Transitioning from Linear to Open World Design with Sunset Overdrive

(Presented at GDC 2015 / 60 minutes. These are my notes, but are not exact and only a guideline.)

INTRODUCTION

Hello, I am Liz England, and I am a designer at Insomniac Games. I'm here to talk to you about our – or, rather, my - experience transitioning from linear to open world design on Sunset Overdrive.

Now, to give you a little bit of perspective, Insomniac Games just celebrated its **20th year anniversary**. Now, I have not been at the company that entire time, of course, but almost all the games that Insomniac has made in its 20 years can be described as linear, hand-scripted experiences. We have some pretty heavy traditions, and some really good processes for making good, quality, linear gameplay.

Open world, though, is a really different beast. So I'm going to be talking about how we adapted and changed to meet many of the new challenges of open world design and development. I'm going to cover in detail:

- What do we mean by **"linear" and "open world"?**
- How did the **roles and responsibilities of designers** change from linear to open world?
- How did our **workflow** change in regards to **implementing spaces and systems** in a linear game versus an open world game?
- What were **the side effects** – the pros and cons - both of open world development and of the changes we made at Insomniac to our processes?

And with each of these questions, I'll be using examples from Resistance 3 as a point of **reference** on linear gameplay, and – obviously – Sunset Overdrive as my point of reference for open world.

Now, one **disclaimer** - Sunset Overdrive is a really unique game due to its emphasis on a traversal system. Much like GTA and Skyrim are very different open worlds, what works for one may not work for the other. I can only really speak to my experience with Resistance 3 and Sunset Overdrive, and the choices we made at Insomniac to adapt our workflow, but it's really up to you to evaluate if these are changes that would work

for your needs. Regardless, I think everyone will gain some insight into both open world but also into our development process.

OPEN WORLD VS. LINEAR

So with that I am going to open up this talk with a high-level design question: what do we mean by a linear game, and what do we mean by an open world game? I'm going to give you how I like to approach these different structures.

When we **illustrate linear gameplay**, it looks like this – a series of nodes where A goes to B goes to C goes to D, but to get from A to C you need to go through B, you can't skip it. This is a deterministic system. There is an order to the game that remains the same, from a high level point of view.

We can **compare that to open world design**, where our nodes look more like a spider web, giving player full movement through these nodes in a non-linear, non-deterministic fashion. You can go from A to C or A to B or A to B to A to D and so on, so forth.

But look closer. In a linear game, each of these nodes represents a space in the world, but also the events that take place in that space. Linear games are not just a corridor, they are a timeline. **Space and events move in lockstep** with each other. WHERE the player is in a linear game is the most useful piece of information you can have about the game.

But in open world games, each node in this diagram represents a space, but does not represent events. Events are still linear, because time (for our purposes) is linear, but they are non-deterministic: events can be shuffled, and they are player-driven. **Events are also not tied to space** like they are in linear games: these don't march in lockstep anymore. **WHERE the player is matters MUCH LESS than WHAT the player is DOING.**

For another way of thinking about this, **think of reading** instead. In a linear structure like a novel, what page the reader on tells you everything you need to know about where they are in the story. But an open world structure is more like a wiki – what page the reader is on doesn't really tell you about what came before or what comes after.

I'll give you an example to illustrate this: take a screenshot of a game and ask yourself, **"Based on where the player is, what do we know about the game state?"**

For our linear game, we'll obviously use a screenshot from **Resistance 3**. Here the player is in New York. From that I know they are in the third act of the story and what story beats they have completed, I know what levels they've done and which are still to come, I know they are 6-8 hours into the game with another 2 or so left. I know they have all of their weapons and have probably leveled one of them up into a fire shotgun. I know they have unlocked all types of grenades and seen all enemy types except for one. I know they recently fought a defensive battle with turrets, and soon they're going to come upon a major battle but right now there's a lull in the pacing.

And so on, so forth. By knowing where the player is, I know so much about the game state and where the player is in progression through the game.

Take our open world example as counterpoint. Here I know the player is in the Little Tokyo district in the central part of the city in **Sunset Overdrive**. Based on this location I know they've completed the introductory tutorial mission, so they are at least 30 minutes into the game. Or they are 30 hours into it. Or 300. The player could be in a multiplayer match. They might be in a mission. They could have completed all the missions. They may have met all of their ally factions or none of them. I have no idea when the last time the player fought enemies was, when they will next fight enemies, or even what enemy types they will fight.

This is what I mean by non-deterministic. This is what I mean when I say that the possibility space in a linear game is very narrow – the number of variables entering the funnel is small, so they can be accounted for. Meanwhile, the **possibility space** of an open world game is incredibly wide because the number of variable has increased drastically.

And this, obviously, requires a major shift in thinking about design.

ROLE OF DESIGNER

So I want to go over **how the role of designer changed** at Insomniac to take into account this new challenge of open world design.

Back on Resistance 3, our linear game, **all designers were generalists**. A single designer owned a physical chunk of the game (1-3 levels) and was responsible for all the gameplay within it: level design, combat, scripting, markup, and organizing with other departments for audio, fx, dialogue, and cinematics. Each designer, for the most part, could work alone and focused only on their slice of the game and where it fit into the overall macro. We had very discrete responsibilities with **clear division of labor**. Designers didn't work in each other's levels until the very end of the project to help out during bug-fixing, so our work was largely isolated.

If you look at the **linear node graph** I drew before, remember how each node represented a space and events within that space. In Resistance 3, each of these nodes would be a level, and each designer was attached to one or more nodes. If I worked on level C, the only other levels I needed to worry about were the level before and after mine to ensure transition was smooth into and out of my space and gameplay.

This designers-as-generalists worked really well in a linear game because of how easily our **responsibilities could be divvied up into those discrete chunks** and strung

together later. Since the game was deterministic – players had very similar experiences and variables were easily controlled for – we just needed to know the state of the player entering our space before beginning work.

But all of this **changed when we moved to open world**. Division of labor was not clear based on spaces, because the state of the player was unknown. The player could be at any point in the timeline of the game when they entered a space. The **possibility space**, like I mentioned, was huge.

We dealt with this by changing our design responsibilities, moving away from **generalists and embracing specializations**. Ownership over physical spaces gave way to ownership over systems – in a way, we all turned into systems designers who also implemented all content and markup related to our systems.

This is how we moved **from vertical design to horizontal design**. On earlier linear games, we had a vertical approach to design: a single designer would do all the gameplay in a single space. With open world our approach was horizontal - a single designer does one piece of gameplay (such as combat markup and tuning) in every space, taking into account how progression changes the state of that system over time. Spaces no longer had a single owner, but several, each responsible for a different gameplay element.

To determine these new specializations, we focused on the player's purpose within a space – what actions were they taking? What was the experience a player had in a space?

- Moving through the space? Traversal System
- Fighting enemies? Combat System.
- Completing objective? Mission & Quest System.
- Searching for loot or collectibles? Open World System.
- Engaging in Chaos Squad challenges? Multiplayer System.

Each of these systems – combat, traversal, mission, quests, multiplayer, open world – were a layer of gameplay that informed a geometric space in our city. So while we might optimize a space for one of these goals, each space had to afford for ANY of these goals or player actions.

This change had a huge practical impact on our workflow, but first I want to go over a few specific details on how and why we decided to move toward specialists.

WHY SPECIALISTS?

So, specializations in design is not something new – many studios are organized along specialist design roles like level designer or even sub-roles like technical multiplayer level designer. Some of you might be a bit surprised that we still had generalists at all at Insomniac. So I want to go over how we made this jump.

The first thing to happen that clearly defined the need for specialists was **the rise of systems** – my example here will be the traversal system, since it was the first we identified as a really specialist role.

So, why did it need all this? Well, level design in Sunset City had to reflect very **carefully controlled metrics** to allow players to traverse through these spaces. You have to understand that our player can move at a top speed of 11 m/s - that's 24 mph, and they need to be able to change direction instantly and move both horizontally and vertically through spaces without grinding to a halt. Heights and lengths and distances all had to be obeyed.

We also needed someone to **coordinate with other departments**. Environment art props were not just props, they were gameplay objects. Any railing, umbrella, awning, balcony, dumpster, car – even air conditioning units and bushes were traversal objects. A blank wall in most games is just that: a blank wall. In Sunset, a blank wall was a path the player could wall-run on. So the art department could not just go through and build geometry or change it to make a good looking game without someone to help ensure that the traversal did not break.

We also had **new tech specific to moving to an open world**: for traversal this was a hex-based environment to allow seamless streaming and building structures and roads that conformed to these hexes. We used an external tool to generate the roads and line them with sidewalks, grindrails, cars, and telephone poles - - all of which had traversal gameplay and had to fit within the surrounding geometry.

The impact on spaces meant that everything had to fit within a hex-based environment that could be seamlessly streamed in an out instead of airlocked content in our old linear games, and now we couldn’t place large buildings at arbitrary angles or cross between hexes. We had a system that allowed artists to modularized designer art, turning a big white box into walls, doors, roofs, columns, and windows with a click of a button – but these buildings needed to be built to certain metrics. These were all technical needs related specifically to world building and traversal.

Now, the last point I want to mention is that we did still have that **tradition of hand-scripted gameplay**. It’s really hard to wrest that kind of control away from designers, and though we all knew how important it was to rely on systems to generalize

interactions, we just could not let go entirely. So as a result, many of our systems involved a great deal of markup throughout the world, trading off hand-scripting in exchange for more custom markup to give AI more information about the world and look smarter. This meant a lot of busywork for designers in each specialist role that monopolized time. This might not be ideal – probably something worth solving for future work - but it was the compromise that we made to balance open world systems with hand-scripted control.

The same restraints also bore down on the other designers and let to splintering into different specializations. Each system had metrics, had to be coordinated with other departments, had to deal with brand new tech built for an open world, and had a ton of markup that required a lot of implementation. All of this was a **full time job**.

Just like moving from linear to open world design meant thinking about spaces and events or actions in those spaces as separate things, designers also had to think about **systems as separate from spaces**.

So a quick recap:

- With linear games designers were generalists. We each had ownership over a space and all the gameplay in that space
- With open world development, we became specialists. We had ownership over a system, and implementing that system across all the spaces in a game.

With that in mind, I'm going to show you what this change had on our workflow.

WORKFLOW

First I do want to point out that one of the key words in this talk's title is "**transition**". You don't just flip a switch and go from linear to open world – there's some growing pains in the process. That said, after a lot of trial and error, we did settle on a workflow on Sunset Overdrive that – to be honest – saved us and allowed us to work rapidly and actually finish the game on time. It's that final workflow – the one that succeeded – that I'm going to highlight today and compare it to our workflow on Resistance 3.

SYSTEMS WORKFLOW

So, let's start with systems. I've alluded to some of this already but now I'll spell it out. I'll use the **combat system as our example**. I'm simplifying a little because combat is a

really major system in our games, but you can apply my example to any system in our games.

In **Resistance 3**, one designer would be in charge of combat. They would have ownership over the system, **define** combat metrics, **work with the AI programmer** and lead designer to get combat systems online. They would then **communicate how to implement** combat and best practices to the design team. Then, each **designer would go back through their level** and implement combat – they would design, script, and iterate on their combat setups and place any necessary markup. Each designer was responsible for implementing combat in their own spaces. The combat designers did not go into other levels and edit them. They simply owned the system, helped designers understand it, and dealt with problems that came up and communicated solutions back to the team.

Let's switch gears to **Sunset Overdrive**. Now, mind you, the team is bigger, but we'll use one person to represent the combat team. So one person is in charge of combat. They still have ownership over that system, still **define** metrics and **work with the AI programmer** and leads. And then, they would go through the open world and **implement combat**. That meant placing markup throughout the **ENTIRE world** for dynamically spawning enemies, taking into account now all the different points in the games timeline – for example, enemy makeup changed based on which enemies had been introduced through story missions, so the markup in a single space may have to take into account several different points in time. The combat designers did not go and tell the open world designers how to implement combat – no! Combat designers implemented combat. And remember, the combat designers no longer cared about mission objective systems or collectibles or other systems: they just focused on combat.

If you go back to those diagrams of a **possibility space** – remember how wide the open world possibility space was, how many variables could affect the game's state in a single space. Well, now we've **filtered** out what goes into that funnel, so we've limited the variables for that designer needs to take into account. We turned an unmanageable amount of content into a much more focused role.

Systems are pretty easy to imagine, but the biggest change occurred when it came to how our workflow affected building and iterating in spaces.

SPACES WORKFLOW

So let's take a look at spaces in Resistance 3, we'll take a single level to illustrate our process – and I'll be focusing on design:

- First, creative directors and leads determined the **high concept** needs of the space, and it was handed off to a single designer.
- The designer creates a **proposal** detailing out the level flow and pacing, including combat, specific environments, moment-to-moment objectives, and paper level designs if needed.
- The designer creates a **whitebox** for the level, laying out the major geometry.
- The designer iterates on the whitebox and then starts **layering in gameplay**, such as combat and objectives and puzzles and story moments.
- The designer works with the **environment artist** determining the visuals and any major architectural iterations, and with **gameplay programmers** for any special one-offs – such as a boss or set-piece.
- The designer continues to **iterate** on the level, getting the gameplay more and more refined.
- Other departments come into the level to add **layers of audio, fx, cinematics**, and so on, working with the designer for their needs.
- The designer **polishes and bug-fixes** all gameplay content in the level.
- We **ship**!

So you can see that the level and gameplay never really changes hands within the design team. Other departments come in and add their own layers, but environment artists are the only ones really strongly paired with designers on their levels. Each level had a single central owner from beginning to end. We sometimes **changed ownership** and passed a level to a different designer, but these were never temporary decisions or only for a period of time: when you owned a level, **you OWNED it**.

Now let's compare that to **Sunset Overdrive**. Instead of a level, we'll take an 'area', a different chunk of physical space. In Sunset there were 4 areas, so this is a really large space compared to a level.

- First, creative directors and leads determined the **high concept** needs of the space, and then this was handed off to the world builder slash traversal designer.
- The world builder created a **whitebox** with giant, blocky shapes and basic roads.
- World builder **iterated on whitebox**, with smaller, more refined shapes that are recognizable buildings and final road placement. This included marking main

critical paths – a spider web that linked spaces throughout the area that needed to have key traversal and sightlines.

- World builder added **a layer of basic traversal geometry** – railing, cars, telephone poles, and awnings. This allowed players to get through the space and approximate traversal, but was not final at all.
- We then **divided spaces into blocks** based on their primary function in the game. When I refer to "blocks" I mean the interior of city blocks: parking lots, apartment building compounds, parks, courtyards, malls. This “primary function” was the main thing that space was optimized for, but not the ONLY purpose the space served.
- Then, we **handed off the spaces to specialists**.
    - **Combat spaces** were passed to a combat designer to optimize its geometry and traversal for a combat arena.
    - **Mission spaces** were given to the mission designers to optimize it for the needs of their mission landmarks.
    - **Multiplayer spaces** went to a multiplayer designer to optimize it for an 8-player challenge.
    - **Exploratory spaces** were given to the environment artists.
    - **Roads, bridges, and other connective tissue** throughout the world remained with the world builders, who doubled as traversal designers.
    - We had a few more categories but these were our main ones.
- The goal for these designers was to **finalize the geometry** from a gameplay standpoint – again, optimizing it for their system, and using the guidelines – the template – that the traversal designer set down for both world metrics and traversal connections. They still had to obey those main critical path flows through the space that connected it to other spaces.
- Once these teams finished the spaces, they were **handed off back and forth** between environment artists, traversal designers, and back to this secondary owner to iterate on its geometry.
- Meanwhile, now that geometry was finalized (even if it needed to be iterated on), the space was opened up to a second wave of designers to add **layers of gameplay** – this is where the systems come in that I outlined before.

    - **Mission, quest, challenge, and multiplayer designers** would build objectives and gameplay onto spaces.
    - **Combat designers** added their markup and set up dynamic enemy skirmishes.
    - **Open world designers** added loot, collectibles, spawnpoints, or even flavor elements like pigeons and blimps.
    - These layers are **similar to how audio, fx, cinematics** and other departments used to go into a level to add layers, except now the layers also consisted of gameplay.
    - These layers could be **added simultaneously** since they were more like instances of gameplay, and were invisible unless active.
    - Also, while Mission designers had to get the most approval and feedback on which spaces they used, most of the rest of the team was really free to **pick whichever spaces** they felt worked best for their needs. Design wasn't really prescriptive other than balancing the density of gameplay to make sure we had coverage across and entire area – remember, spaces were all built optimized for a certain experience, like combat, but they were appropriate for all kinds of gameplay.
- Everyone continued **iterating on their systems** within a space.
- The traversal designers and environment artists took control over **polishing and bug-fixing geometry**.

That's a really huge change between linear and open world. There's so many more steps with our open world workflow – more hand-offs, more interdependencies, more content. It's actually really tough to untangle that web because of how much a single designer's work is intertwined with everyone else, so much more than we ever had to deal with in our linear games.

SIDE EFFECTS OF SPECIALISTS/WORKFLOW

So I've covered three main topics so far:

- What do we mean by **"linear" and "open world"?**
- How did the **roles and responsibilities of designers** change from linear to open world?

- How did our **workflow** change in regards to **implementing spaces and systems** in a linear game versus an open world game?

With our remaining time I want to cover one more question:

- What were **the side effects** – the pros and cons - both of open world development and of the changes we made at Insomniac to our processes?

SIDE EFFECTS

- **-** With so many people working on the game and being able to add content simultaneously, **lead/creative feedback could just never catch up to the speed at which we created content**. Designers could go long stretches of time without any oversight, so they had to rely on themselves and on peer feedback to do the bulk of their iteration work. That's not ideal, obviously, but we had the same number of leads to review about 3-4 times as much content.

- **+** On the flip side we were **creating a lot of content really rapidly** thanks to this new workflow and specialization.

- **+** Another positive was that it **sped up the ramp up time** for new team members who joined the project. Once production kicked off, the size of our team grew, and to bring everyone up to speed we quickly identified designers by the systems they were responsible for – and new designers fit into their slots, and they could get up to speed on, say, the combat system without worry about, for example, how collectibles worked.

- **-** Of course, we ran into **bottlenecks**. Back on Resistance 3 once production started we could hit the ground running because there were no external dependencies – we could whitebox and prototype gameplay ourselves. Now with an open world game, most of the design team had to wait for the world to be built before they could add layers of content.

- **-** During that initial production time when we waited for the spaces, we created **prototype levels**. I list this as a negative, because the result was that we kept designing linear spaces, spaces with very specific, singular purposes, and spaces that weren't taking into account an interconnected open world. Some of this work was useful, but we would have been better off prototyping directly in the open world rather than in isolated levels to force our thinking toward open world.

- **-** Our layered workflow where basically mission or quest content could be layered on top of the open world and worked on in isolation from each other led to a lot of **invisible conflicts**. Ally faction bases are a great illustration of this – it's where

most of the main missions and many quests start. You would go into one and find five copies of the same NPC because they are being used in five different pieces of content, and no one is playing through the game with “natural” progression. There were a lot of narrative conflicts like this, where an NPC would be calling you on the phone for a quest while standing next to you in a mission.

- **-** We also had a problem with **orphaned systems**. Since a system owner was responsible for design and implementation of a system, if a system didn’t have an owner it was, uh, easier for people to ignore it and assume it was someone else’s responsibility. A really awful example of this was our NPC system – about two months before we shipped we realized no one owned it, and everything in the game had really been hobbled together without really following any set of rules.

- **-** Related to that – we also had a bigger problem with what I call **hot-potato bugs**. It used to be that if I got a bug for my level, I was responsible for fixing it. Now if you got a bug, due to how interconnected all our work was, it was really easy to justify “Not me!” and pass it on to someone else.

To be fair, I’ve stacked this list with a lot of negatives because I wanted to save the major positive side effect for the last point in this talk, so I can end it on a really positive note.

- + **Communication EXPLODED within design** because we were handing off our content to each other, and constantly going into each other’s work in order to deal with specifics for your own system. The design department has always been pretty good at communicating, but now we had to share our work in ways we never did before, and we got a lot better about trusting each other.

- + **Communication with art** improved drastically, since moving something as simple as an umbrella could have an impact on traversal, and the needs of a giant city meant we had to be more careful about one-off art requests. Little details needed to be approved. It sounds like bureaucracy but this was really just walking up to a desk or sending an email with a screenshot with ‘Can I?” just so that everyone was in the loop. Changes were fine and quick, but we NEEDED to have everyone in the loop.

- + And finally, communication with gameplay programmers improved! You can see a trend here. We finally successful **paired designers with gameplay programmers**, just like back in Resistance 3 designers were paired closely with environment artists. We tried to get these two departments better at working together in the past, even to the point where we are technically the same

department, but only now – with the focus on ownership of systems – did we really succeed.

WRAP-UP

And that's it – that's the story of how Insomniac Games transitioned from linear to open world design and how that affected the way we had to think about design, how we split up responsibilities among designers, how we built spaces and content and systems for a massive open world game, and what kind of side effects – the good and bad – that came from these changes.

One question I received and even had myself when I originally pitched this talk was: are these changes permanent? Are all designers at Insomniac specialists now? Do we all share content from now on?

The answer is definitely no. After Sunset Overdrive wrapped up, some of the team moved over to our PS4 Ratchet & Clank game and immediately went back to design generalists who owned discrete levels. Because that's what worked for that type of game, and that's the structure that worked best for us. I have no doubt that if we continue with open world gameplay, we'll also use a design team that specializes and creates content in layers again. The point here is to be flexible and allow your team structure to reflect the games structure.

Before we go to questions I want to point out that I'll have my slides and talking notes up on my website at lizengland.com at some point tonight. We don't have a lot of time for questions so I'll be heading to our spill-over room ____ and able to answer more questions there.