Can an Unknowing Participant distinguish between Procedurally Generated and Human Designed Interiors?

Thomas O'Leary

***Abstract*—What's the problem? What am I looking at? How does that help solve the problem?**

Opening, Challenge, Action, Resolution

Attempt to see if procedural generated interiors can be percieved as human designed. Comparing the two together and see if participants prefer the procedurally generated designs.

I. Introduction

Urban open world games such as Grand Theft Auto V CITE:GTA, The Division CITE:DIVISION and Batman: Arkham Knight CITE:BATMAN have such large built-up areas for players to venture in. However, only a few handpicked buildings in these large cities are accessible and have modelled interiors leaving the others to be blocked off for decorative purposes. This could be resolved by modelling and designing each room in these cities, but this would become incredibly impractical. Other issues with this can lean towards rendering and the storage of such heavily dense areas.

Procedural Generation

Procedural Generation (PCG) refers to automatically creating content using algorithms [1]. PCG has many applications in video games, some notorious examples being the world/cave generation in Minecraft [2], the texture [3] and world generation [4] in Spore [5] and the procedural texture and music generation in .kkrieger [6].

Using PCG, this largely time-consuming task of designing room interiors can be automated. And can possibly help maintain a player's immersion within the game. An issue with this however is that PCG tool's can be seen as boring and repetitive [7].

Through my literature review though I have found many implementations and techniques of Procedural Interior Generation (PCIG), none of these get compared to Human designed interiors.

This study looks to see if a participant is able to tell the difference between Human designed and AI generated interiors.

II. Literature Review

My literature review consists of two parts that I believe to be important to my research question. It first describes different implementations of Procedural Interior Generation (PCIG) then explores ways in which Artificial Intelligence (AI) is compared to Humans.

A. Implementations of Procedural Interior Generation

Although PCG has a lot to show and offer in game development - being used for characters, terrain, and textures - the use of Procedural Interior Generation (PCIG) in games however is scarcely come by.

A game that does use PCIG is Catlateral Damage [8], a small indie game developed by Manekoware where you play as a cat on a destructive rampage in its own house. In 2017, Chris Chung (the developer behind the game) wrote a case study about the level design in his game [7, Chapter 6]. When developing Catlateral Damage, Chung was undecided on how to design the levels and ultimately went for PCIG [7, Chapter 6]. Before the interior decoration can take place, a Squarified Treemap algorithm is used to generate the room layouts and floor plans within the level [9]. Each room generated from this algorithm has an associated data file, this file contains data such as furniture available and maximum type of each furniture. The furniture objects that can be placed, have physics components attached to allow them to be accurately placed within the level - for this, a Rectangle Packing algorithm [10] is used to place these objects within allocated surface areas on the floor and on top other furniture objects. Concluding the case study, Chung states that most players could not notice that the levels were procedurally generated - although this is a promising statement, Chung has not shown any evidence to back this claim.

On 29th April 2021, Sony Interactive Entertainment published patent US20210121781, titled "AI-Generated Internal Environments Based On External Geometry" [11]. The patents' description goes onto explain a Machine Learning (ML) tool that takes in data from the external structure of a virtual building and generates an interior environment just from this data. Although this is just a patent for an ML tool, this could be the start of PCIG being used in AAA Titles.

Despite there not being many implementations of PCIG in games, there are however a handful of published papers that have used their own techniques to emulate room interiors.

Multi-Agent System: In 2009, T. Germer and M.Schwarz sought out to procedurally arrange a rooms' furniture in real-time. [12]. A demonstration of this system can be found on

YouTube, uploaded by T. Germer himself [13]. This system involves a Multi-Agent based solution where each furniture object, in a given room, is seen as an individual agent that seeks a suitable parent furniture object. These agents have custom semantic descriptions to allow them to create different object layouts, an example listed by the authors is a chair - a chair can either be set next to a table/desk but can also be isolated in its own surroundings leading it to have many possible parent objects [12].
Each agent has 3 states:

1) Search
   - All agents start in this state, they begin by searching for possible parent objects - if a parent is found that suits its semantics the agents' state changes to *Arrange*, if a parent can't be found at all the agent is deleted.
2) Arrange
   - In the *Arrange* state, the agent attempts to place and orient itself with the parent accordingly. Whilst doing so, it has to check for collisions with other agents in which the Separating Axis Thereom is used [14] - if no collisions are found the agents' state changes to *Rest*.
3) Rest
   - In the *Rest* state, potential child agents are now able to seek this object as a parent. If the parent moves, the resting agent will move along with it - however if this move results in a collision, its parent is lost and the agents state is changed back to *Search*.

Before using this system, it requires a certain degree of user input [12]. Firstly, each room would need to have specific data such as labelled parts of the room (windows, floor, doors), how many objects, what type of objects and how many of these types can be used in this room. A user is also required to write the semantic description of each type of object - this includes object clearance distance, possible orientation values and potential parents (and what sides to correspond with). Due to the parents of each object being manually set by the user in their semantic descriptions, a hierarchy is not explicitly defined - yet handled at run-time by the system just before the agents are initialised [12].

Although a large proportion of the rooms furnishing is handled by the agents themselves, a big drawback is that each agent must have a manually defined semantic description. This could cause a lot of issues especially if many types of objects are to be used in a room.

Rule-Based Layout: A Rule-Based layout approach was proposed in 2009, this method would focus on user defined rules and layouts [15]. Users would be able to specify what objects can be placed within a layout - these would represent an instance of a class and contain certain rules on how it should behave when being placed.

The relationships between different objects could be explicitly and implicitly defined. A user is able to explicitly create a rule in an objects class or use features. An example of explicitly defining a relationship, as told by the authors, is by setting a rule for the sofa class to always face an instance of the TV class [15]. An implicitly defined relationship uses feature types. Feature types can be used for checking what objects should and should not overlap. These are used as tags and are applied to specific parts of objects - for example an *OffLimit* feature type tag would be used for the bounding box of most objects [15].

Rules can be defined in multiple ways too. They can be associated specifically with an object class, this would mean that this rule only applies to this class and any instances of it. Another way is by defining rules in the layout planner - this would help with objects that general rules may not be applicable to. An example listed by the authors is a chair in a meeting room, this type of object is usually sat against a wall - so this rule would be applied in a "Waiting Room" layout [15].

The layout planner is responsible for sending objects to the layout solver. As stated before, the planner can have custom rules itself to allow it to be applicable to different room layouts (living room, factory floor, waiting room). It also has a backtracking rule that is only triggered if an object of interest is not placeable. If this is the case, the planner would backtrack to place previous objects in different positions to allow this object to be placed.

The layout solver is given an object from the planner and the type of layout (dependent if a user has applied rules to the planner). Within this layout it finds all possible locations for the new object - these possible locations are initially based upon the set rules of the object and the rules already set in place for existing objects in the layout. The possible locations then take specific feature types into account, such as the amount of clearance an object requires or the *OffLimit* tag that was mentioned earlier [15]. A Minkowski Sum [16] is carried out containing these inaccessible areas and removed from the list of possible locations. With this completed, the object is then given a list of all possible locations in the layout it can be placed.

Something good. Something bad.

Constraints: In 2017 P. Henderson, K. Subr and V. Ferrari presented a data driven model that learns from the SUNCG [17] database to generate furniture layouts [18]. This database contains over 45000 apartment layouts that are designed by humans, from this 45000 - 2500 CAD models are categorised into furniture, these models are then assigned an object class of which there is 170 of (television, bed, table). Their model is presented in such a way that it can be left to be fully automated [18], but does allow flexibility with the user allowing them to change constraints within the layout.
These constraints include:

- Room size, shape & type
- Exclusion of Object classes
- Furniture clearance
- Locations of specified furniture
- Locations of doors & windows

Upon generating layouts, the results varied dependent on the set constraints applied to the room. Layouts that had no set constraints were found to be generating in 0.04 seconds on average [18]. Whereas layouts that did have constraints generated anywhere between 0.04-112 seconds on average

[18]. The large differences in generation times vary solely on what constraints are applied - room size, object exclusion and clearance constraints all generated in 0.04 seconds on average, whereas the combination of room size and the locations of doors & windows took around 112 seconds to generate.

A user study was carried out in their paper, where 1400 pairs of layouts are presented to 8 non-experts in an image format [18]. These participants are asked to identify the layout that has a more realistic/natural setting. In each pair one layout is from their model, the other being human designed - the order in which these images are shown are randomised per pair. In this study, both constrained and unconstrained layouts are put to the test. For unconstrained layouts, they were presented in 2 different styles; 1st person and an overhead view. Layouts that were presented in 1st person, were seen to be slightly preferred over the human designs and those presented overhead were seen to be indistinguishable from human designs. Constrained layouts were only presented in an overhead format, but two sets of constraints were used:

i. Fixed room size & fixed placement of a singular object
ii. Fixed room size & fixed door/window locations

With the constrained layouts of (i), the model layouts were seen to be indistinguishable from those of human design. Whereas the layouts of (ii), human designs were preferred over the models.

Something good. Something user study. Something bad.

Statistical Relationships: In 2011, a PCIG system using statistical relationships was proposed [19]. This system would take in existing relationships from previous working examples (spatial, hierarchical and pairwise relationships) and pass these relationships into a cost function. The spatial relationship represents the objects distance and orientation to its nearest wall. The hierarchical relationship, similar to other papers, represents a child/parent relationship between objects - an example listed by the authors is a candle (child) placed on a table (parent). The pairwise relationship represents the interaction between different pairs of objects (TV and a sofa), these pairs are subject to each other's distance and orientation constraints.

The cost function is used to quantify the realism/functionality of the state of the furniture arrangement. The higher the cost of an object, the higher the priority it takes. There are 5 overall stages to the cost function.

- Accessibility
  - For an object to be "functional", it must be accessible. Each object has a defined *Accessible Space* as well as *Bounding Box*. Whenever another object enters another *Accessible Space*, the cost increases. Within this function, only the objects positional values are passed as it was found to be better for optimization.
- Visibility
  - Certain objects must be viewed from a specific direction - these objects are given a *Viewing Frustum* (some objects include TV's and paintings). Similarly to the Accessibility cost function, whenever another object obstructs a *Viewing Frustum*, the frustums values are passed in and the cost increases.
- Pathways
  - Within the furniture arrangements, pathways are created using *Cubic Bézier curves*. These curves are represented as rectangles in the 3D space. The cost function is similar to that of Accessibility & Visibility, yet in this case the control points of the Bézier curve is used as the positional value and applied to the rectangles used to represent the pathway in the arrangement.
- Prior Spatial Relationships
  - The prior Spatial relationship (distance and orientation of the nearest wall) is subtracted from the objects current Spatial relationship.
- Pairwise Relationships
  - Similar to the Prior cost function, the pairwise cost function is defined to subtract the distance and orientation of the paired objects.

YouTube video [20]

B. Artificial Intelligence Compared to Humans

This is going to be a little more difficult to write about, as I haven't read a paper on this so far. And I have only managed to find 3 papers that talk about this, but I am not sure that they could be entirely relevant.

III. Research Question

From the above sources, I have formed **actual research question that's totally not wip anymore**

A. hypothesis & null hypothesis

Hypothesis stuff...

IV. Artefact

A. What will be made

AI that procedurally generates interior at runtime in a predefined room size and access to pre-made furniture assets. Will be later compared with human designed interiors (being given the same room size and assets)

B. How will I ensure Quality

Quality control. Roadmap? Unit Testing? Integration testing?

C. How will I create it

The AI will be made in the Unity game engine (Version 2020.3.12f1) **cite unity engine**

D. Why will this answer the questions

The artefact itself will not contribute towards answering the research question but will contribute towards the research that will help with answering the question. Long sentence that needs to be fixed LMAO

V. Research Methodology

A. Experimental Design

A/B test. Paper uses something similar [18].

B. Limitations

Time, resources

C. Sampling Plan

Sample size, sampling method

D. Data management plan

Managing, collecting, & storing data

E. Data Analysis

Something to do with R

F. Ethical Considerations

I plan to not commit war crimes I promise

VI. Appendix

Data analysis code, supporting screenshots, system development life-cycle, list of unit tests & testing plan

References

[1] J. Togelius, E. Kastbjerg, D. Schedl, and G. N. Yannakakis, "What is procedural content generation? mario on the borderline," in *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, ser. PCGames '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2000919.2000922

[2] Mojang, "Minecraft," [PC], 2011.

[3] D. g. DeBry, H. Goffin, C. Hecker, O. Quigley, S. Shodhan, and A. Willmott, "Player-driven procedural texturing," in *ACM SIGGRAPH 2007 Sketches*, ser. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 81–es. [Online]. Available: https://doi.org/10.1145/1278780.1278878

[4] K. Compton, J. Grieve, E. Goldman, O. Quigley, C. Stratton, E. Todd, and A. Willmott, "Creating spherical worlds," in *ACM SIGGRAPH 2007 Sketches*, ser. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 82–es. [Online]. Available: https://doi.org/10.1145/1278780.1278879

[5] Maxis, "Spore," [PC Game] Electronic Arts, Steam, 2008.

[6] Farbrausch, ".kkreiger," [PC Game] Farbrausch, 2004.

[7] T. Short and T. Adams, *Procedural Generation in Game Design*. CRC Press, 2017.

[8] C. Chung, "Catlateral Damage," [PC Game] Manekoware, 2015.

[9] F. Marson and S. Musse, "Automatic real-time generation of floor plans based on squarified treemaps algorithm," *International Journal of Computer Games Technology*, vol. 2010, 09 2010.

[10] E. Huang and R. E. Korf, "Optimal Rectangle Packing: An Absolute Placement Approach," *Journal of Artificial Intelligence Research*, vol. 46, pp. 47–87, 2013.

[11] W. Benedetto, Sony Interactive Entertainment Inc., "AI-Generated Internal Environments Based On External Geometry," Patent Number: US20210121781, April 29 2021. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=US322909393&_fid=WO2021081415

[12] T. Germer and M. Schwarz, "Procedural Arrangement of Furniture for Real-Time Walkthroughs," *Computer Graphics Forum*, vol. 28, no. 8, pp. 2068 – 2078, 2009.

[13] T. Germer, "Procedural Arrangement of Furniture for Real-Time Walkthroughs," [Online] YouTube, June 2009. [Online]. Available: https://youtu.be/xwVUknGeycQ

[14] S. Gottschalk, M. Lin, and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics*, vol. 30, 10 1997.

[15] T. Tutenel, R. Bidarra, R. Smelik, and K. J. de Kraker, "Rule-based layout solving and its application to procedural interior generation," in *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)*, 01 2009.

[16] "CGAL 5.3 - 2D Minkowski Sums." [Online]. Available: https://doc.cgal.org/latest/Minkowski_sum_2/index.html

[17] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic Scene Completion from a Single Depth Image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 190–198.

[18] P. Henderson and K. Subr and V. Ferrari, "Automatic Generation of Constrained Furniture Layouts," *Computer Vision and Pattern Recognition*, 2017.

[19] L.-F. Yu, S.-K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher, "Make it home: Automatic optimization of furniture arrangement," in *ACM SIGGRAPH 2011 Papers*, ser. SIGGRAPH '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/1964921.1964981

[20] HKUST VGD, "SIGGRAPH 2011 - Make it Home: Automatic Optimization of Furniture Arrangement," [Online] YouTube, April 2011. [Online]. Available: hhttps://youtu.be/vlDoSv6uDKQ