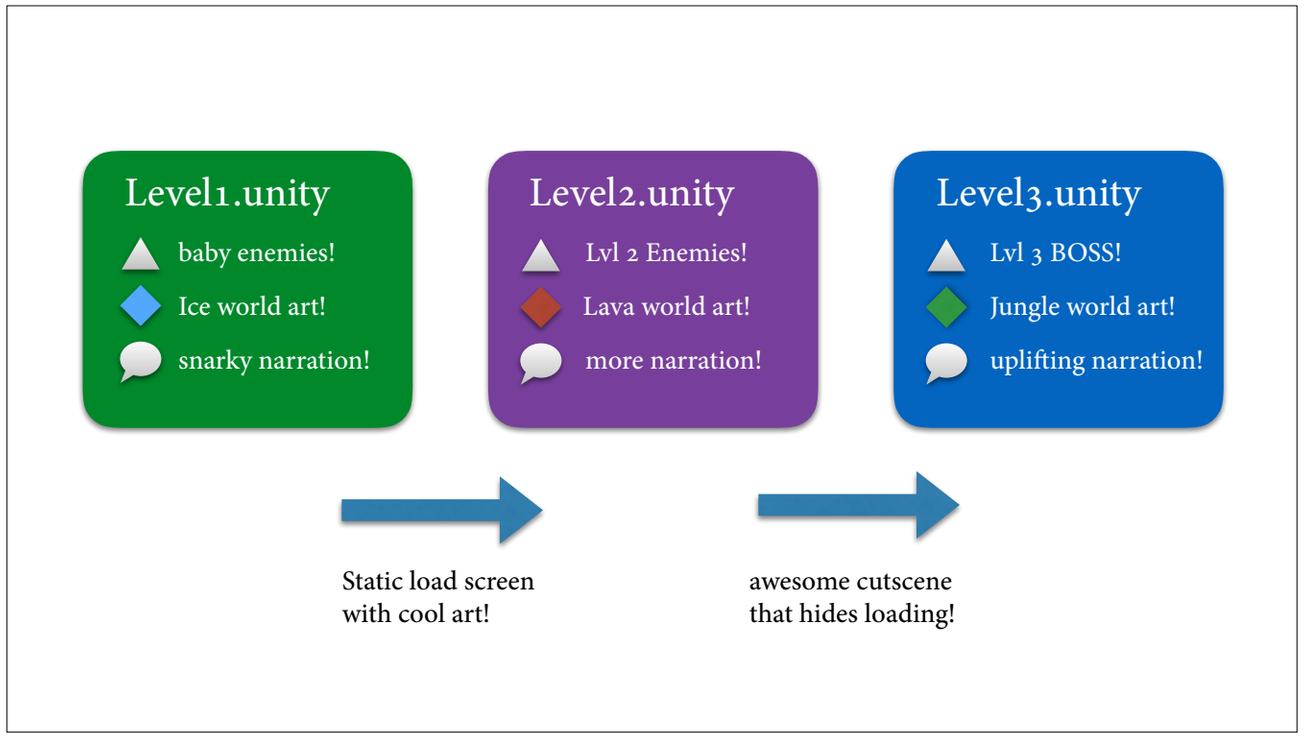
- Concurrent work with 10+ dev
- Big contiguous world
- Lots of foliage and terrain
- World design has to feel natural yet give a sense of narrative progression

In this upcoming hour I will talk in detail about how we dealt with the following challenges.

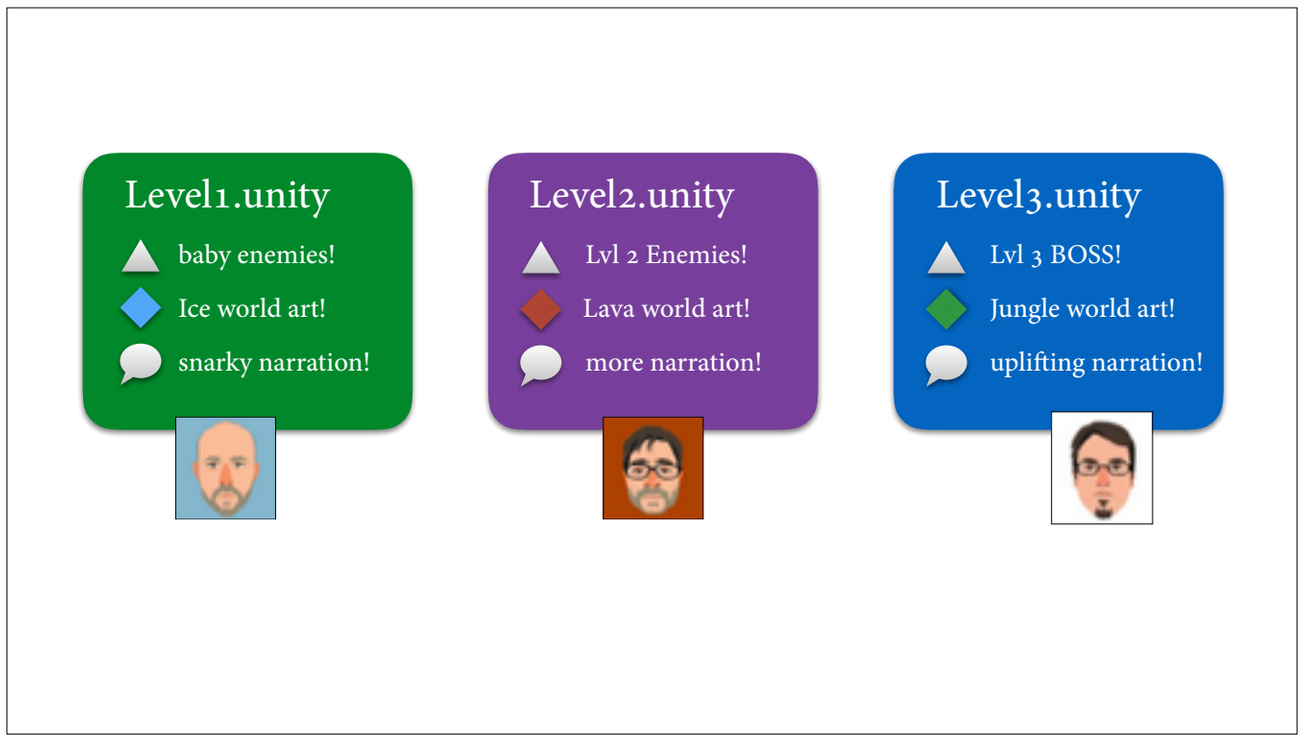
How to do concurrent work effectively in Unity is actually a very common workflow challenge.



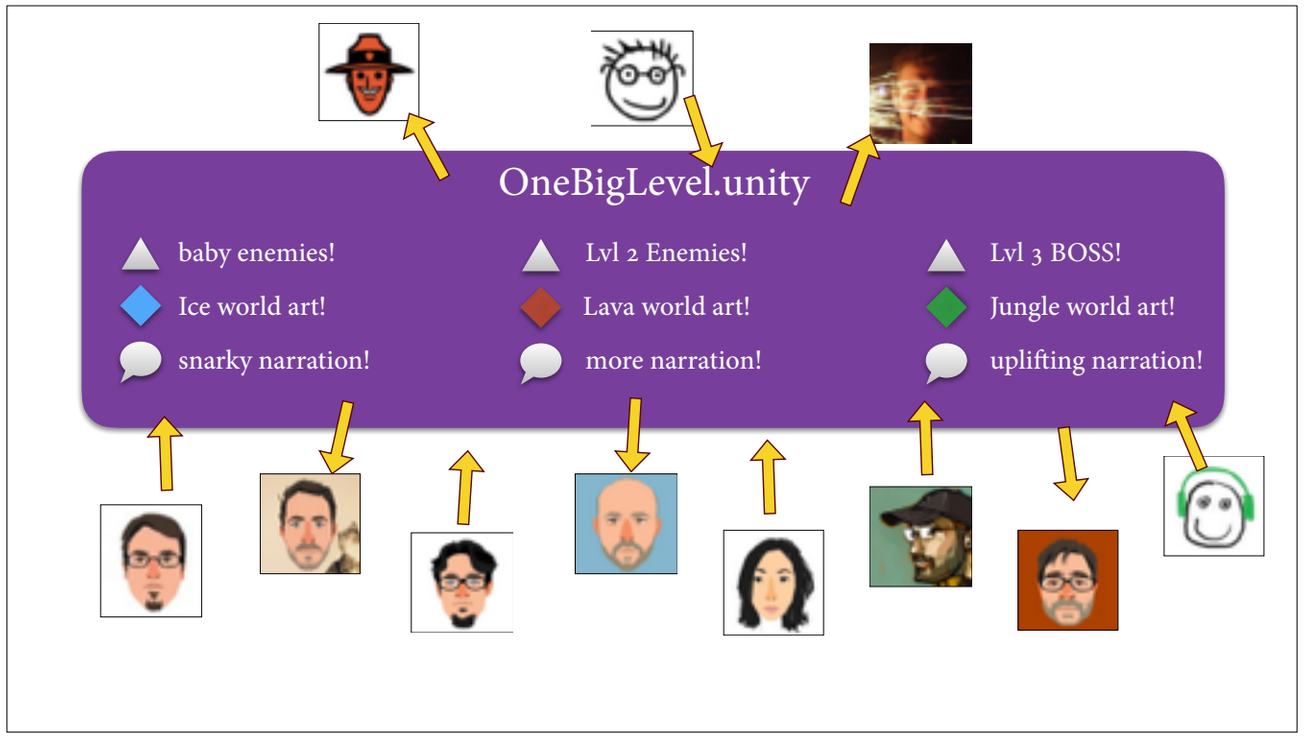
A scene file contains all your game objects, game art, game logic, lights, etc. Think of each unique Scene file as a unique level.



In a very simple game that has let's say 3 levels, you would likely have 3 different scene files. Your entire level1 would be one scene, and a loading screen would hide the game loading in the level 2 scene, and so on.

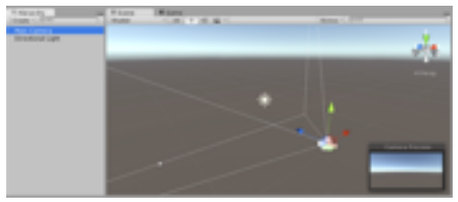


So if you have let's say 3 people making this game, you can easily avoid stepping on each other's work by having diff people work on diff levels at any time.



But what if you game only has one level? And for Firewatch there are 11 people and one level. More than one person can technically work on the same scene file at the same time and you can “merge” your changes together afterwards, but in reality that is a very very messy process.

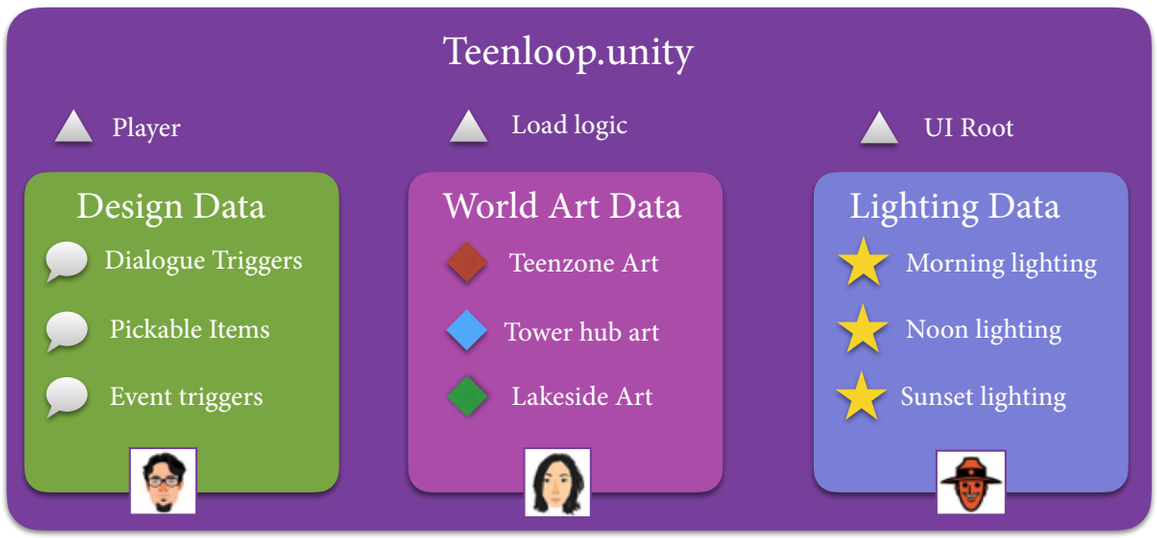
“We just don’t used scene files, we just use prefabs and the game assembles the scene when you run the game.” – my friend Matt



“We had a rubber duck. If you wanted to modify the main scene you had to obtain the rubber duck and put it on top of your screen.” – @DarioSeyb

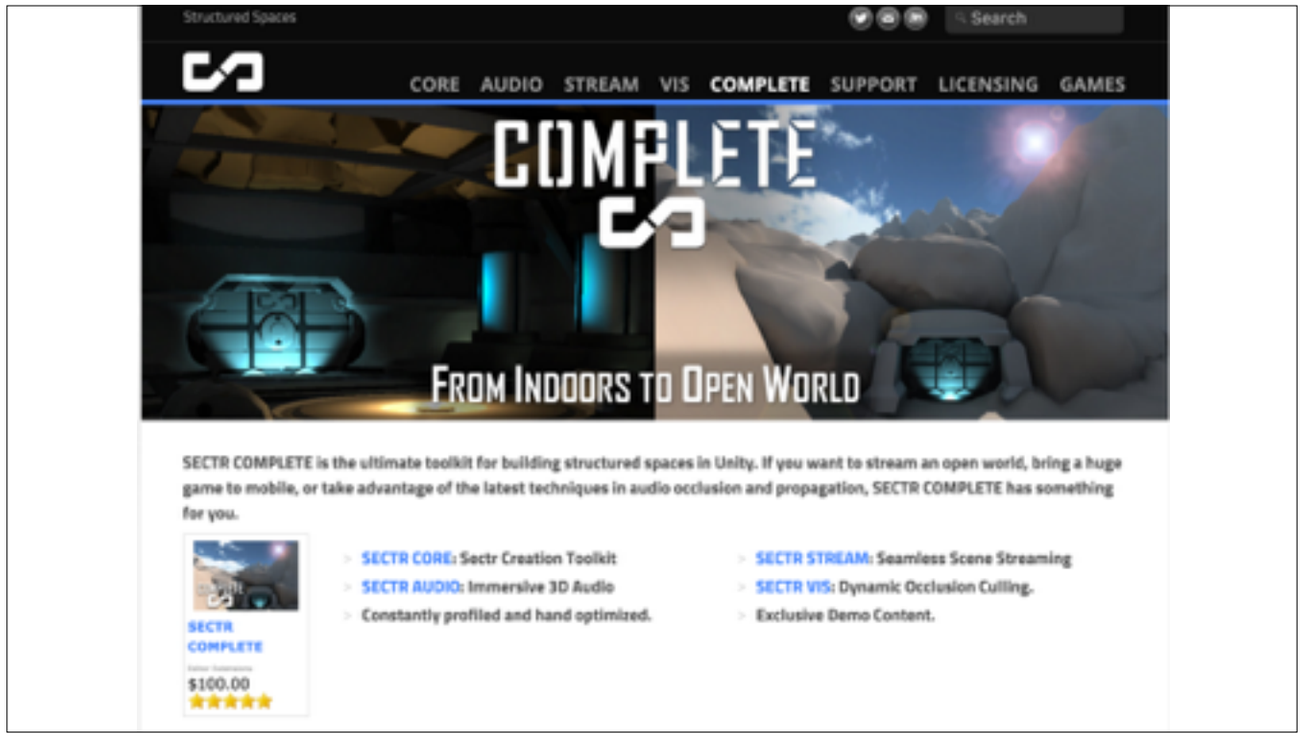
I asked around and there seem to be 2 types of approaches.

For Firewatch, neither of those approaches would work because
A) We need a level editor
B) we have 11 ppl and if we just take turns working on the game it’ll take 11 yrs to make this game.

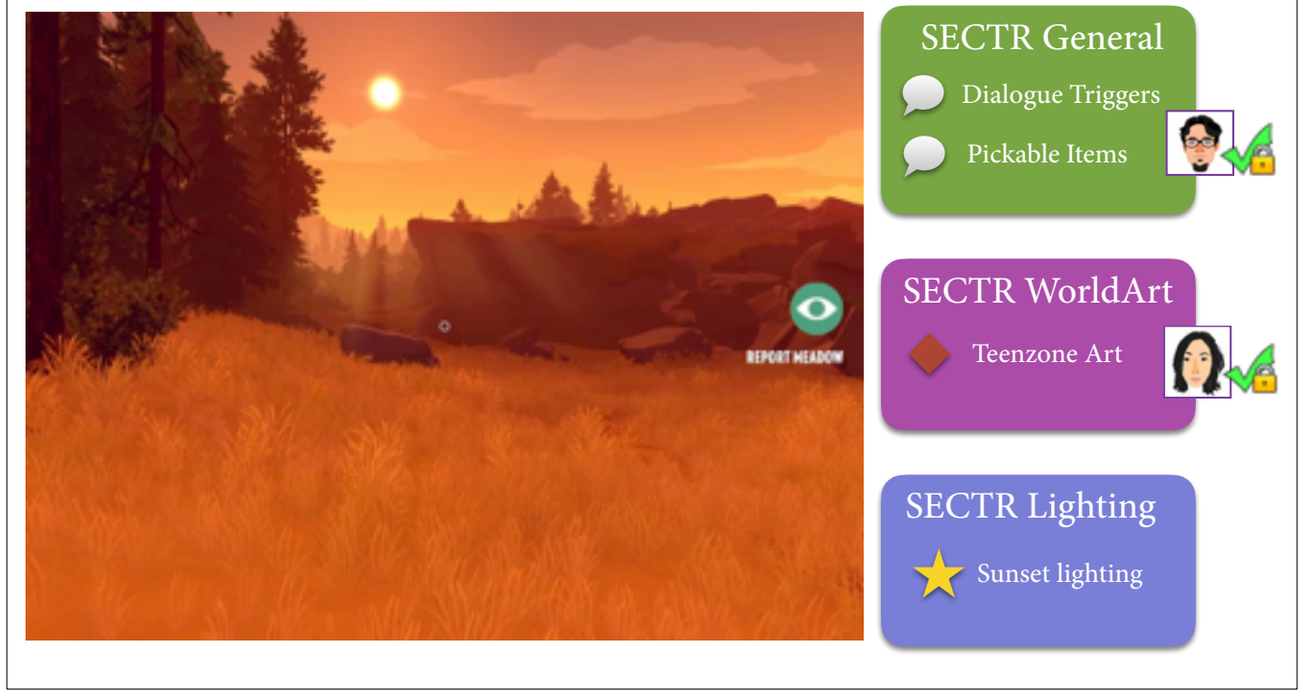


We know that most of the time, people only care about certain subset of game objects in a scene according their role.

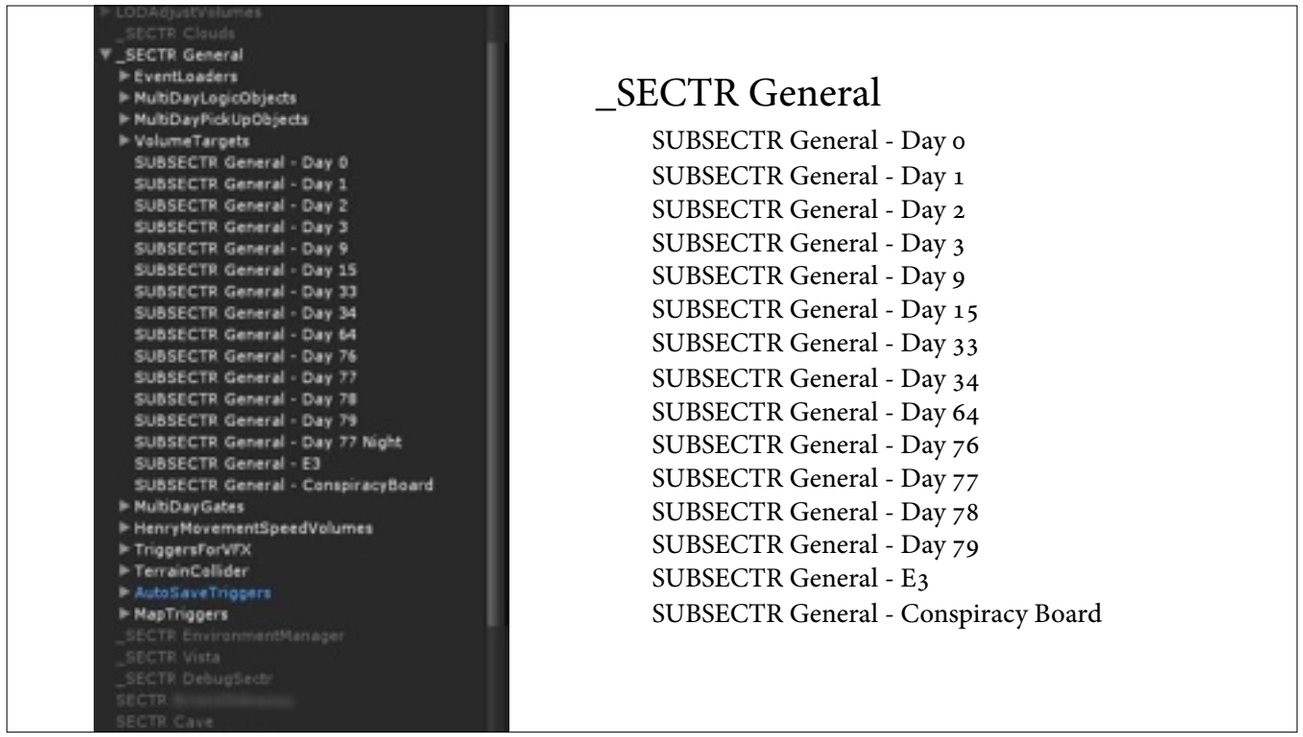
So it makes sense, if we can use a tool to split up our main scene file into smaller per-discipline type of files that can be worked on separately.



We used a unity extension called SECTR to do this. It is a tool that lets you easily separate chunks of data in your scene into data groups called SECTRs. Each SECTR can be worked on separately, and then they can be additively loaded back into your main level as necessary.



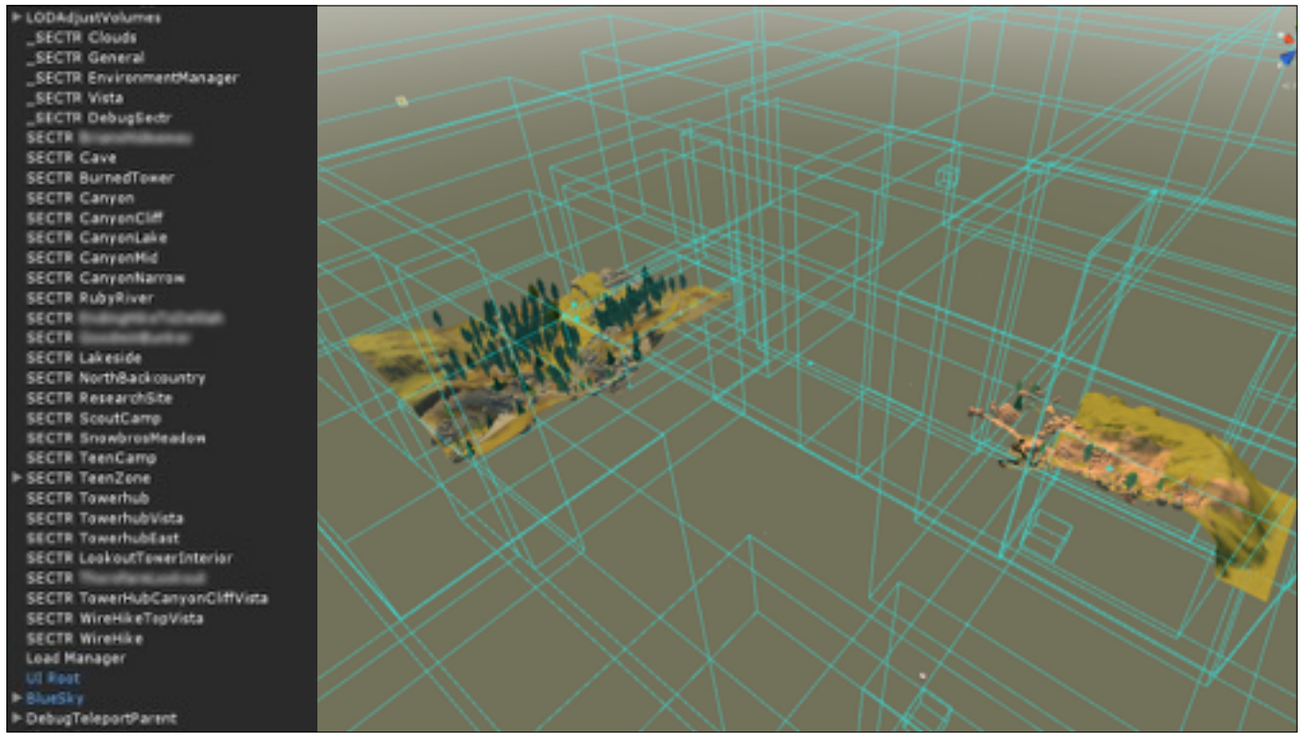
This means our unity scene files can maintain exclusive checkout and we avoid the whole crazy problem of unity scene file merge entirely. Individuals can also revert their own work without affecting others.



_SECTR General

- SUBSECTR General - Day 0
- SUBSECTR General - Day 1
- SUBSECTR General - Day 2
- SUBSECTR General - Day 3
- SUBSECTR General - Day 9
- SUBSECTR General - Day 15
- SUBSECTR General - Day 33
- SUBSECTR General - Day 34
- SUBSECTR General - Day 64
- SUBSECTR General - Day 76
- SUBSECTR General - Day 77
- SUBSECTR General - Day 78
- SUBSECTR General - Day 79
- SUBSECTR General - E3
- SUBSECTR General - Conspiracy Board

So once you start splitting up your data this way, there is no reason to not split it further to fit your specific workflow. For Firewatch, the game is divided into different days, so it makes sense for our design data to also reflect that different days.



For world art, it is split up spatially, so not only that Jake can work on one part of the world, while I move trees around in another part, also it handily facilitates world streaming. Which we will dive into in the next section.

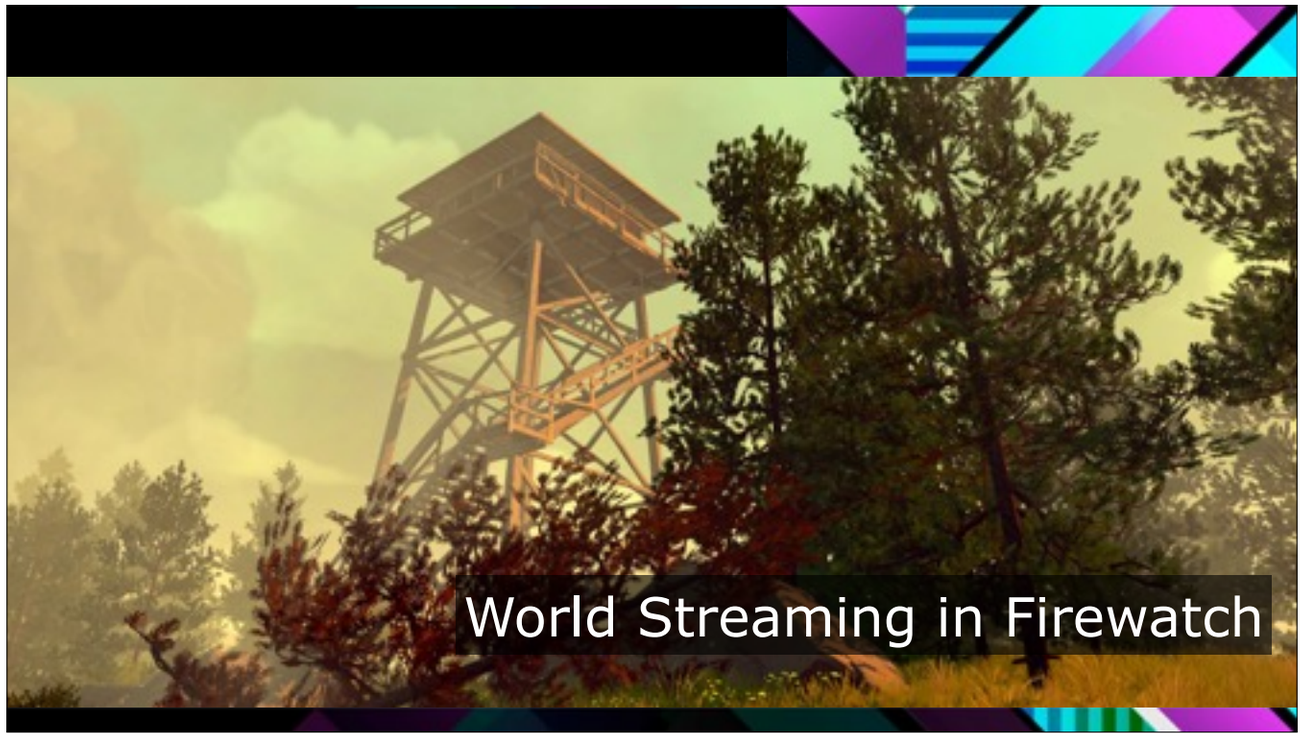
HOT TIP #1

Figure out ways to work in parallel

No matter what your engine you use, try to arrange work so the team can work in parallel and each person can iterate without unnecessary dependencies.

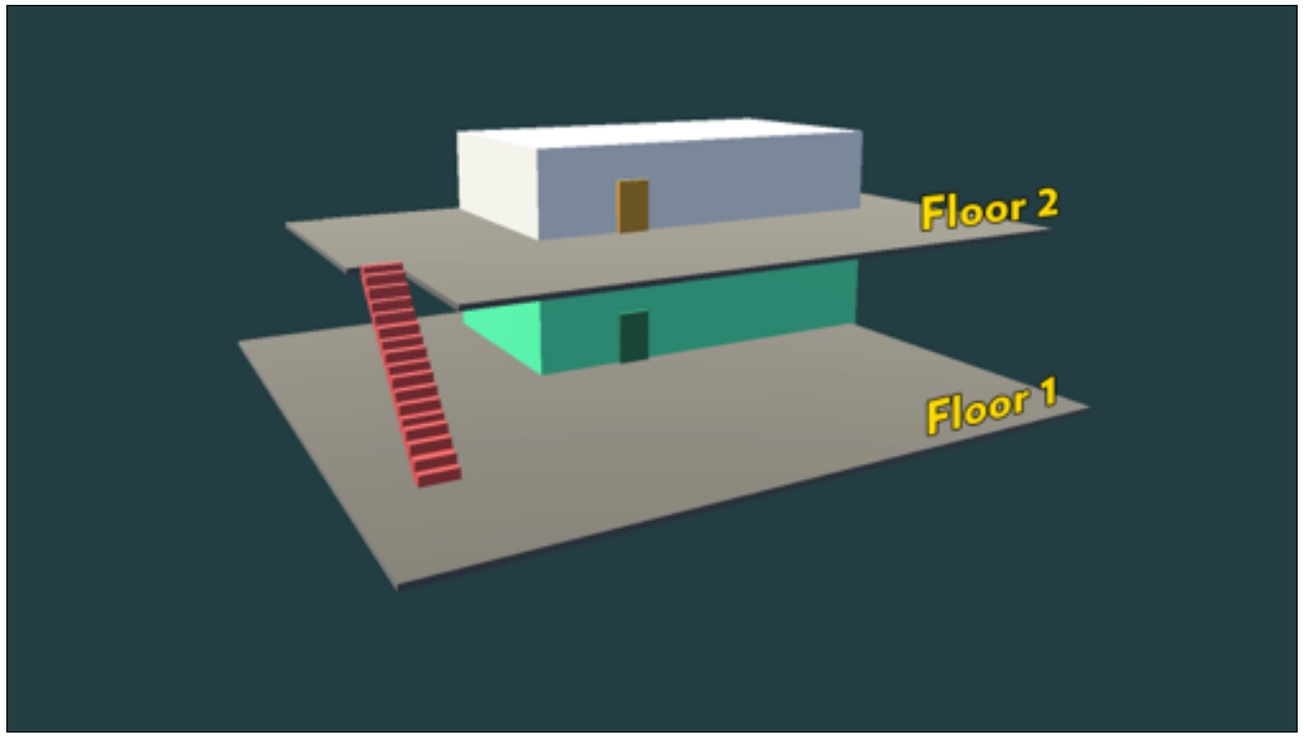
ABSOLUTELY structure your game data to allow for easy concurrent work!

The important takeaway here is that you should think hard about how to split up your game data, to allow for simultaneous work.

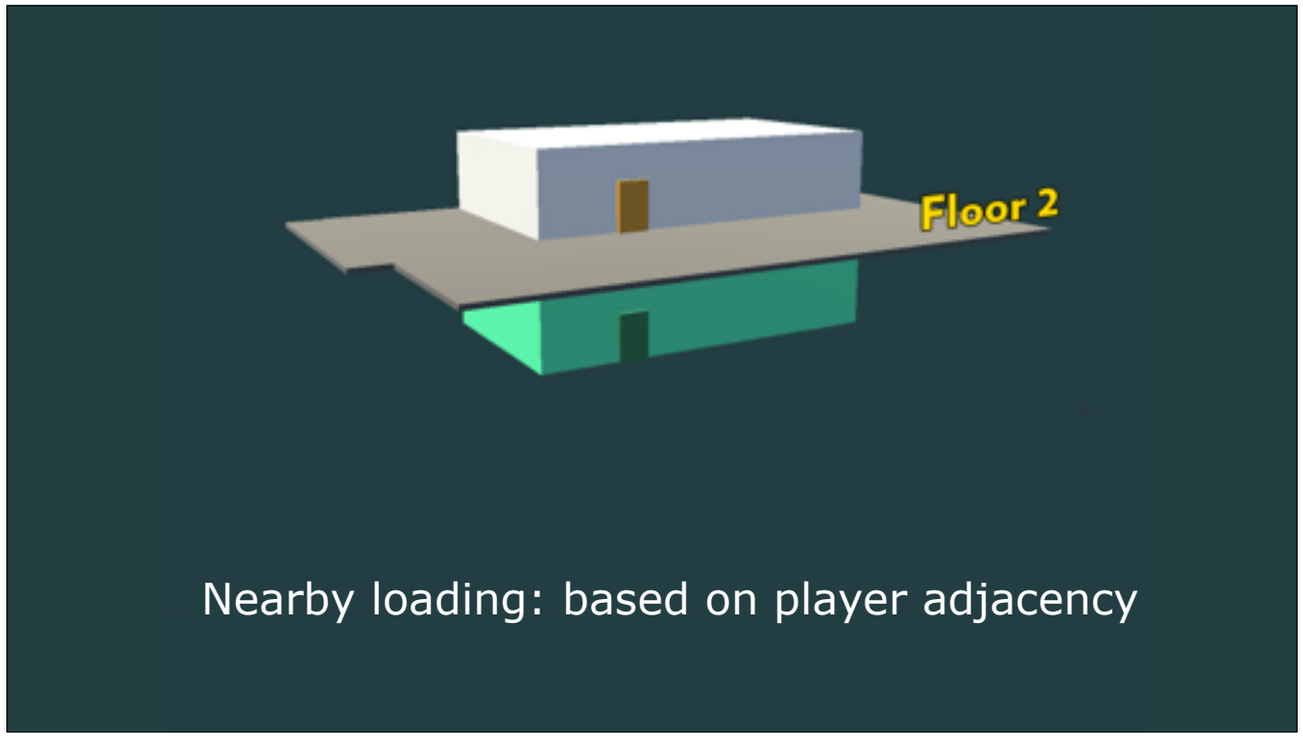


One of the major design tenets of Firewatch is that it is open-world-ish it should exist in a contiguous world, with no pause+load for the player when they are just traversing the forest.

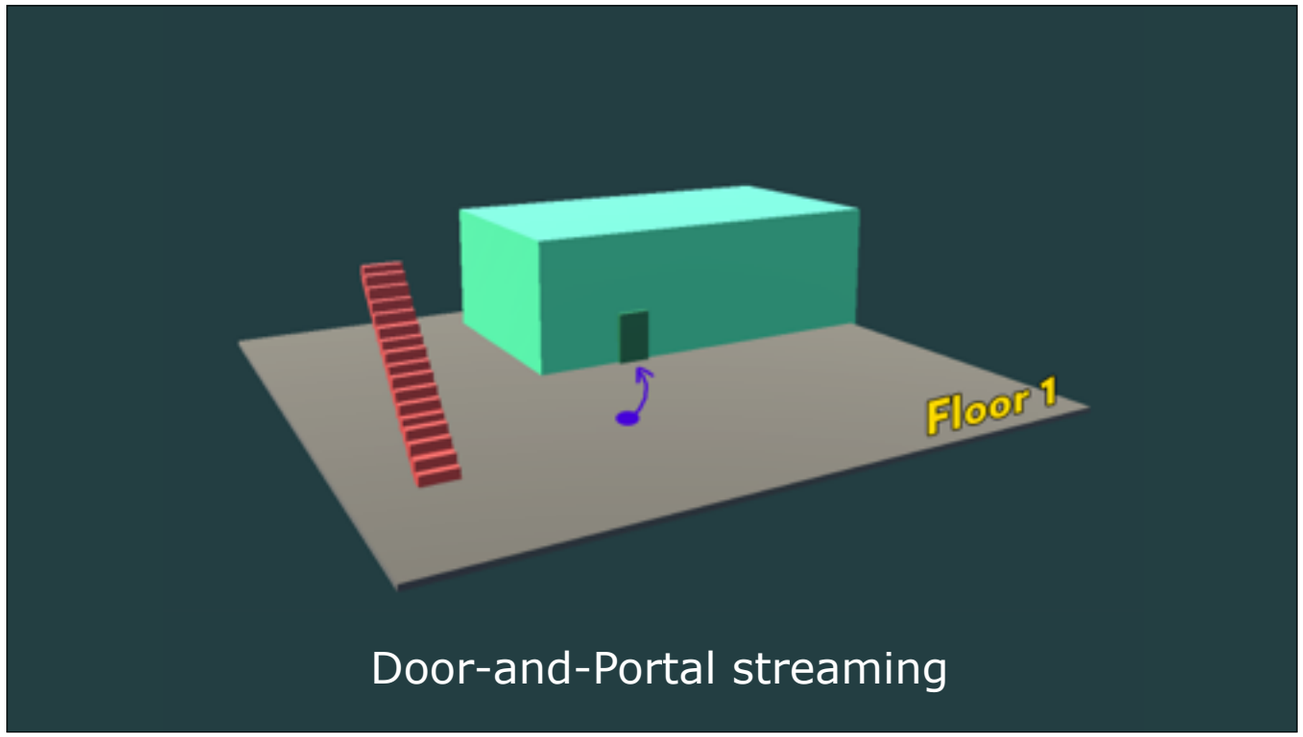
The only way for Firewatch to be at all performant is to utilize streaming.



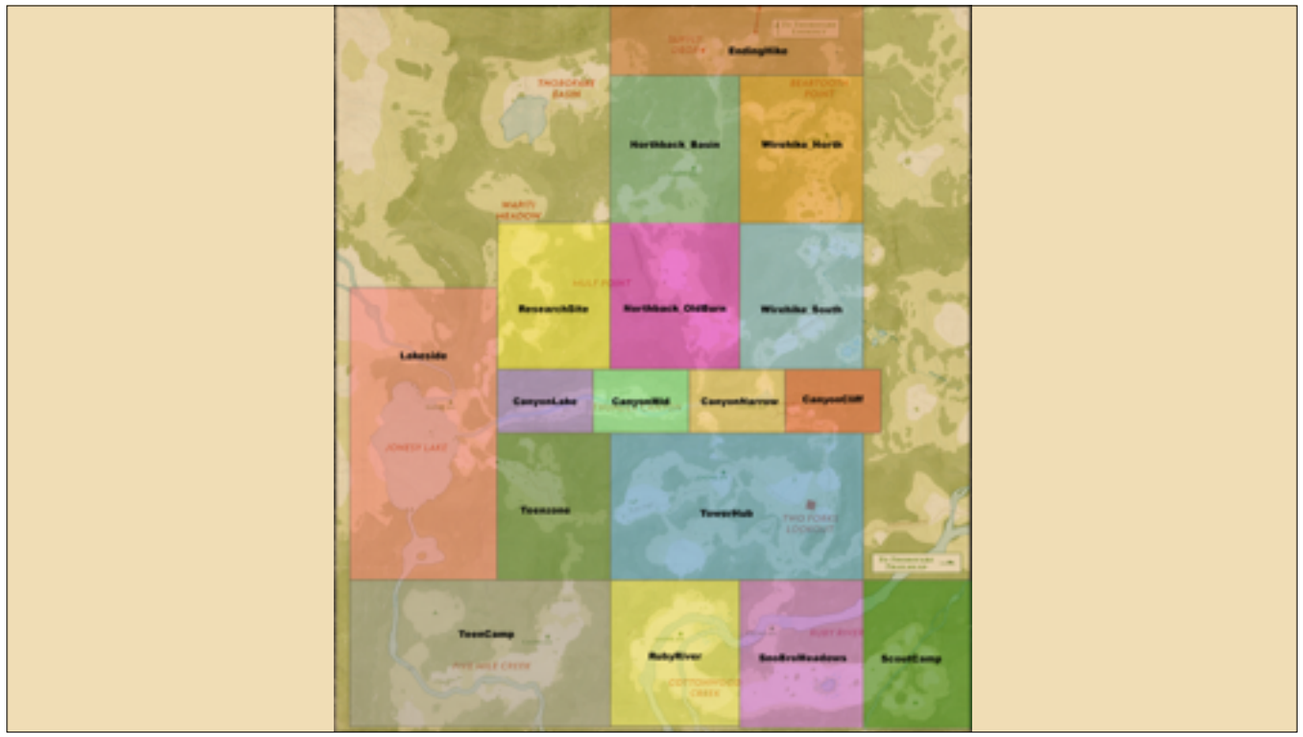
Streaming is a way of selectively loading in game assets or “stream in” data that you need at that point.



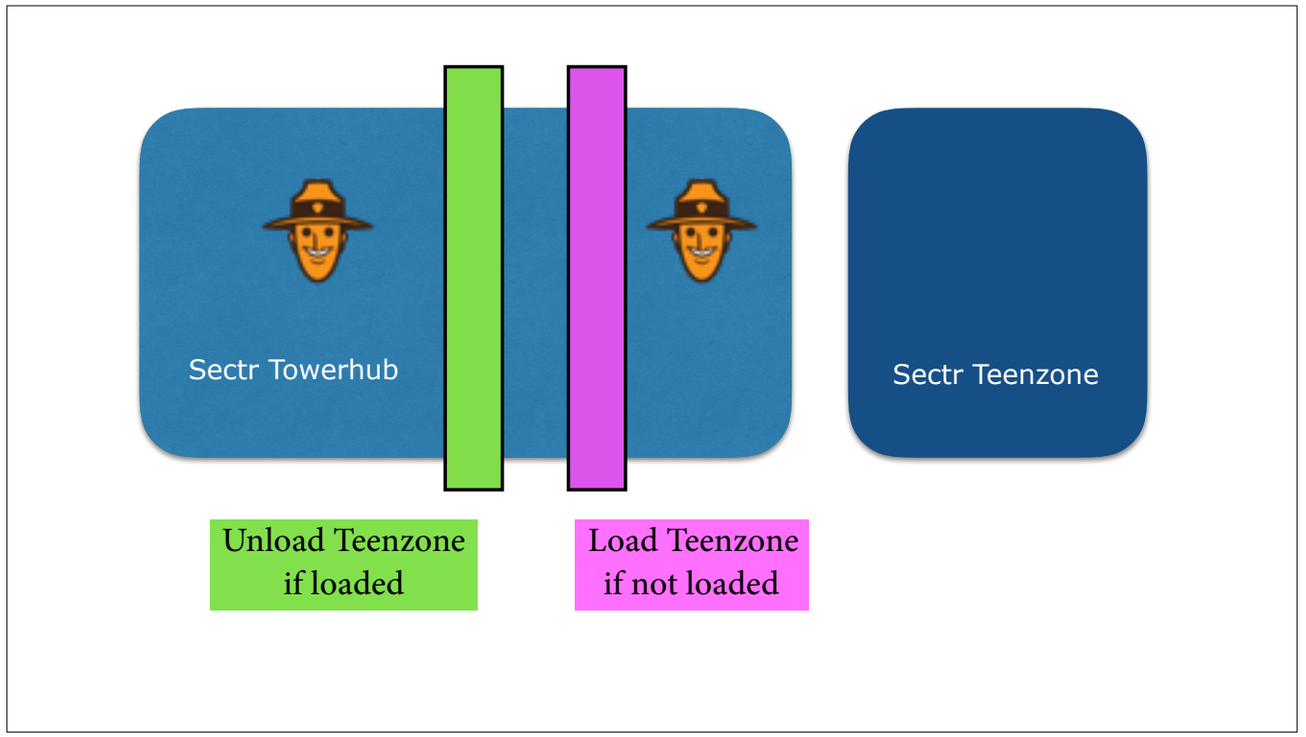
So there are different ways to control streaming.
For example, the simplest form is “Nearby loading”, which is loading purely based on player adjacency.



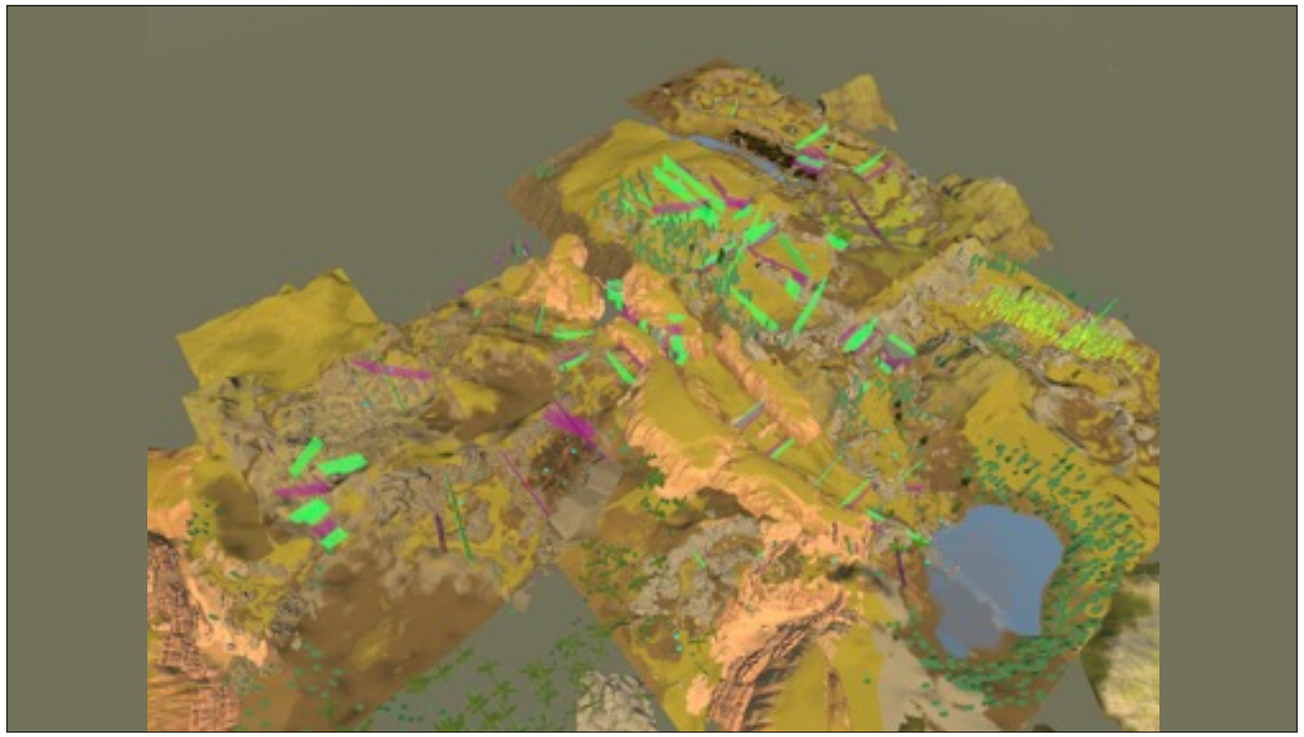
A slightly more sophisticated way is door and portal streaming. You tell the game how all your spaces are connected, which is smarter.



What does it mean to do streaming in a open environment like Firewatch?
Remember we already split the world up into different SECTRS, for concurrent work purposes, we also use this to facilitate our world streaming.

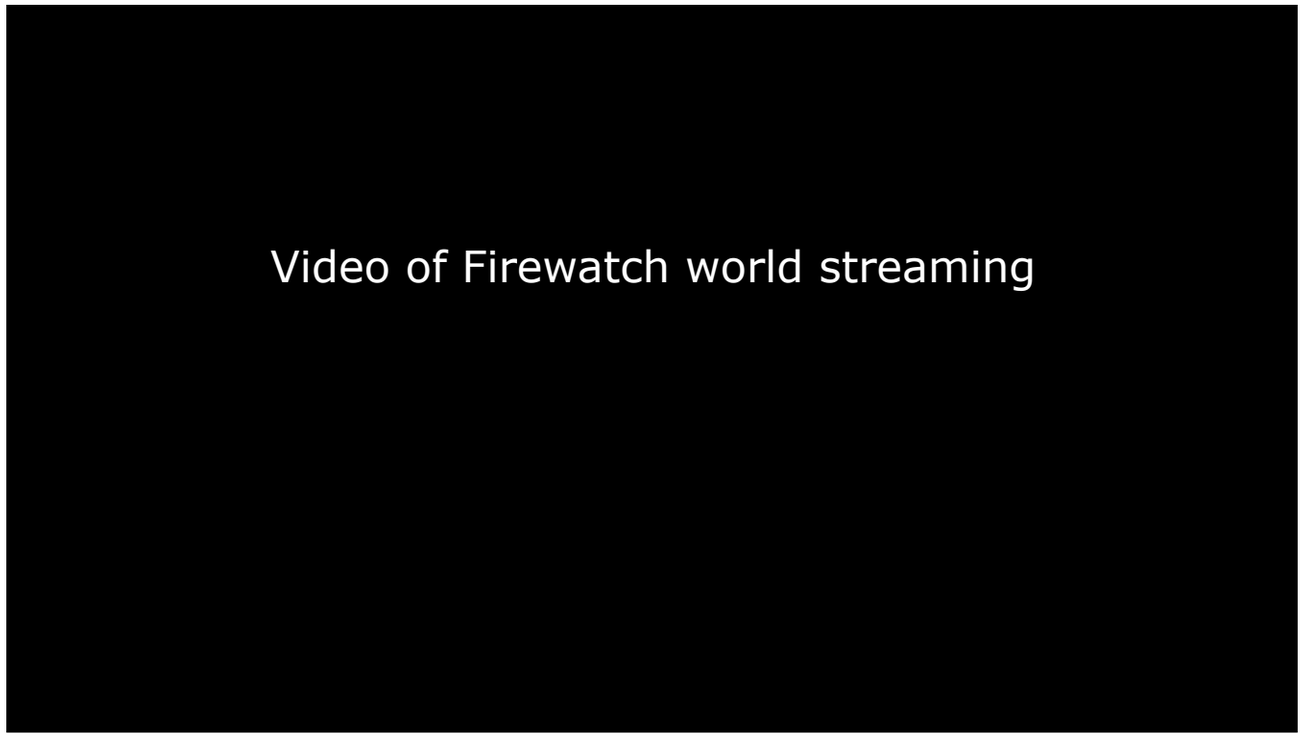


There are no doors in a national forest but we created the concept of one by using these load/unload volumes. The pair of them basically act like a toggle switch.



This is the set up for all the DOORS scene streaming logic in Firewatch. The purple volumes are “Load” doors, and the green volumes are “Unload doors”.

So I’m going to show you a little video of how firewatch world streaming works in the game. Before we start I’m going to show you the path where the player would go, just so you have a bit of an idea.



<video>

Another cool point about world streaming is that, you can use this to swap in different versions of the world given different requirements.

HOT TIP #2

Utilize streaming!

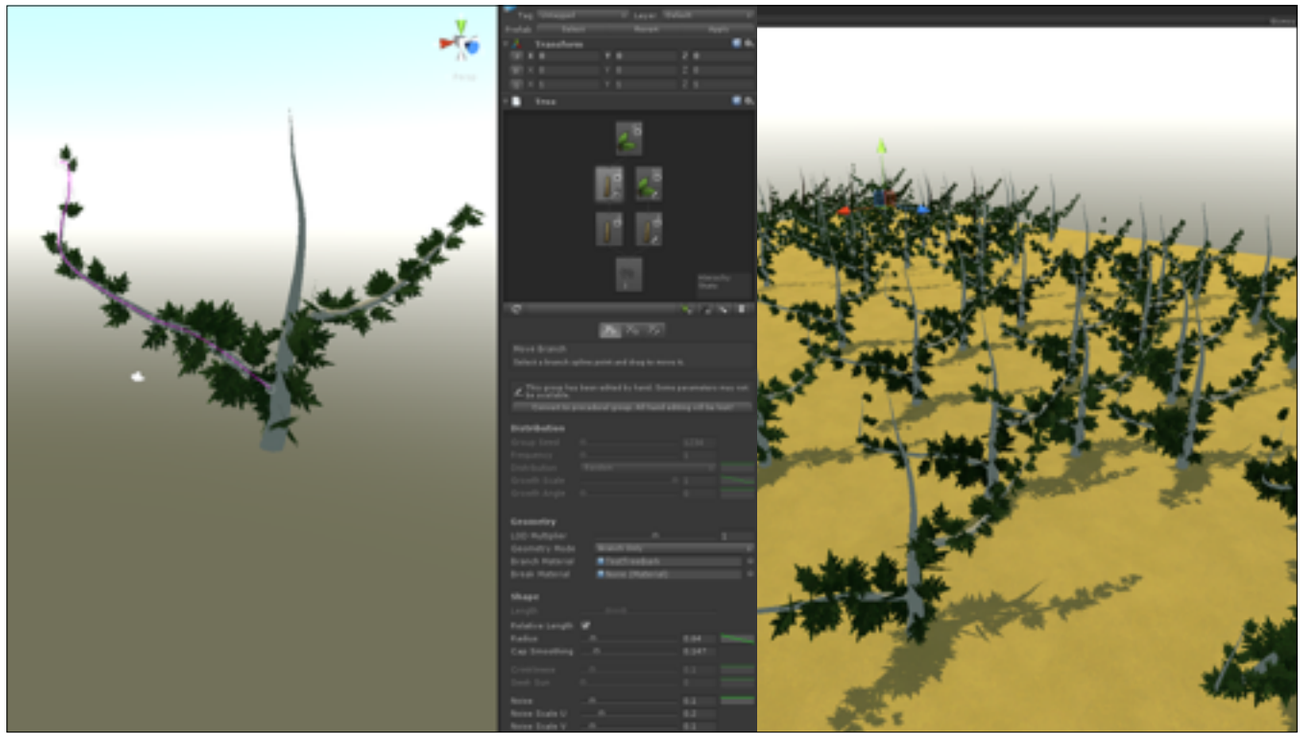
Streaming is easy to set up with the right tools and it will let you add a lot more assets while keeping performance under control!

There is no reason to load in assets you do not need!

Streaming is not just for big environments, in fact if you have a game with walls/doors it is much easier to set up. With the right tools even a small team like ours can set it up so you should really look into it!

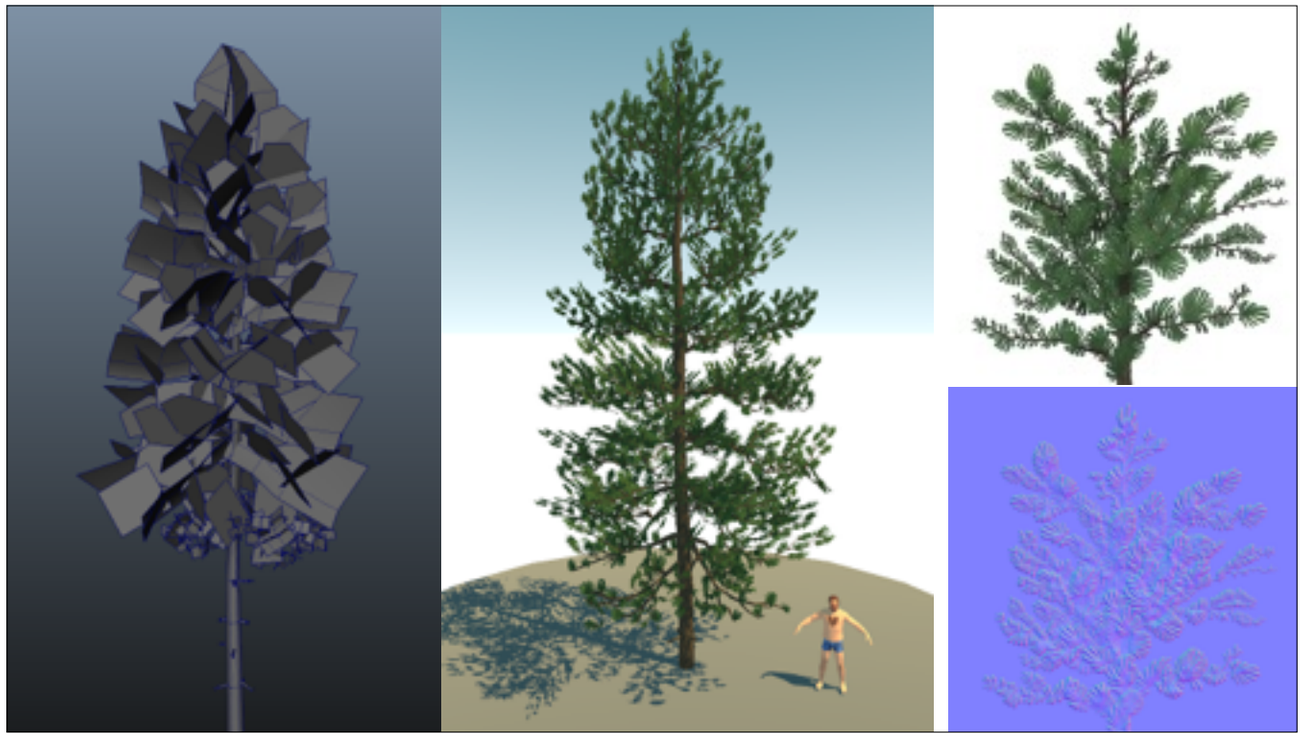
Ok now let’s switch gears and talk in depth about some environment art assets.

We started developing Firewatch using Unity 4.5, Speedtree was not available in 4.5 so we never evaluated it. Our only options were to use Unity Tree creator trees, or make our own custom trees.

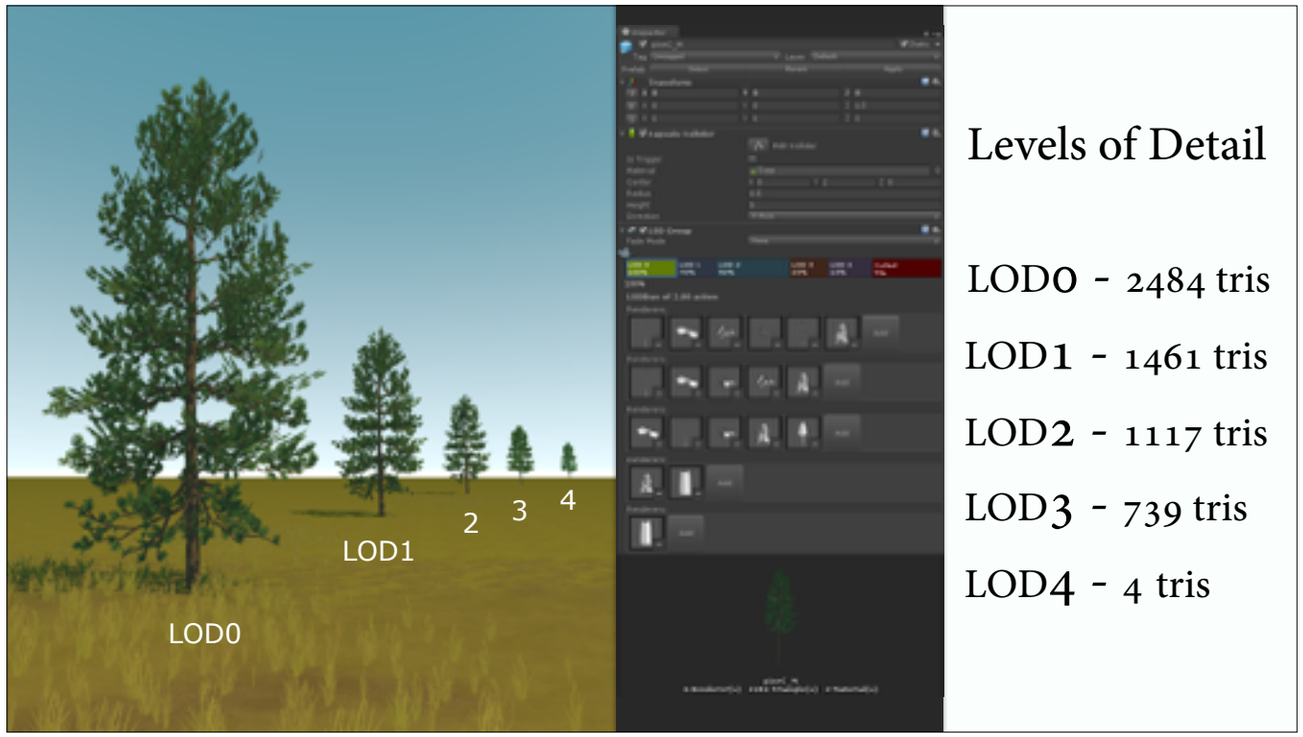


Unity comes with their own custom tree editor, and it creates trees that can be painted onto the terrain. They were quite limited. No rotation on placement, you can't tilt them etc. We decide to manually created trees, because we can have more control.

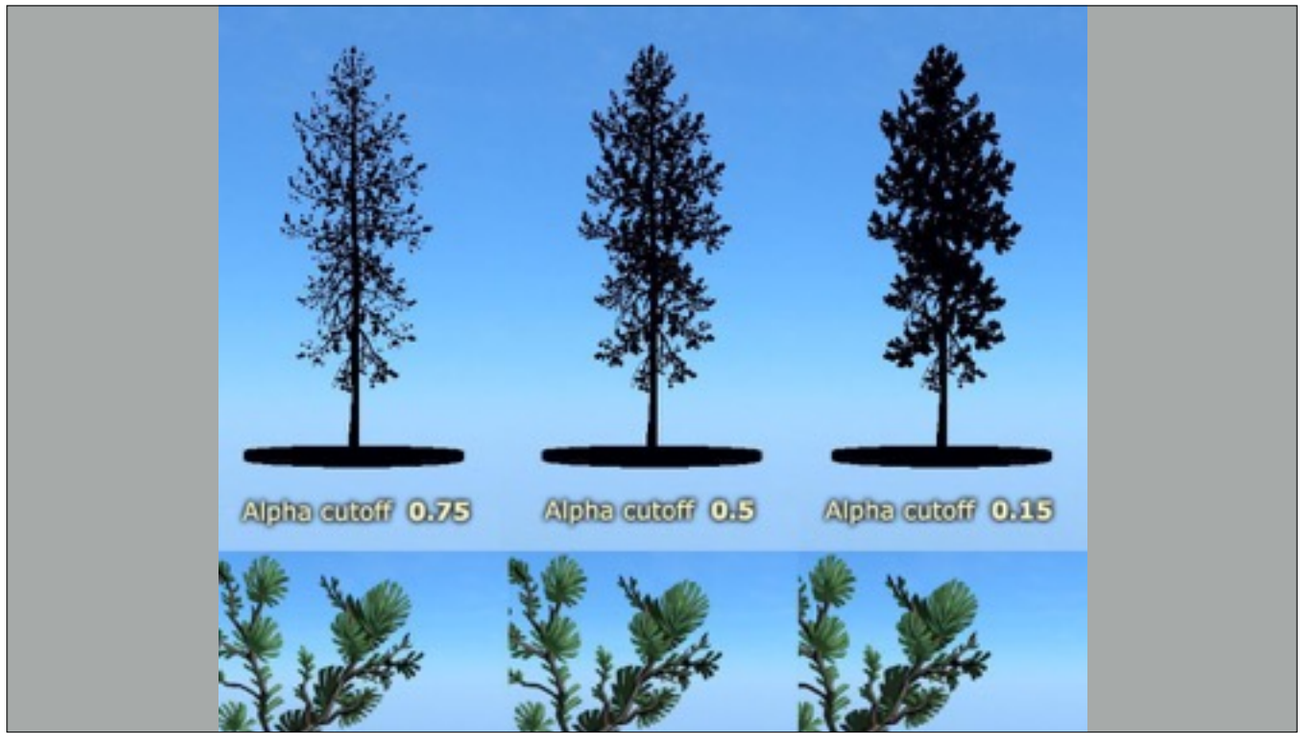
This is a lineup of all 23 different trees we made for the game. Though 9 of those, the aspens and the cottonwoods on the bottom, were kind of unique to only a couple of areas. The 14 in the top row were used all over the game.



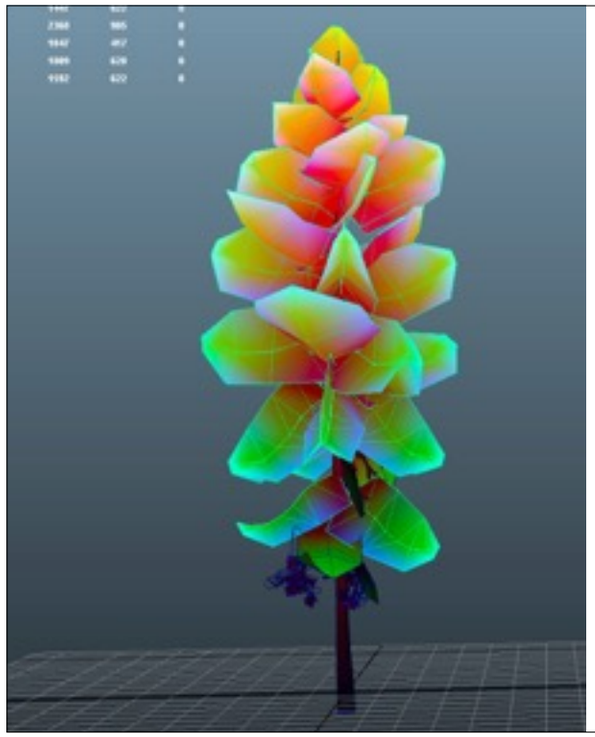
On the left: Maya view
Middle: Unity view
On the right: leaf textures used



Here we have Pine C’s LOD lineup. This is about at “Medium” setting on the PC. The major difference between LOD1 and LOD2 is mostly about dropping some shadowing casting geo.



We have a special shader that pushes out the alpha testing of the trees to make them more stylized. So basically the further out the tree is from the player, the alpha cutoff is lower. This makes the tree more puffy looking when it's far.



Foliage wind response

Red is vertical sway

Green is branch “flap” movement

Blue is leaf flutter

picture of wind manager

So to make our own custom trees, we needed to also create a wind response system. We use vertex color set in Maya to define wind response for each tree.

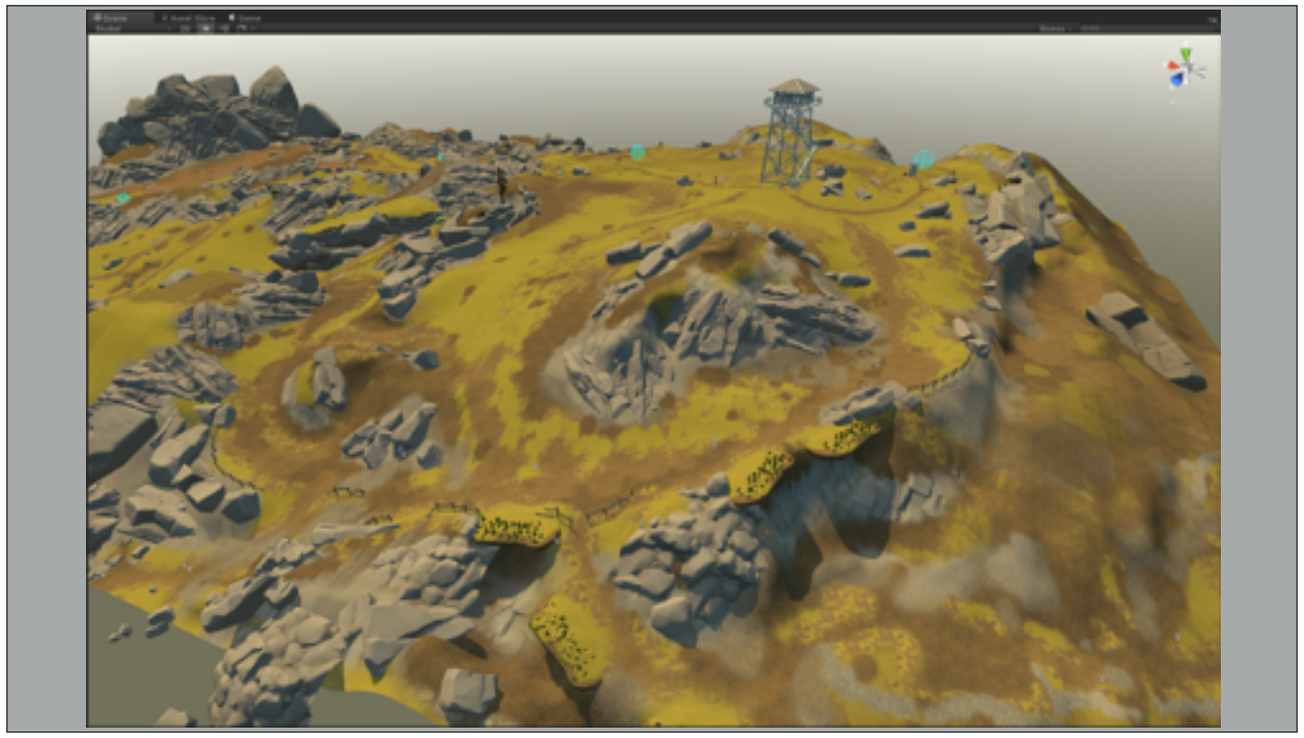
HOT TIP #3

Model variety is overrated!

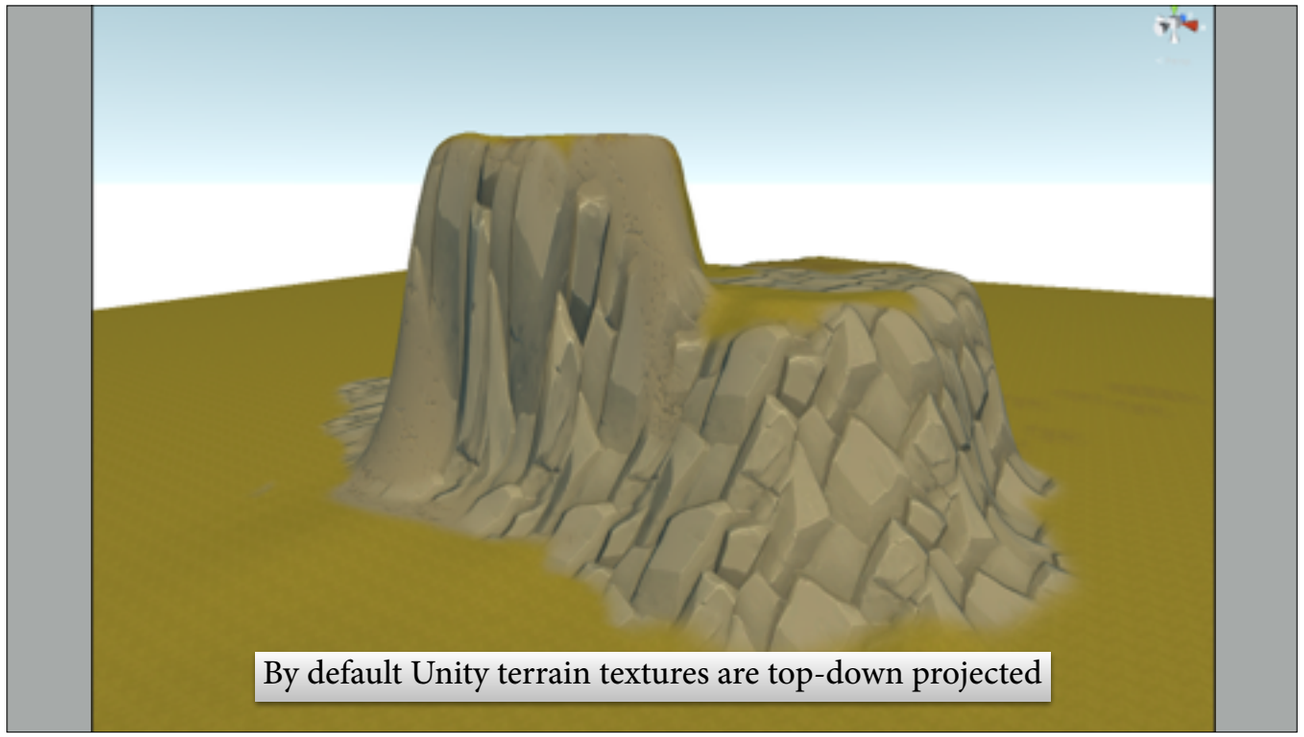
It is better to have 10 really well made models that you can duplicate 20 times in varied ways, than 200 unique assets for “variety”.

Think modular! It’s not just for dungeon levels!

Artists like to make art, no question about that. But don’t just make tons of models because it’s fun. The more models you have the more maintenance you will have to do, like adjusting LODs is a pain.

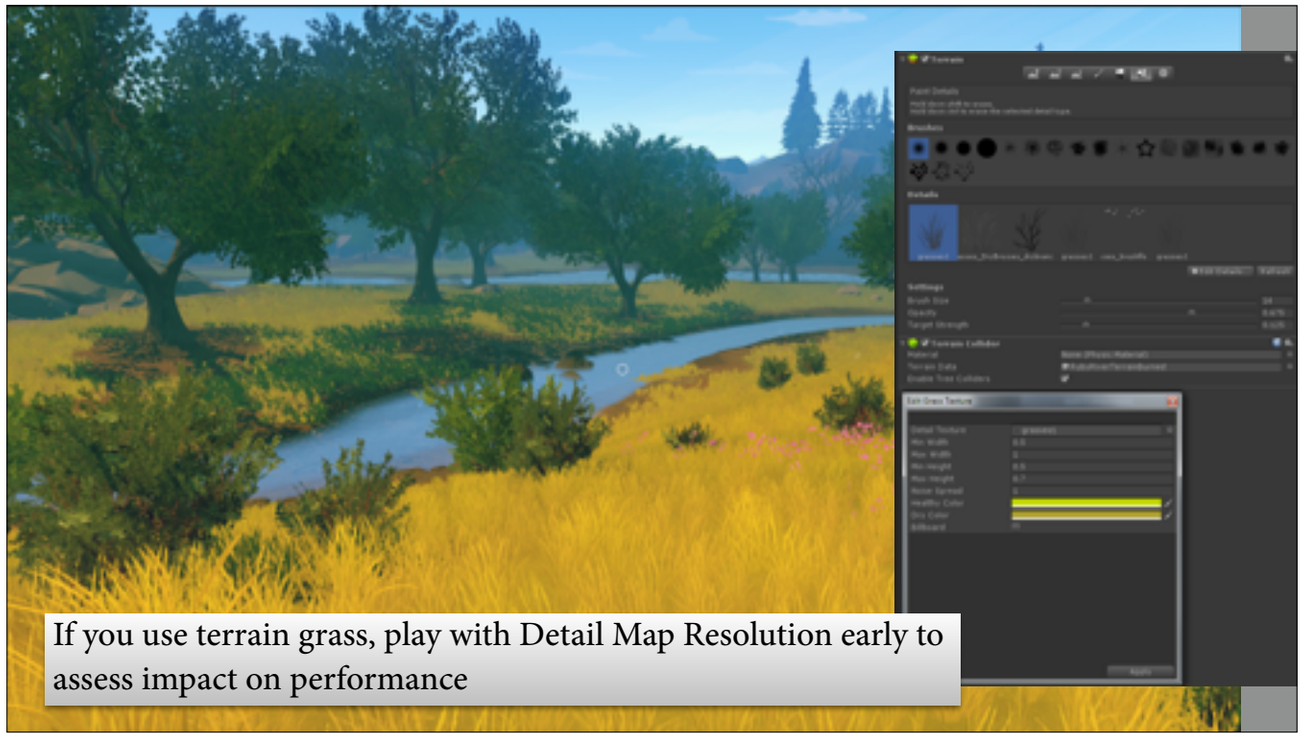


Unity terrain is quite easy to work with in the editor.
However, there are quite a few things you should watch out for if you using it as your in game asset.

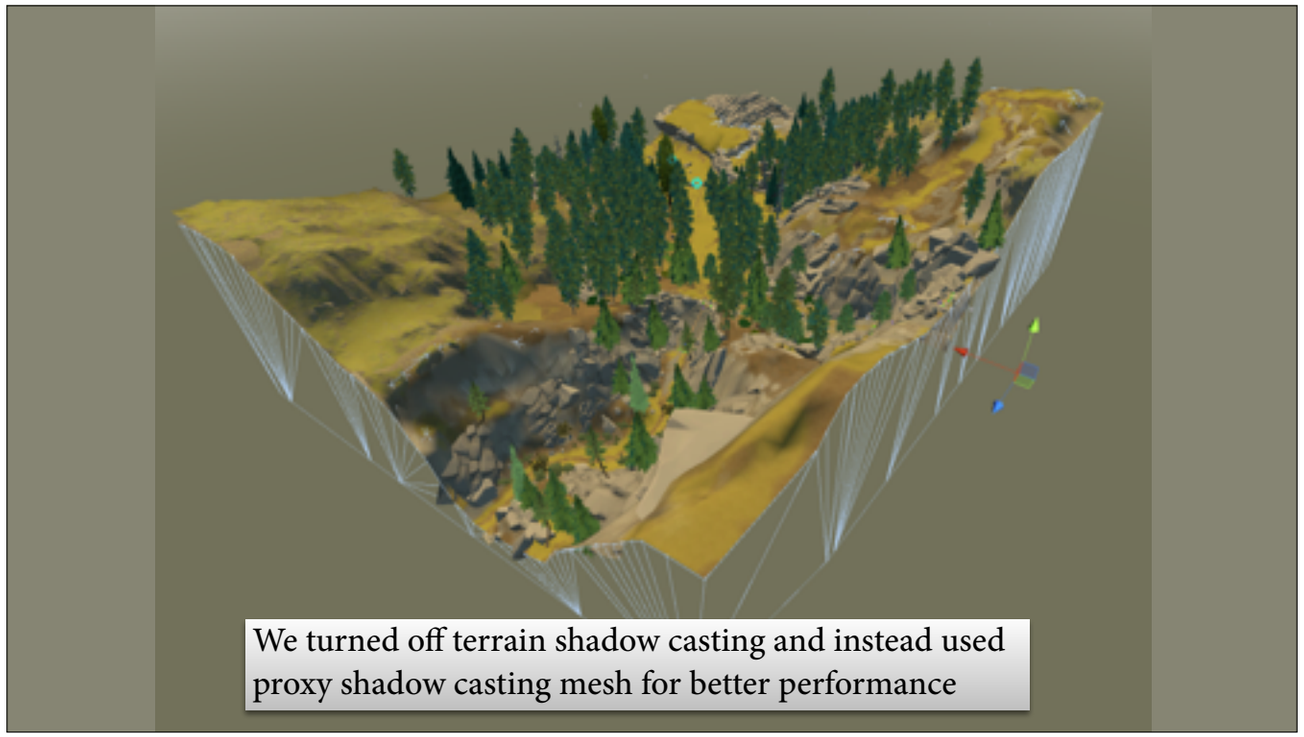


1) As you can see in this picture, if you have a lot of steep cliffs, the UV stretching becomes really problematic.
2) we tried adding our own box projection, but then the terrain had to draw twice and that was not performant.

So we worked around this by layering rock meshes on top of areas with a lot of stretching, but very labor intensive.



Unity terrain grass, like other aspects of the terrain, is very easy to use.
However it is not quite clear how different grass settings affect performance.



Unity uses the visual mesh to also do the shadow calculation, so this could lead quite an expensive shadow time. This means we ultimately turned off shadow casting on terrain and made our own cheap proxy mesh shadow casters underneath.

HOT TIP #4

Do a performance analysis earlier rather than later

You don’t have to to optimize your art too early, but it is still good to get an idea of HOW optimizable each art system is before you go into full production.

This might seem obvious but definitely do a performance capture earlier rather than later to see just what might be a problem. I think we underestimated the challenge of optimizing our terrain until we were past the point of doing something custom about it.

Now let’s talk about some lessons learned while doing world design for Firewatch.

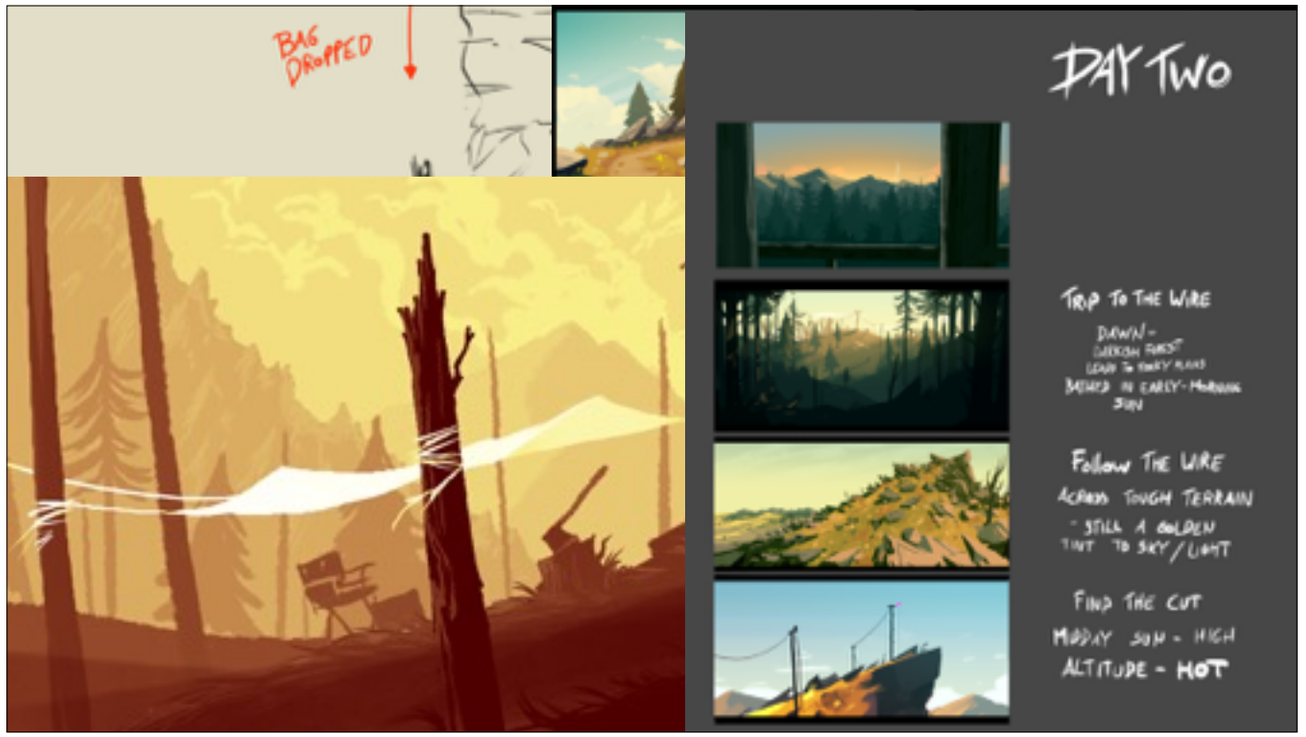
The world design in Firewatch has to achieve two goals:

1) It is a narrative game so level design has to guide the player from story point to story point without too much handholding

2) The world has to feel like a big interesting forest to explore, while being not actually so big that we can’t make it in under 2 yrs.

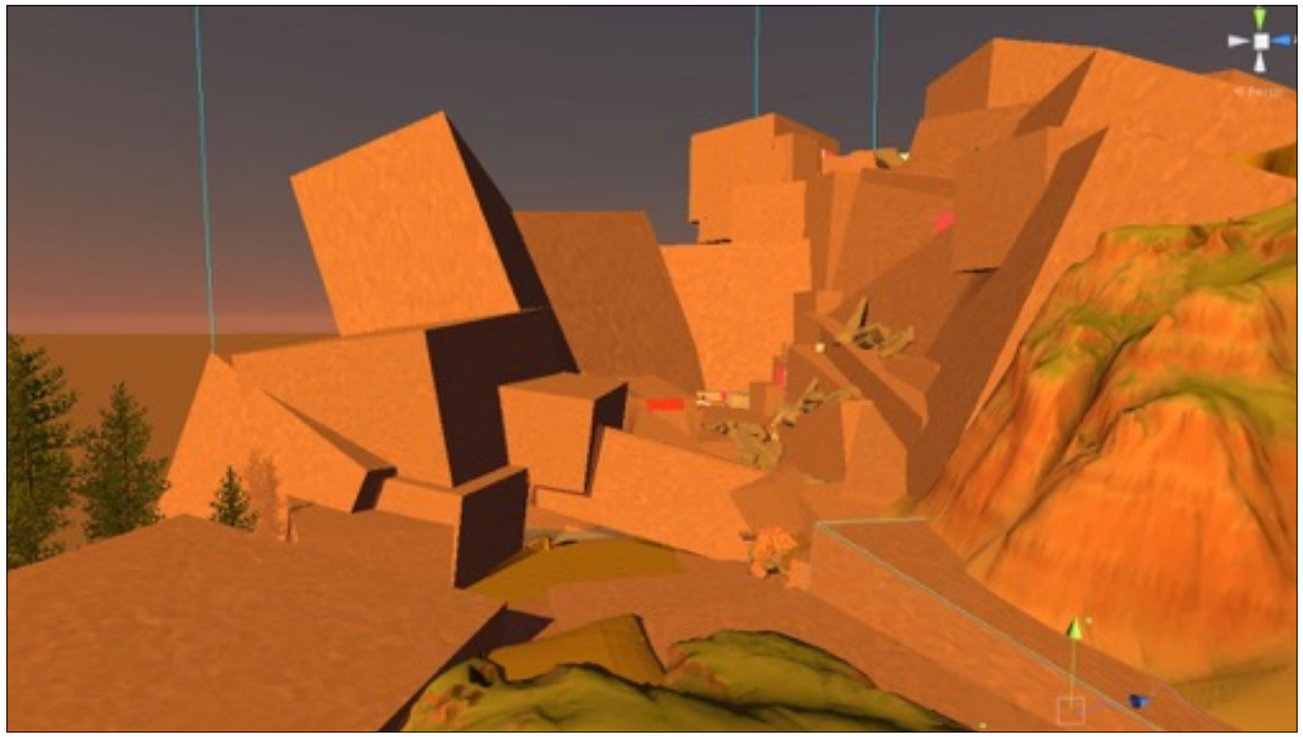
So we started designing the world as the story was being developed. The goal here is that each area in the game serves a narrative purpose.
We just start with paper maps.

While Jake and the story folks are figuring that out, Olly was doing
visual development of different locations the game could be set in, in general.



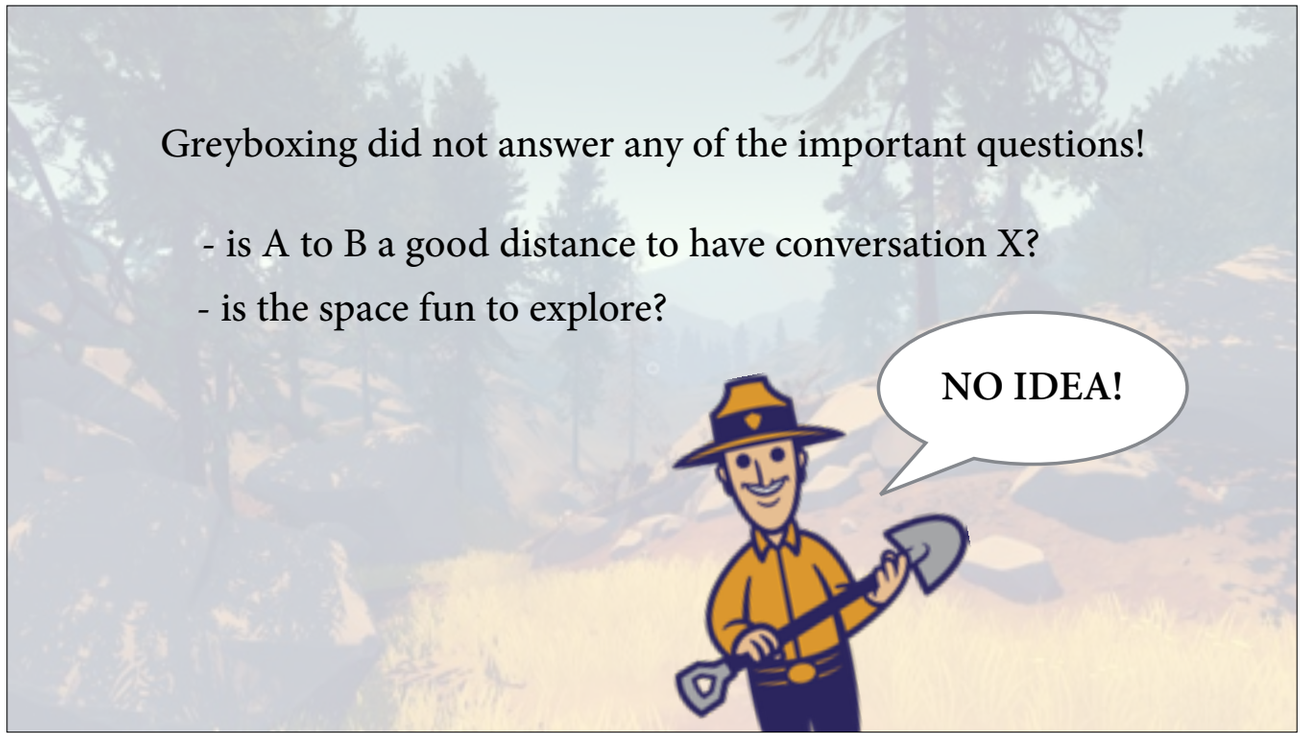
And also more detailed concepts for very specific story moments called out during story development.

We have a good bunch of concept art, and also a paper map.
But concept art and maps don't make equal a 3D space.



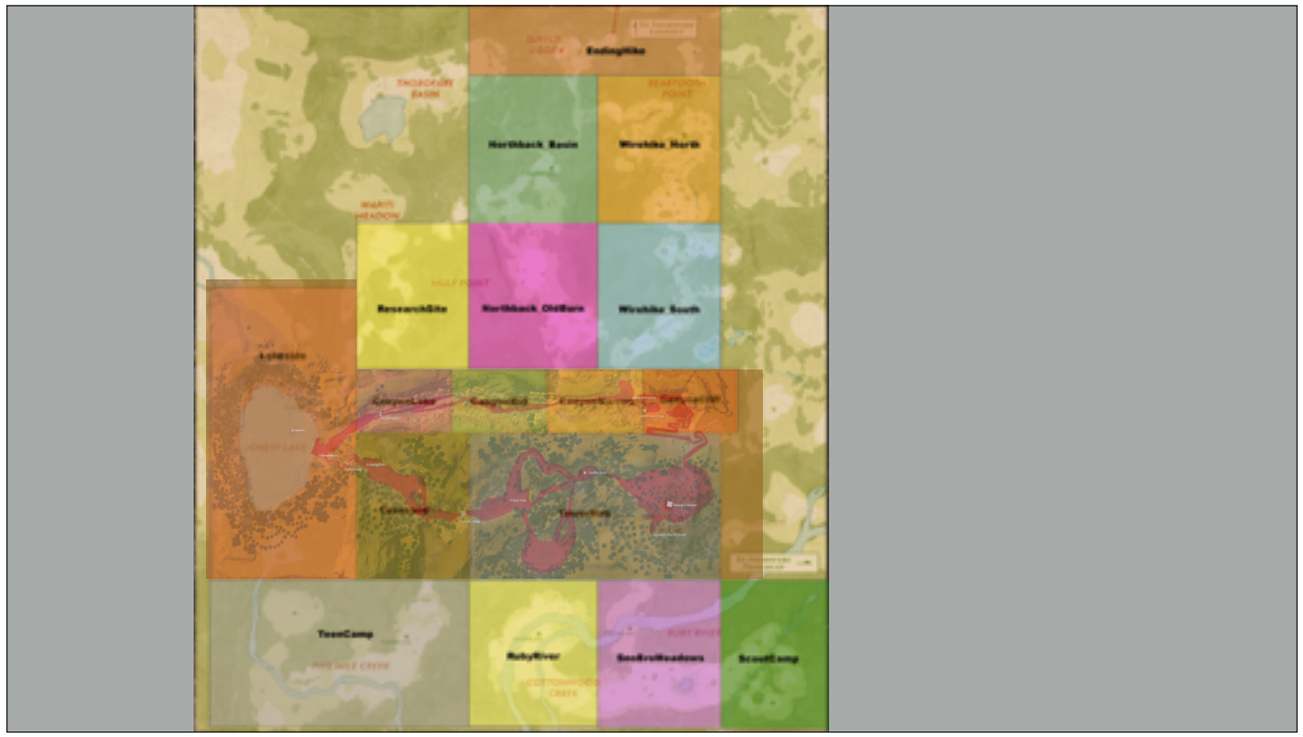
The traditional way of figuring out 3D part is by greyboxing!
Literally just mock out your space with some sort of cubes.

Greyboxing… just.. didn’t quite work for us. Imagine walking from Point A in this grey box. to Point B. With some subtitled Writing. It is going to be a boring eternity.



This is obviously a problem because Firewatch is about walking through a beautiful forest having interesting conversation. The greybox answered zero questions.

To really test our ideas out, we realized we needed to do a vertical slice of the game to really representative, with good enough art and actual VO to be able to judge anything. But that took 15 months.



By the time we had our vertical slice, which we showed to the public in March 2015, world art wise it was only about 1/3 of the game world.
Considering we wanted to make the game within 24 months this is stressful.

But now we know at least, “the distance from Tower to Lake” feels like X, and we used that as a solid yardstick.

Concept art of teen camp

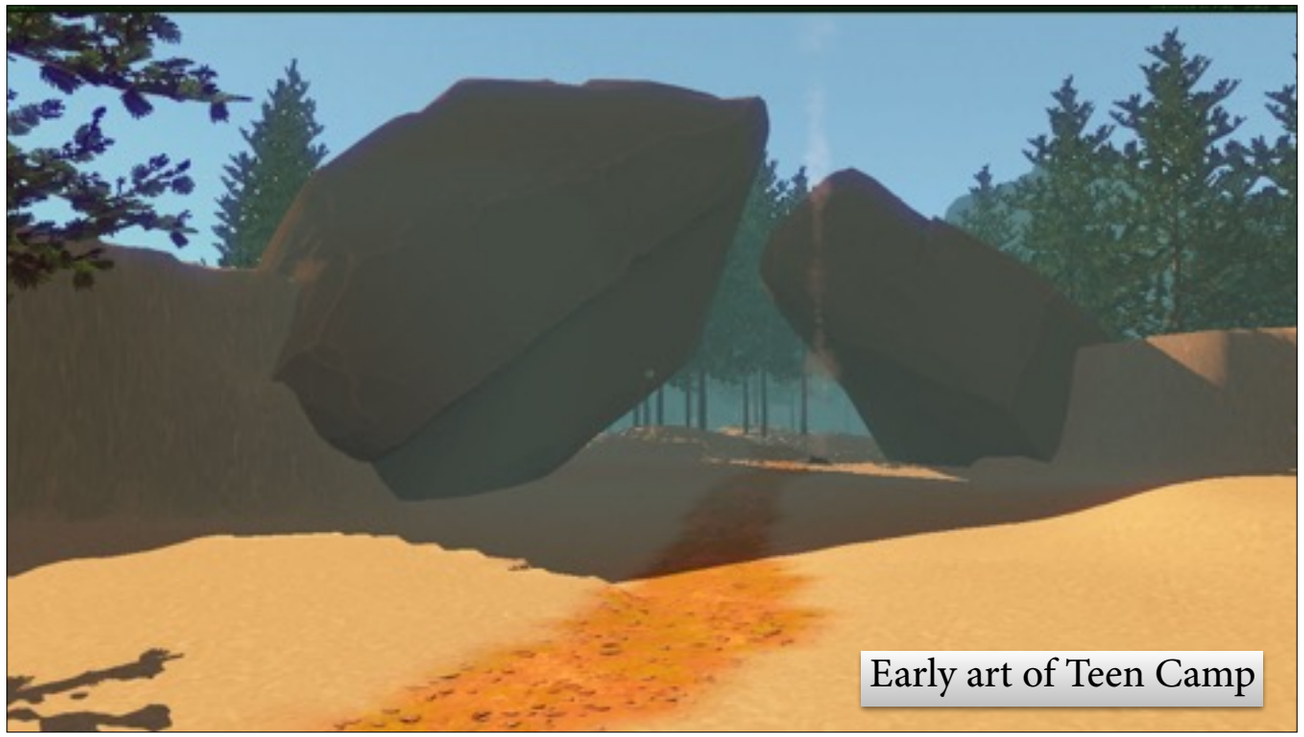
So we picked a bunch of Olly’s pool of env concepts and developed them into “cool story landmarks” to start anchoring spaces.


Early art of Teen Camp

And then we basically scrapped greyboxing, and switched to mocking out “cool story landmarks” around the map, using our knowledge from the Vertical Slice to best guess what kind of distance made sense between them.

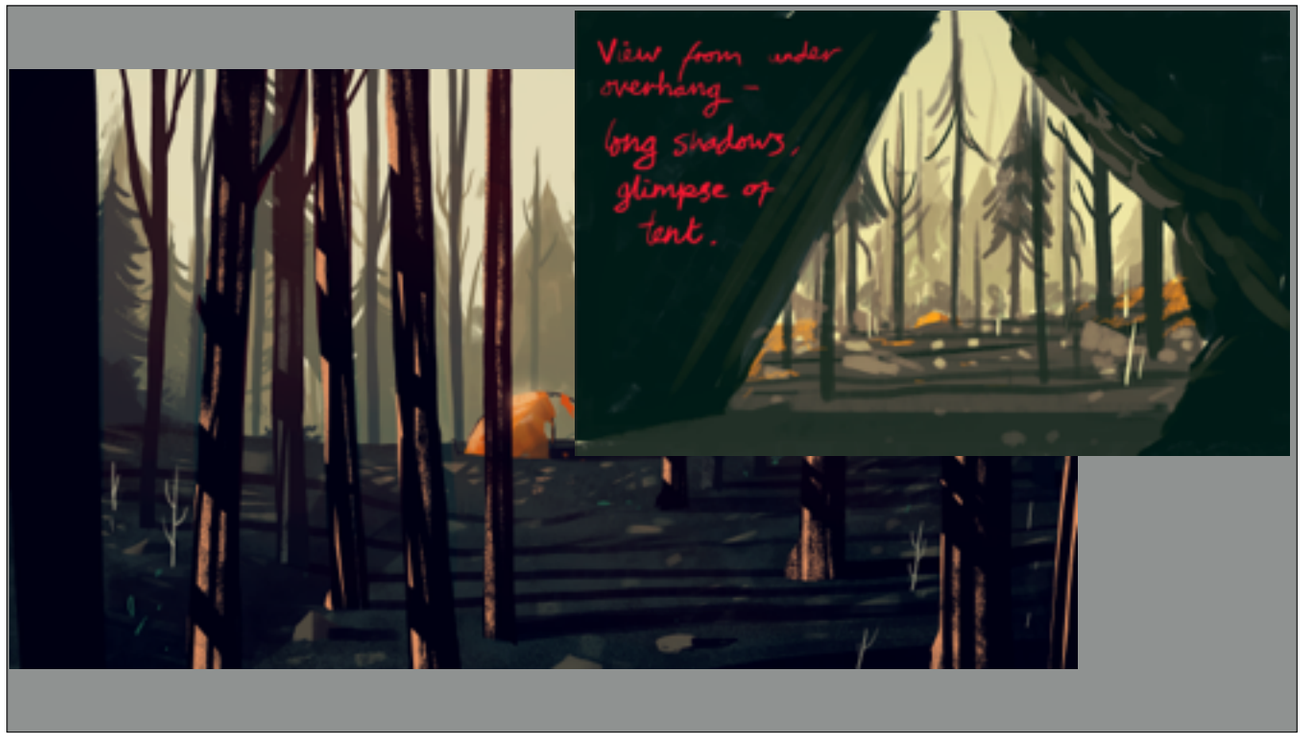
Final art of Teen Camp

Concept art of teen camp



Early art of Teen Camp

Final art of Teen Camp

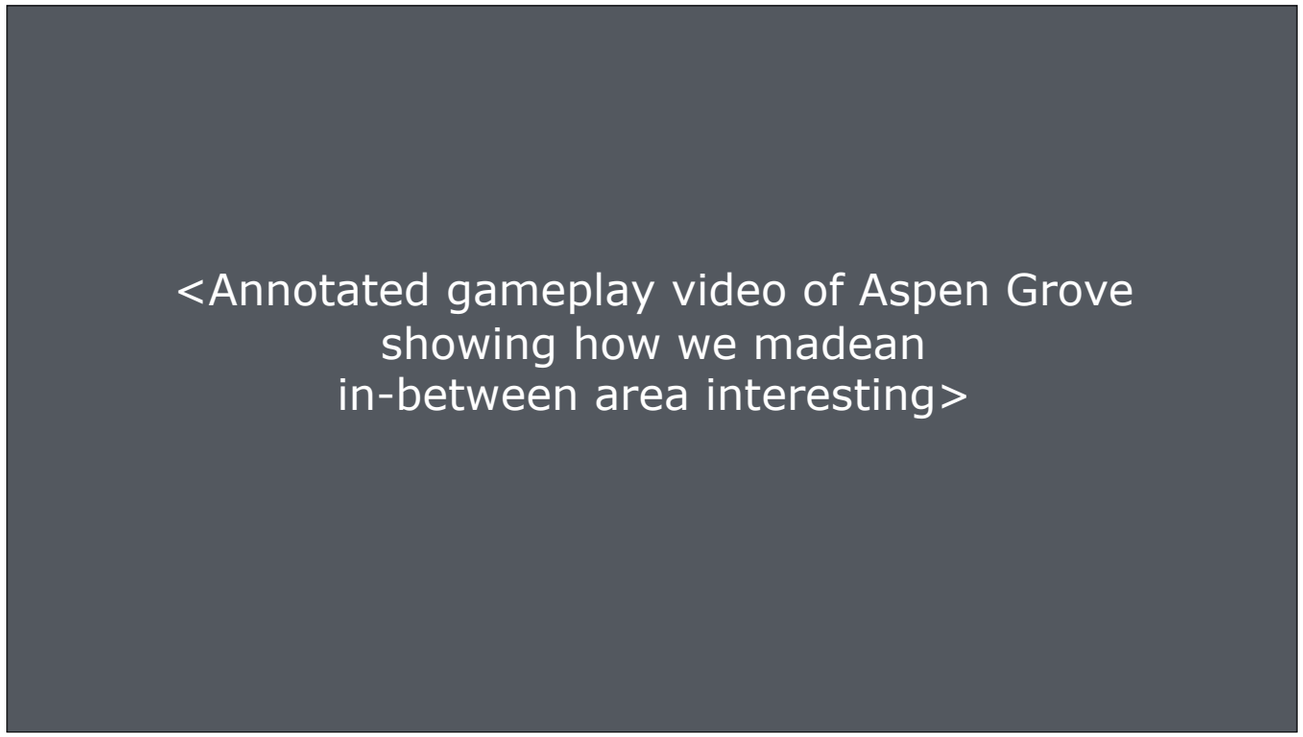

View from under overhang -
long shadows,
glimpse of
tent.

Early art of Teen Camp

Final art of Teen Camp

Once you have your main “cool story landmarks” marked out, then it became an exercise of how to make the in-between areas interesting.

- We could sprinkle more visual interest by introducing new things.
- arrange your level design to provide little discovery moments
- or add objects that you could have a dialogue about.



<video showing why Teen camp ended up being a very interesting in-between area even though it was physically very very large>

HOT TIP #5

It is normal to feel like everything is #%$#

From K.C. Green’s Gunshow comic #648

It always feels like nothing works until that one magical day when it suddenly does.

Part of the illusion of this type of GDC talk is that, it makes us sound like we really knew what we were doing the whole time? The reality was everything always felt like it’s on fire.

Don’t be too discouraged if you feel this way. It always feel like nothing works until magically one day, it just does.

Thank you all for coming! This is my contact info if you want to ask me anything.