Procedural Generation and Information Games

Michael Cook
School of Electronic Engineering and Computer Science
Queen Mary University of London
mike@possibilityspace.org

***Abstract*—Procedural generation is used to achieve a wide variety of game design goals, and has led to the creation of several game subgenres by injecting variance, surprise or unpredictability into otherwise static designs. Information games are a type of mystery game in which the player is tasked with gathering knowledge and developing an understanding of an event or system. Their reliance on player knowledge leaves them vulnerable to spoilers and hard to replay. In this paper we introduce the notion of a *generative forensics* game, a subgenre of information games that challenge the player to understand the output of a generative system, and present two examples.**

***Index Terms*—procedural generation, information games**

I. Introduction

Procedural content generation (PCG), a technique where some game content is created by an algorithmic process, has been used in game development for over four decades [1] for a variety of purposes, including assisting with repetitive or at-scale content creation processes [2], or introducing unpredictability into a game design [3]. While this has led to many misconceptions about the function of PCG, it has also enabled the creation of many genre variants in which traditionally static design elements are replaced with dynamic, unpredictable or variable content.

PCG is often cited as a way to increase replayability; however, many games with PCG lack replayability, while many static games are designed to have their content replayed repeatedly. While PCG can influence replayability, these effects are not guaranteed – PCG must be evaluated and applied properly, like any design tool, to ensure that the game's systems, content and developer workflow will benefit from it. Perhaps a better framing for the use of PCG is that it enables new kinds of game, or changes the context in which other game design elements are experienced. This can be through changing the player's relationship with the game's systems, or allowing the developers to work at a scale that would be otherwise impractical. A good example of this is *Spelunky*, a blend of platformer and roguelike [4]. In his book of the same name [3], designer Derek Yu explains that the use of PCG in *Spelunky* was intended to reduce the reliance on rote learning in platformers by making levels unpredictable.

An *information game* is a game in which the player is tasked with understanding a complex artefact – a past sequence of events, a language, a physical system – by gaining knowledge about it, drawing inferences from this knowledge, and using this to seek out new knowledge [5]. A common theme in such games is a sense of mystery – the quest to gain understanding is fundamental to the game's dramatic arc. This makes information games more fragile to player knowledge. Players may feel less inclined to play an information game twice; the game is more vulnerable to spoilers than most narrative games; and players can accidentally stumble on a solution unintentionally.

In this paper we propose a new variant subgenre of information games we call *generative forensics games*, which use simulation-based PCG to create the mystery for the player to solve. This helps ease some of the problems mentioned above, such as a vulnerability to spoilers, but more importantly it opens up interesting new design possibilities related to generative design, storytelling, and player engagement with PCG systems. Our prototypes demonstrate our early exploratory work in the area, but have already yielded interesting responses from players and a lot for us to reflect on as developers.

The remainder of this paper is organised as follows: in section II we introduce information games and explain our motivation for applying PCG to them; in section III and section IV we describe and evaluate our two prototype generative forensics games: *Nothing Beside Remains* and *Condition Unknown*. In section V we discuss general future directions for the work, and then summarise in section VI.

II. Background

A. Information Games

Information game is a term coined by developer and critic Tom Francis. In [5] he defines an information game as follows:

> *An information game is a game where the goal is to acquire information, and the way you do it is to use information you've already gained and reason about it, deduce things from it, come up with theories and use those theories to go looking for more information.*

Francis cites *Return of the Obra Dinn* [6], *Heaven's Vault* [7], *Her Story* [8] and *Outer Wilds* [9] as examples of information games. While Francis describes the label as a category rather than a genre, other critics have suggested that genre is a more appropriate classifier.

Information games are often highly nonlinear, allowing the player to explore freely and acquire information at their own pace by following leads that feel important to them. As a result, information games often distribute knowledge thinly across a large space, with a lot of redundancy (i.e. the same fact can often be ascertained through many different means).

For example, in *Outer Wilds* the player explores a solar system learning about an alien race that once inhabited it and the technology they developed. Most of the technology is referenced at multiple locations, increasing the likelihood that the player will find key plot beats as they explore.

The information that these games deal with is often rich and complex, and hard for software to model or for AI to reason about. For example, *Heaven's Vault* relies on the player's knowledge of semiotics to help them intuit the meaning of an alien language; while *Her Story* tests a player's ability to read body language, intonation, gaze and facial expressions by presenting them with video of a suspect being interviewed about a crime. This suits the nonlinear nature of information games and their emphasis on discovery. Such games often have very little to 'teach' the player in terms of mechanics, and instead rely on everyday human abilities. However, this poses new design challenges, as it is hard to model how much the player has understood or worked out.

Information games have not been widely studied and have not been the subject of PCG research to our knowledge, perhaps because the canonical examples of the genre are all relatively recent (the oldest cited by Francis, *Her Story*, was released in 2015). However, we believe they are an exciting area for game design research, and their emphasis on knowledge representation, modelling and distribution also makes it an interesting genre for artificial intelligence research. The closest related work is perhaps Data Adventures [10] which automatically scrapes Wikipedia to generate murder mysteries based on real data. While we would argue that Data Adventures does not fall directly in the category of information games, we believe that there is a wider set of game designs, of which information games are a subset, which it would be valuable to classify and taxonomize in future work.

B. Environmental Storytelling

Environmental storytelling is the act of storytelling through the design of physical spaces. Bart Stewart describes it as 'the art of arranging... objects available in a game world so that they suggest a story to the player' [11]. Although still common in games, its overuse has led to it becoming stereotyped in many game design circles. In particular, the use of skeletons in various contexts has become a running joke. As game designer Ben Esposito put it: 'in game design, "environmental storytelling" is the art of placing skulls near a toilet' [12].

Some information games do not have the player inhabit a physical space, but games which do tend to leverage environmental storytelling in order to distribute information. *Return of the Obra Dinn*'s death scenes are a form of environmental storytelling, where the arrangement of people and objects imply a story, often with hidden details that need to be uncovered by careful inspection and deduction. The use of environmental storytelling techniques is prominent in information games because it provides a way to reward observation, perception and exploration, all of which are closely related to the games' themes of knowledge acquisition and use.

C. Motivation

Procedural content generation (PCG) and related terms like 'generative software' are broad terms used to describe a class of algorithms that encode a process for producing something. Often these algorithms are designed to have a large space of outputs, but this is not always the case. Some procedural systems exist to automate repetitive processes at scale [13], or to produce a single large and complex artefact from a comparatively small amount of code [14].

When applied to games, procedural generation is often used as a source of variation or unpredictability. This is used in many different ways, including achieving goals in visual art (e.g. to create convincingly natural spaces [2]) as well as gameplay effects (e.g. Yu's desire to avoid rote learning of platformer levels [3]). Marketing campaigns for games that use procedural generation often leverage the scale of their generators as an indicator of value or quality (e.g. declaring Borderlands 3 contained over 1 billion guns).

Our initial motivation for this work was to design games which eschewed classical PCG aesthetics [15]. We were interested in particular in two common trends in PCG: the attitude of treating generated content as throwaway, and the idea that players learning how generators work is undesirable (as it makes them predictable, undermining their role as a source of surprise). In the first case, we were interested in what we call *small-batch PCG* in which the player becomes invested in a single generated output and is encouraged to explore and understand it. In the latter case, we were interested in the idea of designing a game in which the player is encouraged to learn the PCG system's patterns and behaviour.

As we began planning our prototypes, the connection to information games became clearer. The idea of encouraging the player to engage deeply with a system fits well with the format of an information game, in which the player's progress is linked to their understanding of an artefact of some kind. This both encourages engagement with a single piece of content (for one playthrough) but also encourages the act of understanding the generative system, by making this part of the process of understanding and solving a mystery.

In much the same way that PCG was used to change player experience in traditionally static genres like platformers [4], shooters [16] and survival games [17], we intend to investigate how PCG could change information game design, both in terms of repeated playthroughs by the same player, and variation of playthroughs across a community of players.

III. Nothing Beside Remains

Nothing Beside Remains is played as a top-down, grid-based game in the style of a roguelike, with Unicode characters representing everything in the world, in the style of ASCII roguelikes. We used a two-phase generation system to first simulate a small village using an abstract model, and then render it in the game world for the player to explore. The village is simulated until it collapses, and the rendering process builds a ruined version of the village that is adjusted to convey a sense of how the village came to meet its demise. In this section we will describe how the world is generated, and how gameplay is structured. It can be played online in a browser[1].

The following text is shown to the player before play begins:

> *You've arrived in a ruined village lost in the desert for hundreds of years. Explore the village, see what remains of it, and see what you can learn about the people who lived here. What did they believe? What caused their downfall? You may never know for sure, but perhaps there's a clue or two left in the sand...*

NBR's game structure is open-ended, with no evaluation or in-game goal for the player. Instead, we invite the player to explore as much of the world as they want, gather information, and quit whenever they feel like they have seen enough, in a similar approach to *Her Story*. While we could have simply asked the player what they thought was the cause of the village's collapse, we felt that this would undermine the broader understanding the player was developing of the village's history in general, not just how it ended. Judging by feedback from players, discussed later, we believe this decision was justified.

A. World Simulation

There are two phases to village generation: an abstract simulation phase, and a rendering phase. This structure is inspired by games like Dwarf Fortress which simulate their world in the abstract and only render things explicitly that the player encounters (e.g. events from history are not modelled in detail). In the simulation phase we build a high-level model of the village and simulate it until we meet one of several ending conditions. There are two key components to the model: the *ecosystem*, which models the health of various aspects of the natural world, and *society*, which models the village itself.

The society model is the simpler of the two: it records the number of people living in the village, how many are of working age, how much food is in storage, and then some cultural features such as what materials they crafted with, what numbers they held to be special, what flowers they cultivated, and (after the simulation) records the reason why the village collapsed. The ecosystem model is slightly more complicated. It tracks three key features: the average temperature, the density of hostile fauna in the area, and the health of the ecosystem (i.e. how well it can support agriculture). Each of these features has a current value, a minimum and maximum cap, and a cap on how much the current value can change per simulation tick (to avoid extreme shifts).

Once these two models are initialised, we run a turn-based simulation of the village, with each turn stepping through the same sequence of checks. Temperature and hostile fauna fluctuate randomly. The ecosystem then has a chance to take damage, with the chance increasing with temperature. If the ecosystem health reaches zero, the simulation ends. The population goes down based on the intensity of hostile fauna. If the population reaches zero, the simulation ends. The population increases slightly based on the crop surplus. Crops then grow or shrink based on temperature and population size. If the crops stock reaches zero, the simulation ends.

This simulation has three possible endings: ecosystem collapse, overrun by predators, and famine. The parameters of the simulation, such as the starting values for each variable and their caps/drift rates, were set experimentally in order to achieve an approximately equal distribution of the three endings, which we verified by running 500 simulations and recording the balance of outcomes, then tweaking parameters to affect ending probabilities. The remaining parts of the model, the cultural features of the society like favoured crafting materials, are set randomly as they only impact the final rendering the village.

B. World Rendering

Once the simulation has ended, we construct the game space for the player to explore. We use a 100x100 grid of tiles, which we split into 10x10 regions which we use to assign larger features to. Then we add several features which are always present in every generated village. The first are water sources – we add a number of lakes to the map, the quantity and size of which are based on final ecosystem health. Large lakes or swamps suggest high ecosystem health at the end of the simulation, while smaller and fewer lakes might imply the village collapsed because the ecosystem collapsed (or was simply close to collapse when another fate befell it).

Next, we add two features which are common to every village: the place of worship and the statue. The statue has collapsed into fragments. The plaque at its base, which the player starts in front of, has an inscription describing the ruler of the region. We procedurally generate the inscription text (see below) and the scattering of statue pieces.

The place of worship is situated near the statue, and is a long building taking up a 20x10 space. It is laid out somewhat like a Christian chapel might be – there's a large area with rows of pews, and then an open space at the end with an altar.

After the fixed points have been added, we lay out a path from the worship hall to the nearest water source, and randomly add houses to the edge of the path. This forms the 'inner' village. Once this is complete, we add secondary roads branching off this main road, and keep adding houses to the roads – any spot adjacent to a road that does not contain a building has a chance to branch off a new road. Each house added onto these secondary roads has a chance to be a barn or a field instead, with the chance increasing as the road gets further from the center of town. This means that farmland tends to be kept on the outskirts, with more houses closer to the place of worship. We continue this process until a random number of roads have been added, or there are no more blocks to add roads onto.

C. Decorating Rooms

The outer structure of every building (including fields and places of worship) are laid out next. We first lay the structure out as it would have been originally built, and then randomly destroy or displace parts of the structure to simulate damage

[1]cutgarnetgames.itch.io/nothing-beside-remains

and decay. The chance of damage is related to temperature levels, to simulate harsh weather. We also adjust visual representation (e.g. walls become broken into small rocks). This not only conveys one of the simulation factors visually, but also helps convey a sense of ruin, as the player can move through broken walls and fences to traverse the village.

After laying down the outer structure, each building is populated with items based on its type and environmental factors. Each building has a set of items it pulls from with a specific probability. Houses contain items such as tables, chairs, cutlery, children's toys; fields contain various crops and wild plants; both barns and fields might contain animal skeletons. The chance of some items appearing are affected by the simulation - children's toys are influenced by the birth rate and general population growth; if there are very few children's toys this might suggest the village ended with the population being wiped out. Crops, and plants in general, will be less common in fields if crops failed or the ecosystem collapsed, while weeds will be more common. Throughout all buildings, predator skeletons will be more common based on the hostile fauna density. They have slightly different appearances to cattle skeletons in-game, and different descriptions too.

D. Generating Descriptions

Any object in the game can be examined to read a description. Some are static, while others have generative elements. There are two layers to our generative text system. The first is a Tracery layer which uses standard syntax for generating text [18]. Within these grammars are markers for a second layer of dynamic text, which is handled by our system after Tracery has finished generating. These markers refer to elements which are dynamic, but consistent across all text within a particular instance of the game.

For example, generated descriptions which want to mention the material something is made out of will leave a dynamic marker '@MATERIAL@' in the Tracery output, which the system will later replace with the material the village used most for crafting. This two-phase text generation allows us to mix dynamic elements into the procedural descriptions. Most of these dynamic elements are decorative, but one element in particular is very important: '@DREAMENGRAVING@', which describes the village's hopes and fears, and is specifically linked to how the village ended in ruin. This is found in the descriptions of engravings in the worship hall. One possible engraving is of a lush forest scene. This means the village in this game suffered from an ecosystem collapse. This is the only piece of information in the game that unambiguously confirms the fate of the village, although the player does not know this.

E. Evaluation

1) Player Feedback: By far the most satisfying outcome from this prototype was the feedback from players. In particular, several players wrote responses to the game, some styled as a kind of archaeological report on what they had found. These included descriptions of the village they had explored, as well as theories about the inhabitants. One player wrote: "The evidence isn't conclusive, but it appears that this village lost its ability to grow crops through a change in the environment, with water drying up and plants dying off."

Of course, it is hard for us to know whether this player's assessment was accurate, or if they simply misinterpreted the amount of water remaining. This is one of the motivations for advising players to only play the game once, because it stops them from being able to resample the generator and learn what the limits are on the generator's variation. Instead of seeing many villages and learning what the biggest and smallest lakes are, the player is instead forced to relate features such as the size of a lake to real-world analogues, as they have nothing else to compare it to. We believed this would encourage players to engage more deeply with the world, despite it being procedurally generated.

A second side effect of asking the players to only play once is that they are unable to gauge which parts of the world are simulated and which are static. For example, the same player remarked that "religion was likely a daily part of their lives, as several houses had altars with offerings of perfume". However, repeated playthroughs of the game would reveal that perfume always occurs inside houses with the same probability. Interestingly, some features that were dynamic between playthroughs can appear less meaningful when the game is only experienced once. Each society has a sacred number, and one way this manifests is in the design of chairs, which always have a sacred number of legs. One player reported: "One quirk that I found particularly amusing was the number of three legged seats, I think just about every stool or chair I encountered was missing exactly one leg". Three legs did not seem significant, and in fact seemed strange or arbitrary. Another player encountered a different sacred number, which was more of an outlier, and interpreted it more accurately: "Chairs typically had seven legs, which almost certainly means that the number seven had cultural, probably religious, significance for the villagers." Of course, the nature of history and archaeology means theories cannot always be proven 'right' and thus the idea that these chairs were simply missing a leg is a valid hypothesis. What is interesting here is that because only one village was explored by each player, a dynamic feature was interpreted very differently by each player, leading to completely different interpretations.

2) Abstract Simulation: Abstract or high-level simulations scale better than detailed simulations, and can be simulated faster. Even with abstracted simulation, generating a world in Dwarf Fortress can take many minutes. Even though our prototype is small by comparison, the approach will scale well to larger simulations. Abstraction makes sense in many generative contexts because the player is mostly concerned with the end result of the simulation. Although the history of a generated world in Dwarf Fortress can have relevance and interest to the player, most of the fine details do not impact the player's objectives (for example, the path through a village a farmer took eight hundred years ago to fetch water).

Although abstract simulations have their benefits, we feel that for information games they have a significant drawback, namely that the rendering has too indirect a relationship to the simulation. A simulation in NBR effectively defines a *space* of renderings, which is sampled by the generator in the rendering step. The problem is that the process of generating a rendering makes decisions about the world that do not have a basis or justification in the simulation. For example, the number of farms, their position relative to the water sources, and the quantity of houses have no relationship to the population of the village or its food stocks. However, as discussed in previous sections, information games encourage the player to interpret the environment they explore, which means players end up drawing conclusions that the simulation cannot back up.

For an aesthetic or tonal piece, this is necessarily a disadvantage, and in fact seems to have helped some players get a lot more out of the game. One player commented on the presence of a trident in the statue, wondering if it related to a past as a fishing village, while another hypothesised that the area was underwater at one point. These colourful interpretations of extremely small details are encouraging, and as an open-ended exploration game with a creative writing twist we find it to be quite effective. However, for information games that attempt to validate the player's knowledge (for instance, if we had asked the player to infer the fate of the village) it might be frustrating to have so much information be disconnected from the game's model of what happened.

We believe that abstract simulations can work for generative forensics games, as long as the player is not rigorously evaluated on their understanding of the system. A game with the structure of *Elegy for a Dead World* [19], for example, where players are encouraged to complete creative writing tasks in response to exploring worlds, would be an excellent way to incorporate a personal response to a space generated in this way. However if the player was expected to build on and be motivated by their knowledge (as Francis suggests in his proposal of the term information game) then this approach leaves too many loose ends that the game cannot tie up.

IV. Condition Unknown

Condition Unknown (CU) is a second generative forensics prototype, developed one year after *Nothing Beside Remains*. After reflecting on the response to Nothing Beside Remains, we decided to explore a different approach to generative forensics, this time working with a much more detailed simulation, and setting the player a simple goal. In CU we use an agent-based simulation to model a science-fiction story about a catastrophe at a research station. The player arrives after the event, and is tasked with identifying the station crew and recording their fates, inspired by *Return of the Obra Dinn*.

The following text is shown to the player before play begins:

> *You've arrived on the outskirts of a remote research station that was excavating an unusual find, deep into the Arctic wastes of this planet. Explore the station, discover what happened, and document the fate of the crew that lived there.*

The player does not have to perform this task, and can simply explore the station to piece together the story, much like NBR. However, if they wish to they can identify causes of death and submit a final report, which ends the game. The player is then told how many of their responses were correct. In this way, CU experiments with a more objective-focused approach to information games, closer to *Return of the Obra Dinn* than *Her Story*. As with NBR, there are no combat encounters or challenges, other than the optional challenge to identify crew deaths. CU can be played online in a browser[2].

A. World Simulation

Simulations in CU vary in details, but have the same core structure and premise. A research station is attacked by an 'anomaly' during experimentation. The crew try to fight it or escape, but all of them fail and are eventually killed somehow. World generation has two phases: a construction phase where we set up the initial conditions for the simulation, and a simulation phase which begins at the point the anomaly attacks and stops when everyone in the station is dead.

1) Construction Phase: In this phase we build the basic structure of the research station, place all characters in their starting locations, and check initial conditions. First we place a random number of corridors down, all connected to each other, and then we place a random number of rooms attached to the corridors. We then assign room types: the most important room is the station entrance, which is always the southernmost room, and is the only room that starts with an exit door into the outside world. Other room types are assigned randomly. There is always exactly one of the following rooms: Mess Hall, Residences, Laboratory and Security Office. All remaining rooms become secondary laboratories.

We then assign scenery to each room. Whereas NBR had a flat chance to place scenery in any space, in CU we use grammars to ensure we place scenery in patterns that make more sense. For example, a desk will have a chair in front of it. A custom scenery placement system loops through the room, placing certain kinds of scenery down depending on room type, the available space and whether or not it will be adjacent to a wall or doorway. We ensure scenery is never placed in front of a doorway, and every pattern leaves a 1-tile gap around it to guarantee that the entire station remains walkable. This was less important in NBR because the player did not need to explore everywhere, but in CU navigation is vital both for the player to discover clues, and for crew members to have free movement during the simulation.

Between five and six characters are then generated. Their two main properties are their name – drawn from a pool of 15 hand-authored names – and their profession. There is always one Security Officer, one Logistics Officer, and the remaining staff are all Scientists. We place characters in the station based on their job: scientists start in one of the labs, security guards always start in the security station, and logistics officers always start in the mess hall. Their profession affects their description,

[2] cutgarnetgames.itch.io/condition-unknown

which helps the player identify bodies later. Finally, we add the last character, the creature attacking the lab. This is referred to in the project as 'the anomaly'. The anomaly always begins the simulation in Lab #1.

2) Simulation Phase: Once the world has been constructed, we begin simulating the events that led to the destruction of the station. This is done using a turn-based system in which the station staff, the anomaly and any dynamic objects all take turns. Dynamic objects are non-intelligent entities such as fire, which has a percentage chance to spread to nearby tiles, as well as a chance to extinguish by burning out. Dynamic objects also respond to events, such as fuel barrels exploding if shot, or walls being destroyed by explosions.

The anomaly's AI system is the next simplest. The anomaly moves towards any human it is targeting, as long as it can see it still. If it has no target and sees a new person, it will chase them instead. Finally, if someone attacks it, it will turn and target them. If the anomaly begins its turn adjacent to a human, it will kill them (they combust, the cause of death is burning). If too much time has passed since it saw a target, we simply provide the anomaly with the location of a random crew member (we discuss this later). It also has a small chance to set the tile it is on on fire, and will set fire to a larger number of nearby tiles if it is shot.

The station crew have the most complex AI. The AI uses a priority-based series of checks to respond to events based on their severity, as well as the current state the crew member is in (what they are doing, what other things are happening to them, and so on). For example, seeing a dead body is a high priority event that will trigger a response and make them stop what they are doing. However, if they are already doing something urgent (such as fleeing the station) they may not regard the event as being as important. Events can be triggered by what the crew member sees (e.g. fires, dead bodies, anomalies), hears (e.g. explosions, screams, gunshots) or knows (e.g. has been told about the anomaly, has been told someone has died).

We do not have space here to exhaustively describe the web of priorities and events, however we can provide an overview by talking about the narrative structure the crew's AI was designed to tend towards. Although the initial conditions and the simulation itself are fairly unconstrained, we designed the AI systems with two important 'points of no return' that function as barriers between different acts in the narrative. These acts are not explicitly defined in the code, but are useful abstractions for dividing the system into different phases. Broadly speaking, these acts are:

- 1. Opening: no-one is alerted, crew working as normal.
- 2. Panic: station is alerted, crew seeking shelter or fleeing.
- 3. Climax: crew formulate a plan to resolve the situation.

The transition from the first to the second act is not uniform – crew members transition at different times depending on how knowledge of the anomaly spreads through the station. Each crew member has a panic level that is increased by witnessing or being told about certain events. Seeing a fire is a cause for concern and will cause the crew member to seek shelter in another room, for example, but it does not tell them specifically about the anomaly. If their panic reaches a certain level, they will attempt to find a safe place to meet up with other crew members. There are two event sequences that commonly occur to bring the entire station to full alert: the first is a crew member witnessing the anomaly and surviving long enough to radio their sighting in to the station. The second is the crew hearing something suspicious, at which point the security officer will go to investigate, which leads to them witnessing the anomaly and alerting the station.

Once the station is alerted, crew members run to rooms far away from reported danger. There are many events that can trigger here, because movement often triggers new sightings of other dangers, including dead bodies, and they can also meet living crew members too. In addition to this, because the crew members have imperfect information they may attempt to flee to locations which are dangerous – for example, fleeing a fire and running into a room with the anomaly. We prevent the crew from leaving the station by having it set in an inhospitable environment. This means that over time safe areas become smaller, and the anomaly will eventually track down crew members if it does not encounter them naturally. The transition to the third act is fixed, and always triggers when there are only two crew members left.

During the third act, remaining crew members pick from a set of 'endgame' plans for how they intend to kill the anomaly or escape. There are two possible endgames currently in CU: in one, they raid the security office for weapons and confront the anomaly; in the second, they run outside and attempt to flee the station. Both plans are guaranteed to result in their death. If any crew member spends more than five turns outside the station they die of exposure, and the anomaly cannot be killed so confronting it will result in it attacking them. When all crew members are dead, the simulation terminates, and the anomaly is removed from the game.

3) Message Passing: Although the final state of the station does impart some information about what happened to the crew, it is not enough to identify the crew, and chronology in particular becomes difficult to put together because events may overlap (for example, fire may spread through an area twice). To provide additional information to the player, as well as to convey a sense of a real sequence of events taking place, the crew's AI doubles as a messaging system in which they radio each other with reports about events and actions. After the simulation is complete, every recorded message is assigned to a *terminal* – a special world object that appears all over the station. We place one message in each station, with the most chronologically early messages placed nearest the entrance, and then later messages being placed in terminals deeper into the station. This is a simple attempt to stagger 'exposition' so that later messages are found as the player sees more and more of the station's destruction.

Messages fulfil several roles in the design. Most obviously, they contribute to the tone or atmosphere in some way, adding small details to the narrative. We partition the information content of the messages into three types: reports, intentions and updates. Reports describe an event that a character has *witnessed*, such as finding a dead body. Intentions describe what a character *plans* to do next, such as going to a particular room to find shelter. Updates are a special kind of message where we force two characters to exchange information somewhat artificially in order to help fill in gaps in the player's knowledge later. For example, when a lot of action is occurring at one end of the station, characters in the safe end may have no reason to send messages about their whereabouts. To balance this, characters sometimes mention their location when in dialogue with other characters, usually in the form of a simple 'Where are you now?' query or a 'I'm in ¡location¿, get to us if you can!' request as part of another update.

Every message, regardless of its content, is also marked with a timestamp and the name of the person sending the message (some messages also include responses from other people, but we only explicitly provide the first person's name). The timestamp is calculated as an offset in minutes from a random start time, based on the number of turns that have elapsed. As an example, if the random start time is 10:41 am, a message sent on turn 10 will have the timestamp 10:51 am. This allows the player to put messages in context, and also allows important deductions to be made about the state of certain crew members. In this way, even a message that contains no useful content in its message still conveys important information, in that it confirms the sender of the message was alive at the given timestamp.

B. Evaluation

1) Player Feedback: NBR's presentation to the player emphasised exploration and discovery, even framing the player as a historian or archaeologist of sorts. There was no explicit objective, and we instructed the player to explore at their own pace and quit when satisfied. With CU we gave similar instructions, but also provided an optional task. In addition to this, the game is built around a more specific event in a shorter time period, with less ambiguity and more first-hand information to read. As a result, we found that response from players was less creatively interpretive than before, and more focused on the task. Many players reported their success rate on the final task to us, but none retold the story of their playthrough, or attempted to retell the events that took place in their station. In both cases our playerbase is small – at the time of writing, CU has had 500 players in three months, while NBR has had 1,000 players in a year – but we suspect the more task-focused nature of CU is an important factor in the changed response.

General feedback to CU was positive. Although we recommended to players that they did not replay the game, many reported that they enjoyed doing so, and felt that it was more enjoyable to replay compared to NBR because the purpose of the game *was* to understand the generator, and effectively become more skilled at investigating and solving mysteries. However, another player noted: "on the second playthrough... having figured out the clockwork of [the generator], the task seem[ed] much more mathematical". This matches our expectation that players rapidly adapt and understand what the generator is responsible for, but it seems players differ in how much this bothers them.

2) Robustness of Deep Simulation: Although one might assume that finer-grained simulation would lead to less flexibility, we found that due to the specific design of CU our simulation was very robust to certain kinds of failure. When developing agent AI for characters the player can see, it's important to ensure the AI does not do anything that would break the illusion of being a real thinking, feeling human being. However, in generative forensics games the player only sees the end result of the simulation, and all other information they glean is information we have chosen to release. This allows us to develop AI agents which are more loosely specified and that make mistakes, as long as those mistakes don't affect the integrity of the information left behind.

For example, the exact route a character takes from one room to another in CU is not seen by the player. We use basic A* to path between locations, which means characters often take more dangerous routes because they are more optimal in terms of distance travelled. Although we forbid A* to path through tiles which are on fire, we don't discount tiles *adjacent* to fire, which are dangerous as they may catch fire when the character is on them. This leads to characters dying unnecessarily in fires. If the player were able to see the precise route they took, this would make the characters seem less intelligent, less human and less engaging. However, in CU the player only learns about this event through what we choose to report: the character might report they are leaving to move through the facility; another character may report seeing their body; and the player can discover the character's final resting place. At no point is the player exposed to the problems with their pathing algorithm.

This can be seen as an invitation to write worse AI, but in fact it should really be seen as an opportunity to write *better* AI. This design structure allows us to vary our focus and level of detail in our agents, meaning we can put extra effort into writing AI to handle social situations or knowledge representation, and worry less about the exact strategy an agent uses when fighting a monster or pathing through a burning building (since the player will only learn about the event's outcome). This gives us a chance to focus on different things, and gloss over the rest with good writing and presentation. This is similar to the idea of *story sifting* introduced by Ryan in [20], but rather than identifying potential story templates in a trace of events, instead we simply choose not to report events which do not contribute to the story (in the case of CU this is baked into the system, rather than being a decision an AI narrative manager must make on-the-fly).

V. Future Work

Both of our prototypes led to useful feedback and raised more questions about what generative forensics can do for game design and how these games can be best designed. Many points of future work exist, but we outline a few prominent lines of inquiry here.

1) Knowledge Guarantees: We explicitly avoid trying to reason about whether *Condition Unknown*'s core mystery is solvable at generation time, although it would be fairly straightforward to do so. There are many procedural generators which try to make firm guarantees about their output, many of which involve ensuring there is a valid solution to a problem. We decided not to do this for our prototypes: we wanted each output to not necessarily be solvable, to encourage players to consider when they have exhausted all avenues of investigation, and to reflect the fact that sometimes investigations or archaeological surveys do not yield answers.

However, this is a departure from traditional information games. Many allow the player to end the game without understanding everything, but all of them guarantee that the truth can be known if the player explores and understands enough (with the possible exception of *Her Story* which maintains an element of ambiguity for some plot points). We believe that games which do not guarantee a solution are interesting and valuable, but it is also worth exploring how we can build generative forensics games that guarantee solvability too. This might initially involve quite clumsy solutions, for example by hiding unambiguous details in every station in *Condition Unknown*. It might also be possible to write solvers or reasoning systems to play the game, but this might encourage the games to move further towards more easily-modelled knowledge, which arguably departs from the spirit of information games.

2) Alternative Evaluations: Information games usually either do not evaluate player knowledge, or do so in a fairly strict, unambiguous manner. However, we were struck by player responses to *Nothing Beside Remains*, and how performing a creative act in response to gameplay (in this case, writing a report on their findings) seemed to spur a deeper engagement with the game, as well as encouraging a different response than games normally ask of the player. A similar experience can be found in *Elegy for a Dead World* [19] in which the player completes creative writing tasks in response to exploring hand-designed planets.

Incorporating creative writing tasks as part of the explicit response to the game is an interesting design extension that embraces the unpredictable and varied nature of generative forensics games. Although the game itself cannot understand or respond to such work, we envisage a meta-level community of players who write reports on their experiences, and collectively gain a better understanding of how the generator works simply by reading the reports written by other players. This would be an interesting way to encourage community discussion about the game, and would also add the potential to explore social game design ideas as espoused by designers like Dan Cook and Tanya Short [21], [22].

VI. Conclusions

In this paper we examined 'information games', in which players solve a mystery through exploration, understanding and reasoning, and suggested a new class of game, 'generative forensics games', in which the player tries to understand the output of a generative system. We described two prototypes that explore this idea from different perspectives, *Nothing Beside Remains* and *Condition Unknown*. We motivated and described their design, discussed preliminary player feedback, and pointed to future considerations for the genre.

Procedural generation is often seen as a way to obtain 'more unpredictable stuff' [23], but as a design tool it has many different uses, and combining it with unfamiliar design structures can be enlightening and entertaining. We believe generative forensics games are a promising design space to explore, and may also point to a need for better tools to help designers work with generative systems [24]. Being able to use generative systems as a natural part of the design process, as one might with a physics system or a dialogue tree, is necessary to open up this design space to more people.

VII. Acknowledgements

Thanks to Andrew Dunn for naming Condition Unknown, and to Tom Francis for discussions about information games. This research was funded by the Royal Academy of Engineering Research Fellowship scheme.

References

[1] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Springer, 2016.

[2] SpeedTree, "Speedtree, open research content archive (orca)," 2017. [Online]. Available: http://developer.nvidia.com/orca/speedtree

[3] D. Yu, *Spelunky*. Boss Fight Books, 2016.

[4] Mossmouth Games, "Spelunky," 2008.

[5] T. Francis, "Information games," tinyurl.com/informationgames, 2019.

[6] L. Pope, "Return of the Obra Dinn," 2018.

[7] Inkle, "Heaven's Vault," 2019.

[8] S. Barlow, "Her Story," 2015.

[9] M. Digital, "Outer Wilds," 2019.

[10] M. C. Green, G. A. B. Barros, A. Liapis, and J. Togelius, "Data agent," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 2018.

[11] B. Stewart, "Environmental storytelling," tinyurl.com/stewart-es, 2015.

[12] B. Esposito, "in game design..." tinyurl.com/esposito-es, 2016.

[13] F. J. Budinsky, M. A. Finnie, J. M. Vlissides, and P. S. Yu, "Automatic code generation from design patterns," *IBM Systems Journal*, vol. 35, no. 2, 1996.

[14] A. R. Brown and A. Sorensen, "Interacting with generative music through live coding," *Contemporary Music Review*, vol. 28, no. 1, 2009.

[15] G. Smith, "The future of procedural content generation in games," in *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2014.

[16] Vlambeer, "Nuclear throne," 2013.

[17] Mojang, "Minecraft," 2009.

[18] K. Compton, B. Kybartas, and M. Mateas, "Tracery: An author-focused generative text tool," in *Interactive Storytelling: 8th International Conference on Interactive Digital Storytelling*, 2015.

[19] I. Lambe and Z. Scott, "Elegy for a dead world," 2014.

[20] J. Ryan, "Curating simulated storyworlds," Ph.D. dissertation, 2018.

[21] D. Cook, B. Fulton, J. Gonzales, Y. Bialoskursky, and M. Fitch, "Game design patterns that facilitate strangers becoming "friends"," in *Proceedings of the Eleventh Project Horseshoe*, 2016.

[22] T. Short, D. Hurd, J. Forbes, J. Diaz, A. Ordon, C. Howe, S. Eiserloh, and D. Cook, "Coziness in games: An exploration of safety, softness, and satisfied needs," in *Proceedings of the Twelfth Project Horseshoe*, 2017.

[23] M. Cook, "More unpredictable stuff," https://tinyurl.com/cook-unpredictable, 2015.

[24] M. Cook, S. Colton, J. Gow, and G. Smith, "General analytical techniques for parameter-based procedural content generators," in *2019 IEEE Conference on Games*, 2019.