Designing Procedural Game Spaces: A Case Study

Michael Nitsche
michael.nitsche@lcc.gatech.edu

Calvin Ashmore
gtg126z@mail.gatech.edu

Will Hankinson
simian@gatech.edu

Robert Fitzpatrick
robfitz@gatech.edu

John Kelly
gtg088p@mail.gatech.edu

Kurt Margenau
gtg264q@mail.gatech.edu
,

School of Literature,
Communication & Culture
Georgia Institute of Technology
Atlanta, GA 30032-0165

ABSTRACT
Procedural content generation holds many promises for the design, art, and production of video games. It also poses a number of challenges. This paper concentrates on the procedural generation of game spaces. We specifically argue for a connection of a player's agency with the procedural world generation. First, space generation in games is broken down into four main approaches: designer-created, random, player-created, and procedural spaces. Then, the paper introduces our experimental game prototype *Charbitat* that merges these four stages and provides a practical case study. *Charbitat* generates game worlds based on the gaming style of its players, who create the world as they play it. We describe how the project met the challenges in design and implementation. Finally, we point out new questions opened up by the project and relevant for procedural content generation.

Keywords
Real-time, virtual space, procedural, video game, architecture, game design

1. INTRODUCTION

Computers are procedural engines but when it comes to procedural content, video games rarely utilize their inherent powers. Procedural game content is game material – such as AI, sounds, spaces, objects – that is generated during the game's runtime. Few game use this form of content generation; instead, they rely mostly on pre-fabricated elements. Dialogue is scripted, levels and objects pre-modeled, sound pre-recorded, character behavior pre-defined. Most game research deals with these games because they dominate the market. Procedural content generation is far less discussed, although it stands out as a core feature of the computer as media. That is why this paper will look at the procedural side of game design and discuss specifically procedurally generated 3D game spaces. These worlds are not fixed but created "on the fly." Because procedural approaches shift the content generation into the algorithm and away from a pre-producing development team, they add unique qualities to the world of gaming. Their flexibility opens up new forms of gameplay and game experiences that have been recognized as an important element of game design [6]. Procedural techniques have been applied in the field of graphics [3], AI [8], character animation [12], and level generation. This paper concentrates on the procedural generation of 3D game worlds.

There are a number of video games that apply procedurally created game worlds, mostly in the form of some kind of level generator. We will argue that too few of them include the world creation as part of the player's agency. Yet, this is where such a feature would change the playing experience most and where the world generation would merge with the play. Thus, our argument is twofold: we call for more exploration in the field of procedural game content; specifically in the area of generating 3D game worlds. In addition, we argue, that the player should affect this generation; the newfound freedom should enrich the player's agency in the game.

To present our argument, we deliver an overview over different forms of game space generation. Then, we introduce *Charbitat*, a game research project conducted at the Experimental Game Lab at Georgia Tech that provides a practical example for procedural space generation. We will discuss main design decisions and technical approaches to provide our answers to the question of procedural content

generation. Finally we look at new challenges that grew out of this project.

2. DEVELOPMENT OF GAME SPACES

The role of space has been debated in games research for some time [1, 9, 16]. This paper attempts to take the debate into procedural 3D game spaces. 3D worlds have not materialized over night, but are the result of a continuous evolution. The following paragraphs will summarize some key approaches to virtual space production. The goal is not to provide a history of game spaces but to find core plateaus in the evolution of their generation. This evolution can be broken down into four main approaches: designer-created, random, player-created, and procedurally generated game spaces.

2.1 Designer-created Spaces

Designer-created spaces are entirely pre-modeled 3D game worlds generated by the game developer and delivered with the game. They are rigid and largely passive game environments. They offer little or no opportunity for change but are optimized for a certain game experience. Once mastered and fully explored, such a fixed space offers nothing new in replay. Examples include the levels of the single player campaign of *Half-Life*. Such worlds become interesting only, when the characters' behavior within them reaches significant complexity. The battle arenas of a multi-player-session of *Half-Life* remain attractive not because of the flexibility of the game world, but because of the players' spatial behavior in them. The performance not the virtual stage is the dynamic format and players have little to no affect on the world itself. The unique movement of avatars through a *Counter-Strike* level produces new spatial configurations between the player characters driven by social interaction. The value of these social spaces has to be applauded and has been investigated (e.g. [4, 14]), but is not topic of this paper, that looks at the worlds that hosts such a social gathering. In this respect, these games lack the notion of interactive space generation.

2.2 Random Level Generators

Games like *Diablo* or *Rogue* use mostly random level generators to build a space within a subset of set "level elements." Level elements are space sections, objects, and entities that are combined to form a new level. Due to the wide range of possible arrangements, a player is not likely to ever get the exact same *Rogue* level twice. This feature raises the replay value and the dynamics of the game, but there is no way for a player to predict or shape the space generation. With no impact on the level generation, the agency of the player does not extend onto the new possibilities of level creation [10]. They always remain beyond the player's interaction. Game worlds can grow into endless levels in which the variety stays limited and the player's agency remains unconnected to the space generation. Random level generators do not fulfill the second demand posted at design of procedural game spaces as they do not involve the player in the process.

2.3 Player-created Spaces

Many games have tapped into players as a powerful space generating force that can extend the life cycle and design range of a game significantly. Referring again to *Half-Life*, its player-created mod *Counter-Strike* is a good example. Developers provide players with the tools to create own game worlds or modify existing ones. The result can be an exponential growth of game environments. The player-created worlds are basically designer-created spaces as the main game engine still does not provide for more responsive environments, but now there are a lot more of them. They are a child of the re-mixing and modding culture and depend on special editor programs that provide access to the game spaces and allow modification. There is a clear differentiation between playing and content generation. Players have to work in an external editor to change the game world, recompile it, before they can play it. Play and space generation still remain separated.

The Massive Multiplayer Online world of *Second Life* offers an interesting hybrid approach, where players can shape and code their own spaces within the game world. Here, the space creation and the exploration of the game world can be combined. But the original interaction design in *Second Life* lacks many features of a game as defined in [5, 13] and is more focused on the creation and maintenance of a social space.

2.4 Procedural Spaces

The powers of procedural space generation are tempting: players can create not only their own path through a game world but also their own game universe itself. For example, *MojoWorld* is a fractal-based world generator that allows for the creation of whole planets to finest detail. Yet, these worlds lack

any active inhabitants or objects to interact with. They are visually stunning pieces of extremely low interactive range.

Other references come from architecture, where researchers have developed algorithms to generate virtual cityscapes [17] or abstract "liquid architecture" [11]. The results are illustrative structures with little or no interactive ingredients.

Very few games use procedural space generation. *Rescue on Fractalus* generated 3D flythroughs of fractal-based mountains and valleys processed in real-time. The world was inhabited by enemy forces, pilots to be rescued, and suicide flying saucers. Insofar, *Rescue on Fractalus* features clear interactive and goal-driven game settings and but it remains simplistic in its presentation form due to the original platform (Atari 800 and 5200). *Vib-Ribbon* uses sound to create vector-graphic game levels that consist of long strings of obstacles. The player can provide the music to generate the world and has at least indirect control over the level generation but lack any direct control over the outcome.

The new features pose practical challenges to technology and design. The main question for our context was: How to integrate play and world creation? There cannot be a single solution to such a challenge but only artifacts that develop the field further. *Charbitat* sees itself in that position. The following paragraphs will provide some possible answer using our practical game project *Charbitat*.

3. THE CHARBITAT PROJECT

3.1 Outline

The principle answer of *Charbitat* to the above posted question is: In order to integrate space generation and gameplay, one has to become part of the other. *Charbitat* is an experimental single-player game prototype in which the player creates the world *as s/he plays through it*. There are no points to win or records to break but the gameplay is a goal-driven exploration and generation of space. This strategy of level generation differs from the above outlined space generation methods but combines them. The space is player-created insofar as players can steer the world generation through their in-game actions; it has random elements in the shaping of the terrain and the positioning of entities and objects; it has elements of designer-created spaces as it uses pre-fabricated local objects and some pre-modeled set pieces; all of these elements are combined in the procedural space generation.

The background story of *Charbitat* tells of a young Chinese princess, who gets poisoned. She is beyond any doctor's help and falls into a coma-like state. The player controls this little girl, now trapped in a dreamlike state of internal chaos and conflict. The poison has destroyed her body's and mind's natural balance of the five elements of Taoism. This state between death and life is the game world of *Charbitat* and like a real coma, it can be infinite. It is the goal of the game to overcome the obstacles, master one's own world, balance the elements once again, and finally leave this borderless dreamscape.

Figure 1. view of one game world

3.2 Technical Implementation

Charbitat uses a full modification of *Unreal Tournament 2004* and a Java program to generate the game territory. It changed the *Unreal* system to modify AI, generate the space, implement a new interface, and trace player behavior. The Java program makes up the backend, handling the form, ingredients, and consistency of the world.

To provide a scalable and continuous game world, the environment is split into individual tiles, each of which is about 500 virtual meters across. Instead of one central seed value, *Charbitat* uses seed values for each single tile. This way, it applies countless individual seed values that spawn smaller locations, which add up to one seamless 3D world.

Based on these values, the Java backend generates individual heightmaps for each section. The underlying terrain is determined using noise functions and filters, which always include an element of randomization. The local terrain may

then be mixed with other filters determined by global landscape features, such as rivers, lakes, cliffs, and coastlines. These features are selected based on what is logically permissible for that tile and consistent with the surrounding world. For example, if there is coastline on the bordering edge of a previous tile, then the coastline must continue into the adjacent tile as well. In that way, we form the space through continuous and legible features.

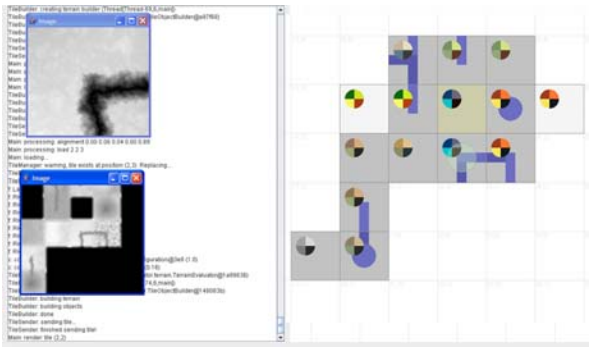

Figure 3. Java backend: a single heightmap (top left); joining tiles together (left); into consistent worlds (right).

Each tile contains a multitude of instantiated objects: pre-modeled items, creatures, effects, and lights. These are selected and placed using very similar techniques. For example, a *fire* themed tile will contain primarily *fire*-oriented objects, *fire* creatures, orange lights, and may have ash raining from the sky. Objects are also organized into categories according to size, so that there may be lots of small objects like shrubs and small rocks, a few larger objects like trees and rocks, and possibly one major one like a pagoda or a stone arch.

Figure 4. facing an attacking enemy in a *metal* area.

Although *Unreal* certainly was not designed for such an experimental game application, it still proved to be stable enough for the prototype. At the same time, the lack of access to the underlying engine code also limited the content generation. For example, we were not able to implement procedurally generated textures.

3.3 Design Issues

Entering *Charbitat*, the player controls the poisoned princess as she starts on a single tile surrounded by emptiness. Whenever the character reaches the borderlines of the existing world and steps into the void, a new part of the world, a new tile, is generated and added to the world. The Java backend keeps the overall world is consistent and manages the seed values, so that the world can be saved and reloaded to keep the quest coherent. It can even be swapped between different players. The main design issue in Charbitat was to map the player's actions onto the world generation. Without any interaction, the game world will never materialize and remain limited to the first start tile. In order to involve them in the space generation, players have to know what defines the underlying seed values, how they can affect them, and what the current seed values are.

3.3.1 Elemental seed variables

A key design decision was the use of the Taoist elemental system as the main variables in *Charbitat*. It consists of 5 basic elements: *wood*, *fire*, *earth*, *metal*, and *water*. The whole world of *Charbitat* is based on these five elemental variables of which three have been fully implemented. A *fire* area is filled with objects reflecting its nature (e.g. bright red fire flowers and autumnal trees), a *wood* area features different props (e.g. lush trees and shrubs); a combination of both values would generate a mixture of objects. The same is true for the underlying terrain that can be formed differently in each zone through different filters.

In addition, every entity in the world is connected to the five elements. There are five varieties of elemental beings; each one can be either infected (hunting predators) or healthy (passive prey). Predator types fight both the protagonist and each other. They constitute the enemy force to be dealt with. Whenever the player defeats a predator of a certain elemental value, the player's character statistics are updated accordingly. Defeating a lot of *fire* enemies, for example, leads to a strong tendency

towards the *fire* element in the character data set. The character data set is the "elemental value" of the main hero based on the tracked player interaction. It is one basis for the space generation. The other influence stems from the environment built so far. In order to guarantee a coherent world, tiles cannot jump from a pure *water* environment to a neighboring *fire* section. Transitions have to be gradual and the current and each adjacent tile have to affect any new world section. Thus, whenever a new tile is generated, the player-dependent character data is combined with the elemental values of adjacent tiles and the seed value for the new space section is calculated. Players influence the space generator through in-game actions but the system defines the limits and makes sure that the world stays consistent. At other occasion, we have outlined the space generation in dependency to the player behavior in more detail [10].

3.3.2 Character quests and space generation

Mapping the player behavior onto the space, *Charbitat* merges the creation and playing phases. The development of space depends on the world being played. Playing the game world and changing the elemental variables is key to the space generation. Without spatial progress and interaction by the player, the game world of *Charbitat* will not form.

To give this agency a purpose and distinguish it from open-ended concepts such as *Second Life*, players are set on a mission in the game world. They have to cure the heroine and find certain elemental core cells to heal the poisoned elements. Core cells are designer-created pre-modeled tiles that serve as goals for the player's quest to heal the princess. While the form of these core cells is fixed, their position is not predefined. The player has to generate the way to each core cell by playing in a certain style and thereby forming a procedurally generated and unique path. For example, the player must consciously act to maximize her *fire* element. That way, she can create an increasingly pure pathway of adjacent tiles that turn more and more into pure *fire* environments. Once the fire value reaches a certain threshold, the unique *fire* core cell is spawned and the player has located one quest goal.

Figure 3. Player hovering over one pre-modeled core cell

Because the creation of the path is part of the quest, the game world turns into a form of spatialized history of the play.

3.3.3 Informing the player

In order to make reasonable choices in the game and control the world generation, players need to be constantly informed about the current state of the elemental values and the player's influence on them. *Charbitat* provides this information in the only Heads Up Display (HUD) of the current game: a compass-like indicator. This "compass" does not offer any spatial orientation help but tells about the two main forces that affect a possible new world generation at the given moment: the player data set and the current tile the player is in.

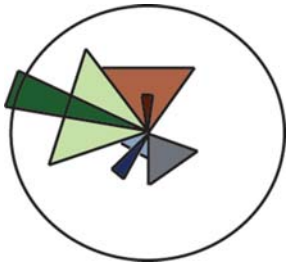

Figure 3. Creation compass; different elemental values and their relation between world and player state

Representing all five elements, the compass is divided into five color-coded sections. The wide triangles represent the player's elemental alignment that depends on the player's interactions. When the player comes to the edge of the current tile, the elemental value of the neighboring tile is represented

by the narrow triangles. While this visual presentation does tell the player know exactly what the next tile will look like, it informs about the current relationship between the values. This information is sufficient to know what to do within the game world. The player knows her own elemental values, and can predict the elemental configuration of new tiles. Players need to be informed about the tendency in the world generation and how they can adjust their playing style to steer this tendency towards their own goal. This kind of information is immediately accessible in the "compass." A strong *fire* value shows up as a huge red spike and a large red triangle in the compass. These symbols indicate a strong tendency towards a *fire* section. But the compass also shows opposing values: if one stands in a *water* dominated section and the character values are strongly concentrated to a *fire* value, the next tile will be more of a *fire* type but still co-shaped by the surrounding *water* dominance.

While the designer-created core cells serve as goalposts for motivating a player's journey, procedural terrain filters, object positioning, and spawn points of entities support a continually growing unique game world. By tracing the player's behavior and informing the player of this trace in the elemental compass, the player actions are mapped onto the evolving world. In order to accomplish this combination of procedural space with player interaction, we have mixed the various forms of space generation outlined above. Play and editing are combined and the player's agency is expanded from the immediate fight to the longer-term world generation. One is not only involved in the action inside the game world but also in the forming of the game world itself.

4. DISCUSSION AND OUTLOOK

Charbitat answers a number of design challenges that are posed by procedural content generation and builds bridges between different approaches of world creation. It achieved its main goal to utilize the power of procedural space generation and combine it with a play experience. Perhaps the most innovative aspect of *Charbitat* is this mapping of the character behavior onto space. The player creates a game world while playing in it. This introduces a relatively new interactive option to game design that leads to new questions – most of them still in dire need of more research.

Once Pandora's Box of procedural content generation is opened and frameworks about its implementation established, the possibilities and challenges are legion. Music, game characters, in-game camera – all can be generated in reference to the underlying game space. For the game world itself, two main points stand out.

One pressing question is that of context. The mere fact, that we can generate space does not mean that this space necessarily makes any sense or is of any value to the player. Procedurally generated game worlds can stretch into infinity but the meaning of each single locale can be thinned out by that. It is a prime argument for designer-created worlds that they provide control over possible story developments and a tight structuring of the player experience.

How can we fill these endless virtual procedural playgrounds with significance and context? *Charbitat* features the quests for the core cells, which lend structure, motivation, and direction to the player's actions. But these quests were still pre-defined. In a first step towards procedurally generated spatial quests, Ashmore has implemented a key-lock puzzle generator into the world of *Charbitat* [2]. It introduces procedurally generated basic quest elements that can be spawned in the generated worlds. Players can encounter basic key-lock tasks that send them on the search for a key (a solution) to overcome a designated lock (a threshold). Ashmore presents a basic groundwork for procedurally generated quests that project a context onto the game world.

A second and related spatial question is that of orientation. Navigating in a potentially endless space is a challenging task. Although we have not yet applied any direct navigational model (e.g. as suggested by [15]) to *Charbitat*, we have incorporated the necessary expressive vocabulary to address this task. *Charbitat* not only arranges entities within each individual tile of the game world, but also provides for larger, overarching structures that span over multiple sections. Rivers, cliffs, walls, and roads are elements that continue seamlessly from one tile to another and can form obstacles and landmark features. The combination of local elements and these larger entities answers

Lynch's call for a legible space [7]. In addition, these landmark features can work as thresholds in the quest and have significance for the overall gameplay. Rivers, for example, remain thresholds or obstacles until the player finds the "swim" key.

It is such a combination of generated space and inherent meaning for action that provides a wide spectrum for more work. *Charbitat* presents one example for the realization of player-shaped procedural space generation. On top of the framework applied here one can imagine a second one that could refer to theories like Alexander's pattern language or Hillier's space syntax. Ultimately, the mere creation of the game space cannot be the end goal of any game world, but it has to be the shaping of experiences in that space.

5. ACKNOWLEDGMENTS

The *Charbitat* project received generous financial support from Turner Broadcasting; former members of the project team include Jason Alderman, Matthias Shapiro, Katherine Compton, and Martin Walsh.

6. REFERENCES

[1] Aarseth, E.J. Allegorien des Raums: Räumlichkeit in Computerspielen *Zeitschrift für Semiotik*, 2001, 301-318.

[2] Ashmore, C. Key and Lock Puzzles in Procedural Gameplay *Ivan Allen College*, Georgia Institute of Technology, Atlanta, 2006.

[3] Ebert, D.S., Musgrave, K.F., Peachey, D., Perlin, K. and Worley, S. *Texturing & Modeling. A Procedural Approach. Third Edition*. Morgan Kaufmann Publ., Amsterdam, Boston, London, New York, Oxford, Paris, San Diego, San Francisco, Singapore, Sydney, Tokyo, 2003.

[4] Jenkins, H. Game Design as Narrative Architecture. in Harrington, P. and Wardrup-Fruin, N. eds. *First Person: New Media as Story, Performance, and Game*, MIT Press, Cambridge, MA, 2004, 118-131.

[5] Juul, J. *Half-Real. Video Games between Real Rules and Fictional Worlds*. The MIT Press, Cambridge, MA, London, 2005.

[6] Juul, J. The Open and the Closed: Games of emergence adn games of progression. Mäyrä, F. ed. *Computer Game and Digital Cultures*, Tampere University Press, Tampere, FI, 2002, 323-329.

[7] Lynch, K. *The Image of the City*. MIT Press, Cambridge, MA, 1960.

[8] Mateas, M. Interactive Drama, Art and Artificial Intelligence, Carnegie Mellon University, 2002.

[9] Murray, J.H. *Hamlet on the Holodeck. The Future of Narrative in Cyberspace*. MIT Press, Cambridge, MA, 1997.

[10] Nitsche, M., Alderman, J., Ashmore, C., Compton, K. and Shapiro, M. The Many Worlds of Charbitat *Game Set Match II*, Delft, 2006.

[11] Novak, M. Dancing with the Virtual Dervish: Worlds in Progress. in Moser, A.M. ed. *Immersed in Technology*, MIT Press, Cambridge, MA, 1996, 303-307.

[12] Perlin, K. Can there be a Form between a Game and a Story? in Wardrip-Fruin, N. and Harrigan, P. eds. *First Person: New Media as Story, Performance, and Game*, MIT Press, Cambridge, MA, 2004, 12-18.

[13] Salen, K. and Zimmerman, E. *Rules of Play: Game Design Fundamentals*. MIT Press, Cambridge, Mass., 2003.

[14] Taylor, L.N. Video Games: Perspective, Point-of-View, and Immersion *English*, University of Florida, Gainesville, 2002, 43.

[15] Vinson, N.G., Design Guidelines for Landmarks to Support Navigation in Virtual Environments. in *SIGCHI Conference on Human Factors in Computing Systems*, (Pittsburgh, PA, 1999), ACM, 278-285.

[16] Wolf, M.J.P. (ed.), *The Medium of the Video Game*. University of Texas Press, Austin, 2002.

[17] Wonka, P., Wimmer, M., Sillion, F. and Ribarsky, W. Instant Architecture. *ACM Transactions on Graphics*, *22* (3). 669-677.