**Abstract**

Manual game content creation is an increasingly laborious task, with each advance in graphics hardware, a higher level of fidelity and detail is achievable and, therefore, expected. Procedural methods are a promising but underused alternative to manual content creation. Commonly heard drawbacks are the randomness of and the lack of control over the output and the absence of integrated solutions. This paper surveys procedural methods applied to terrain modelling, evaluating realism of their output, performance and control users can exert over the procedure.

Most games that use procedural level generation aim to keep certain characteristics in the generated levels. This helps the players to quickly adjust to the new worlds. Sometimes these characteristics might be certain buildings, certain attributes tied to certain biome types, the main point being that there is always something that is recognizable.

The purpose of this project is to explore a practical application procedural generation through the use of a basic technique of this process, a Procedurally generated terrain, to create a simple PCG Game. The Game is able to create a variety of cases with unique details (such as water bodies & vegetation), and has the potential to be scaled up to accommodate additional information or assets as needed, but with the implementation of an input system for a user's answers it serves as a functioning programming example of the procedural generation process. It’s helpful in generating new worlds, terrains, characters, enemies and so-on for players to interact with, but it can also be used to great effect for practical purposes, and even to create music and art.

**Introduction**

Game content construction and generation are laborious and expensive. Procedural content generation (PCG) aims at generating game content automatically using algorithms, thus reducing the cost of game design and development. Moreover, PCG can also provide a way to generate personalized games that can adapt content according to a player’s preference and optimizing their gaming experience. For those reasons, PCG is getting increasingly popular in game development field.

The keystone of this methodology is the concept of randomness: Using a few parameters, the application of PCG ensures the creation of a high number of possible contents of a game, all differing from each other. It is important to define the meaning of the word “content” because according to its definition we can distinguish whether a content falls within the domain of the PCG or in other domains. The results from the application of PCG algorithms can be all kinds of elements affecting the gameplay: terrain, maps, layers, stories, dialogues, quests, characters,

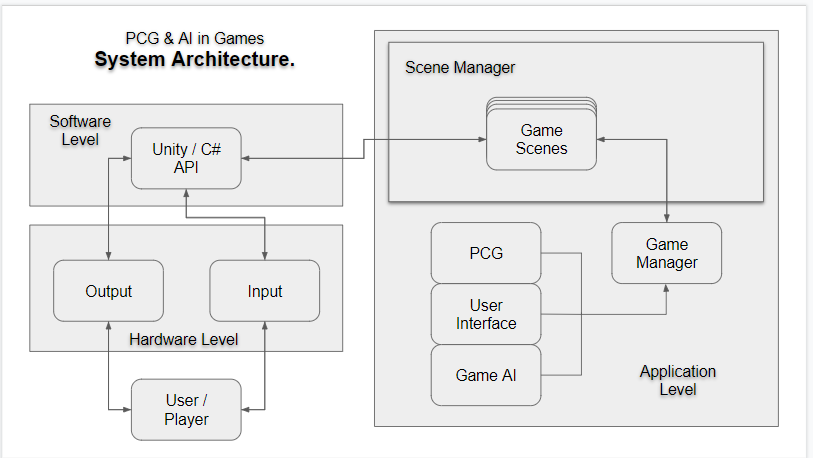
rules, dynamics, or weapons.

There is still not a clear distinction between PCG and PG (procedural generation), and often the two terms are used as synonyms, since they are based on the same algorithms to achieve different results. The main difference lies in the results generated: PCG generates content consisting of several components, while each individual component can be generated from the PG. PG also creates content that does not affect the gameplay directly, for example, procedural textures and procedural animation. A texture created by procedural generation techniques relates to PG, while the set of textures that form the setting of a level created by procedural generation techniques relates to PCG.

The advantage of using PCG is that content represented procedurally is “compressed” as long as it is not required by the gameplay. The adventure game Elite is the example par excellence: It could handle thousands of star systems in a dozen kilobytes of memory, representing only a few

numbers of planets in compressed form.

The use of procedural content generation (PCG) techniques in game development has been mostly restricted to very specific types of game elements. PCG has seldom been deployed for generating entire game levels, a notable exception to this being dungeons: a specific type of game level often encountered in adventure and role playing games. Due to their peculiar combination of pace, gameplay, and game spaces, dungeon levels are among the most suited to showcase the benefits of PCG. In general, what current procedural content generation methods are missing is not performance, but more powerful, accurate, and richer control over the generation process. Recent research results seem to indicate that gameplay-related criteria can provide this high-level control. However, this area is still in its infancy, and many research challenges still lie ahead, e.g., improving the intuitiveness and accessibility of such methods for designers. We also observe that more research is needed into generic mechanisms for automating the generation of the actual geometric models. We conclude that the foundations for enabling gameplay-based control of level generation are worth being researched, and that its promising results may be instrumental in bringing PCG into mainstream game development.

**Architecture of the Game**

**White Perlin Noise algorithm to generate a realistic island terrain surrounded by water.**

Noise is a series of random numbers, typically arranged in a line or a grid. In signal processing, noise is typically the unwanted aspect. In a noisy room it’s harder to hear someone than in a quiet room. Audio noise is random numbers arranged in a line (1D). In a noisy image it’s harder to see a pattern than in a clean image. Image noise is random numbers arranged in a grid (2D). You can also have noise in 3D, 4D, etc.

Procedural map generation is to generate a set of outputs that have some things in common and some things different each time.

All the Terrain maps have a lot of similarities: the set of biomes, the size of the grid, the average sizes of biomes, the heights, the average sizes of caves, the percentage of each type of rock, and so on. But they also have some differences: where the biomes are placed, the location and exact shapes of caves, the placement of gold, and so on. As a designer, we need to decide which aspects are the same and which aspects will vary, and how they will vary.

Non-uniform random selectors (Random Number Generators) change what happens in each grid location in isolation but instead we want something where the random selection in one location is related to the random selections in nearby locations. This is called “coherence”. That’s where noise functions come in. They give us a set of random numbers instead of one at a time.

**Perlin Noise**

Perlin noise is a [procedural](https://en.wikipedia.org/wiki/Procedural_generation) [texture](https://en.wikipedia.org/wiki/Texture_(computer_graphics)) primitive, a type of [gradient noise](https://en.wikipedia.org/wiki/Gradient_noise) used by visual effects artists to increase the appearance of realism in [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics). The function has a [pseudo-random](https://en.wikipedia.org/wiki/Pseudo-random) appearance, yet all of its visual details are the same size. This property allows it to be readily controllable; multiple scaled copies of Perlin noise can be inserted into mathematical expressions to create a great variety of [procedural textures](https://en.wikipedia.org/wiki/Procedural_texture). Synthetic textures using Perlin noise are often used in [CGI](https://en.wikipedia.org/wiki/Computer-generated_imagery) to make computer-generated visual elements – such as object surfaces, fire, smoke, or clouds – appear more natural, by imitating the controlled random appearance of textures in nature.

Figure 7: Perlin Noise

It is also frequently used to generate textures when memory is extremely limited, such as in [demos](https://en.wikipedia.org/wiki/Demo_(computer_programming)), and is increasingly finding use in [graphics processing units](https://en.wikipedia.org/wiki/Graphics_processing_unit) for [real-time graphics](https://en.wikipedia.org/wiki/Real-time_computer_graphics) in [computer games](https://en.wikipedia.org/wiki/Computer_game).

We can creating Terrain using Perlin noise, look at the perlin noise map carefully. We can see a bunch of white & black regions. What connects these regions together is gradual grey region between a complete black or a complete white spot. This coherence is necessary to create realistic terrain.

If we consider the darker spots as the regions with lower height & the lighter spots as regions with greater height. (height depends on how dark or light a region is, pitch black = 0% height, Grey = 50% height(optimal) & white = 100% height).

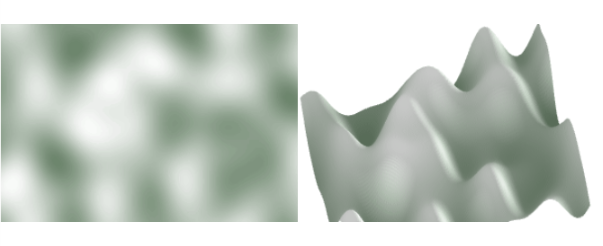


Figure 8: 2d/3d Perlin Terrain

Now, generating terrain using the perlin noise map gives us something like the above figure. The image to the right is called a height map. The above height map looks rather smooth to be used as a terrain. To make Height maps more interesting we’re going to add noise at different frequencies

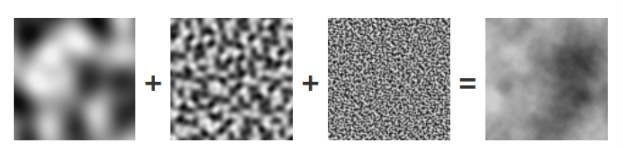


Figure 9: Multilayer perlin noise

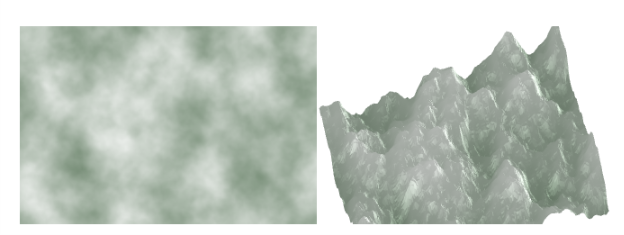
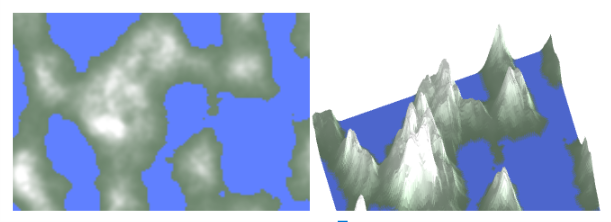
Now generating the terrain from the output noise will give us a rugged terrain with mountains & valleys.

Figure 10: Generating height map from perlin map Terrain

Now, separating the terrain into land & water based on their elevation, we get something like the below figure.

Figure 11: layer implementation terrain



**Mesh based system.**

Meshes make up a large part of your 3D worlds. Aside from some [Asset store](http://unity3d.com/unity/asset-store/)  
plugins, Unity does not include modelling tools. Unity does however have great interactivity with most 3D modelling packages. Unity supports triangulated or Quad angulated polygon meshes. Nurbs, Nurms, Subdiv surfaces must be converted to polygons.



Figure 12: Mesh Objects

A class that allows creating or modifying meshes from scripts. Meshes contain vertices and multiple triangle arrays. The triangle arrays are simply indices into the vertex arrays; three indices for each triangle.

**A seed based Procedural Terrain Generation Algorithm on a mesh based game.**

Using a string input as a key (seed) to create a procedural generated terrain. This is different from noise maps, as we need to mention the key which will be used to create the terrain.

The seed value inserted by the user is converted into hash code & passed into into a System.Random object.

This System.Random object has a .Next() method which provides us with a constant stream of random values which is checked against a threshold value. Surpassing the threshold value qualifies as 1 (true) else we get 0 (false).

This is stored in a 2D integer array “ int[,] map ”.

Ex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |

Table 1:Seed for PCG

s

Then we use a smoothing algorithm to group the 1’s together & the 0’s together. Then using a complex mesh generation process we convert the 2D integer array called “map” into an actual map using meshes, we get the following map.

Seed value = Steven

Threshold value = 50

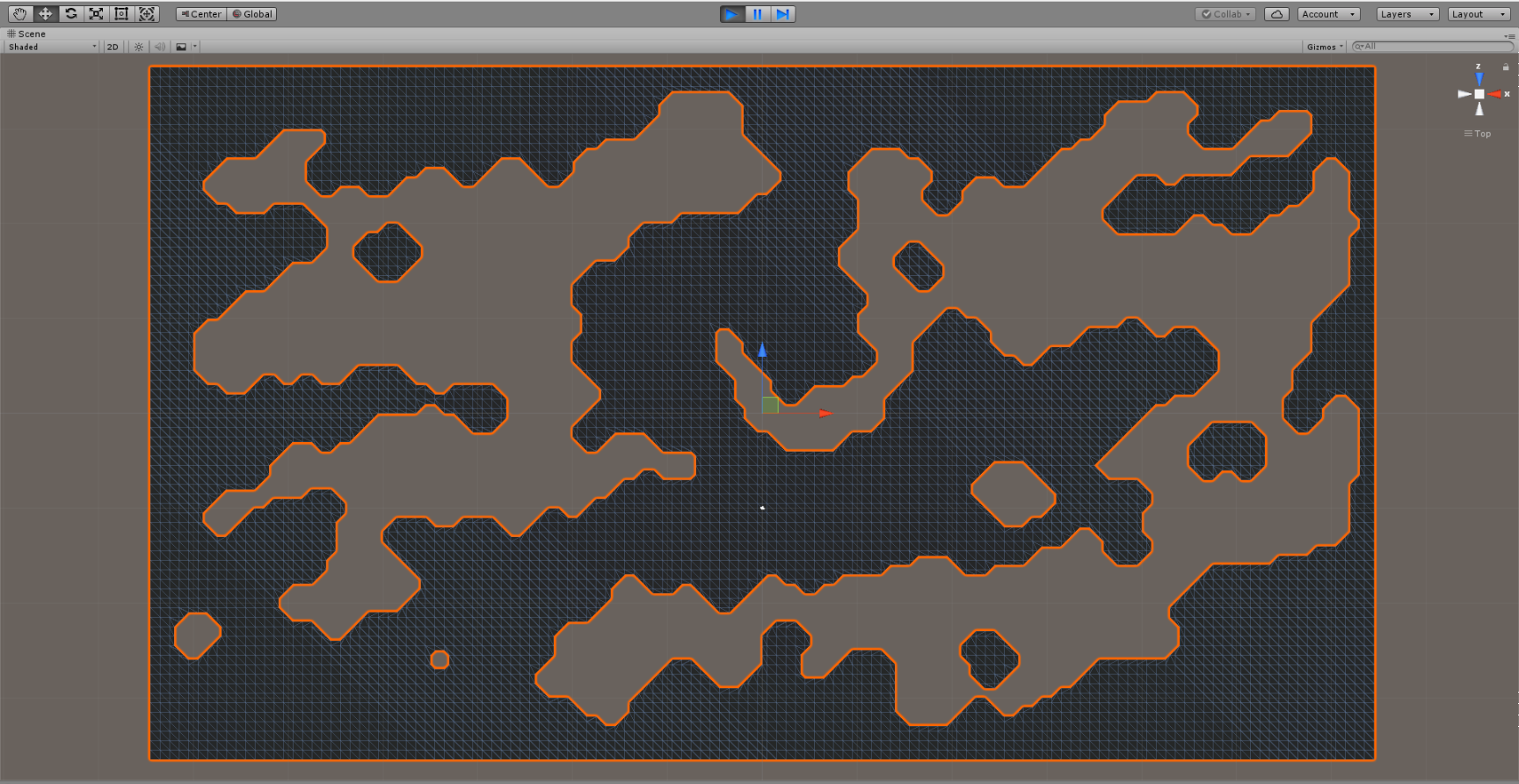


Figure 13: Mesh based terrain

**A TAXONOMY OF METHODS FOR PROCEDURAL CONTENT GENERATION**

PCG has probably been used by too many people with too many different perspectives to arrive at a definition of procedural content generation that everybody agrees on. A graphics researcher, a designer in the game industry and an academic working on artificial intelligence techniques would be unlikely to agree even on what “content” is, and even less on which generation techniques are interesting. We can group, according to the latest classification of Hendrikx et al. (2013), the type of content that can be produced by procedural generation techniques into five main classes:

* Game Bits
  + Textures can be generated procedurally through a variety of techniques, including PRNG, IF,SA, and CS techniques. Perlin noise and other PRNG-based techniques are commonly used for texture generation. Noise-generated textures can be mapped easily on complex objects, unlike raster 2D images. The implementation of noise is relatively simple, and is present in many software shaders and hardware graphics cards, such as NVIDIA’s. Pattern-based procedural texturing is an IF-based technique to imitate the level of detail that a large resolution texture can provide .This technique also requires a procedural algorithm for breaking the regularity of the pattern, without putting limitations on the mesh. Convolution filters such as sharpening or smoothing can improve game textures, even dynamically. SA techniques such as tiling and layering are common in games; their small computational and memory requirements lead to their adoption by many commercial RTS games. To imitate different ornamental styles, geometric decorative patterns can be developed using a range of mathematical algorithms. From the AI techniques, Genetic Algorithms have been used [Sims 1991] to generate textures procedurally. Chromosomes describe texture generating functions, such as noise, as expression trees. Crossover occurs by combining branches from the expression trees. The fitness of a gene (texture) is evaluated against a goal set by the use.
  + Vegetation is used in many games for a more realistic and thus immersive look. For example, trees can cover the ridge of a mountain and reeds can be found along river banks. The presence of vegetation is not merely aesthetic; it may serve as hiding place or raw material for inventive players, and clarify the climate in which the player operates and thus lead to changes in gameplay. Thus, vegetation can be procedurally modeled using GG, SA, and CS techniques. For example, self-similar plants such as ferns and cauliflower can be generated using fractals, and L-systems can be used to model non-self-similar vegetation–symbols may represent plant parts such as the trunk or a leaf, and rules may determine the “growth” of the plant from the initial trunk.
  + Buildings are essential to represent urban environments in games. Even more so than for vegetation, buildings may be significant to players and often influence gameplay players often plan their game activities in relation with buildings, for example by ending resource collection activities closer to warehouses or marketplaces. Modern game buildings need to be diverse yet belong to a unitary architectural style; even fictional environments such as World of Warcraft’s require interesting and believable buildings to maintain immersion. Procedural generation of buildings aims at generating many different buildings from a limited set of rules or user-generated content. GG techniques have been used to generate buildings. For example, the extended L-systems used by City Engine starts from a symbol representing a bounding box for the building, then iteratively transforms the current building or generates new buildings; the generated buildings belong to the same architectural style.
  + Behavior is the way in which objects interact with each other and the environment, for example by breaking or exploding when hit. The behavior of objects makes a game more lively and interesting. Procedural behavior is often employed in games to create the illusion of complexity; in response, players may find creative ways to bend or use object behavior.
  + Fire, Water, Stone, and Clouds are often used in games to create a more believable world. In early games, the role of these elements of nature was purely decorative, and only textures and sounds were necessary for this type of content. However, recent advances in computation and modeling have made more common the use of detailed, realistic, and interactive representations of these elements. The procedural generation of a variety of complex elements, in particular fire, water, stone, and clouds, has received much attention in the past. For brevity, we focus here on the procedural generation of clouds. Perlin noise and other PRNG techniques, and IF and CS techniques can be used to generate a realistically looking sky.
* Game Space
  + The game space is the environment in which the game takes place, and is partially filled with game bits among which players navigate. The game space plays a major role in creating dramatic experience, as players often construct their interpretation of the game starting from the game space.

we consider space as only the one, two, or three-dimensional area where residing artifacts have relative position and direction. Our distinction between indoor and outdoor maps stems from aesthetic, cultural, and technical considerations. Aesthetically, outdoor maps of natural and even urban areas have consistent themes, which is less likely for indoor maps. Culturally, indoor maps increasingly feature many inter-related, possibly one-of-a-kind artifacts. Technically, outdoor maps require different algorithms and data structures than indoor maps to generate and operate.

* + Indoor Maps are depictions of the structure and relative positioning of indoor space partitioned into rooms. Rooms may be interconnected by corridors, overlapped in layers interconnected by stairs, and grouped altogether in dungeons. Another form of indoor maps are caves, which can have varying and unusual geometry. Yet another form of indoor maps focuses on human-like buildings, where the position and size of rooms, and sometimes their content, are important. The game genre Multi-User Dungeons (MUDs), the ancestor of today’s popular MMORPGs, is a type of game in which indoor map navigation is central to gameplay. The platform game Prince of Persia, which is set inside a multi-leveled dungeon, is one of the few to make extensive use of indoor maps.
  + Outdoor Maps are depictions of the elevation and structure of an outdoor terrain. It is common for games with outdoor maps to also have indoor maps; due to important technological differences in the representation and rendering of outdoor and indoor maps, the transition between the two is often made discrete. For example, World of Warcraft has large outdoor areas and numerous indoor areas players transition between them through the use of special entrance areas and teleportation portals. Different approaches have been suggested to give the designer finer control of the produced environments. Declarative modeling and procedural sketching allow the designer to efficiently create assets while staying within an interactive workflow. Terrain synthesis using procedural brushes allow for local control, up to full automatic terrain generation. Software agents are another technique for creating different types of terrain while leaving the designer in control.
  + Bodies of Water such as rivers, lakes, and seas are often used as map obstacles or even as interactive game space. Features such as mountains and sea shores are important to players, and have led to the development of specific procedural generation techniques. Among these features, we focus in this survey on bodies of water; although mountains may be another feature of interest, in contrast to water they are usually un-traversable map features. Procedural generation of bodies of water, such as rivers, lakes, and oceans, is often done during or directly after the generation of the outdoor height map. The shape of the river or sea is either determined by the height map (for example, water starts in mountains and follows the elevation gradient), or the water shapes the height map. The generation of lakes and oceans is usually done after height-map generation by a flooding algorithm, which “floods” the terrain starting from the lowest point, or by simply setting the water level to a specific height–all heights below the water threshold are considered to be under water.
* Game Systems

The use of complex systems theory and modeling to generate or simulate parts of a game is not uncommon. The use of game systems can make games more believable and thus appealing. Many systems used in games are similar to textbook complex systems and models.

* Ecosystems govern the placement, evolution, and interaction of flora and fauna through algorithms and rules. The designers of Ultima Online, a game which established the game genre of MMORPG,focused on generating large ecosystems that included complex food chains.
* Road Networks form the basic structure of an outdoors map, serving different purposes such as transportation between points of interest, and structuring of and transportation within cities. The main difficulties in generating road networks are finding the right balance between randomness and structure, and conveying the view of the game designer about places of interest, such as remoteness (through interrupted roads), difficulty (labyrinthine roads), and importance (broader roads).
* Urban Environments are large clusters of buildings where many people live together and interact with their surroundings. Realistic cities take centuries to grow, and evolve by influence of the people that live there. Procedural algorithms often take a different approach by first generating the road networks, then dividing the terrain between the roads in building lots, and at last generating buildings on these lots. The realism of these generated cities can be improved by applying effects like erosion caused by people and weather.
* Entity Behavior Many types of complex player-environment interaction need to be possible to make the player experience that a virtual world is life-like. Entities such as non-playable characters(NPC) that interact with the player are a powerful tool to achieve this illusion. Procedurally generating entity behavior based on player action and interaction has the potential to create immersive and realistic experiences. Not only player interaction requires complex entity behavior; Group movement patterns are examples where procedural algorithms could achieve more realistic results.
* Game Scenarios

Game scenarios describe, often transparently to the user, the way and order in which game events unfold:-

* Puzzles are problems to which the player can find a solution based on previous knowledge or by systematically exploring the space of possible solutions embedded in the problem . For puzzles, the process of finding the solution is the game and thus a rewarding experience. Many quests present in commercial games may be expressed as a sequence or graph of puzzles. Examples of puzzles are riddles, crosswords, and chess endings. The size of the solution space and the previous experience of the player largely determine the difficulty of a puzzle.
* Storyboards are design aids for the game developer or player. Storyboards are often presented as comics, with sequential panels describing scene events through a visual/textual hybrid. Storyboards can also be used to entertain and guide players, for example through cut-scenes interleaved with normal gameplay. Depending on how they are produced and used, storyboards may also be an example of derived content.
* The Story of a game is often key in creating a good gaming experience. It keeps the player motivated, presents a logical basis for the events that unfold in the game, and provides a goal for the player to accomplish. Besides revelation through cut-scenes and quests, the game story may be embedded entirely in the game universe. Albeit rare in commercial games, perhaps mostly in text-based adventures such as Skoto’s Castle Marrach and in artistic games such as Facaade, Story-intensive quests may not be easily represented as graphs of individual puzzles; for these quests, the dramatic arc plays an important role.
* The concept of Levels is used in nearly every game as a separator between gameplay sequences. For example, a level in a platform game, such as Nintendo’s Super Mario World, Rare’s Donkey Kong Country, and Sega’s Sony the Hedgehog, would consist of a separate, playable game space in which the player may be required to move from the start to the end position through a series of movements on and jumps to/from platforms, while completing (optional) tasks and avoiding obstacles. Level generation is currently one of the most popular types of PCG-G. Although nearly every genre can benefit from generated levels, 2D platformers and puzzles have especially attracted attention. By generating levels for 2D platformers based on the concept of rhythm: the pattern of hand movements of a player when playing the game. The personalized levels can be generated online for platform games based on a player model. More general than platforms, [Dormans 2010] uses a grammar to create a mission structure using graphs, which is then translated to a 2D level by using a shape grammar.
* Game Design

The design of a game is comprised of content such as rules (what can be done in the game?) and goals (what is the player trying to achieve?); an aesthetic component, such as a dramatic arc or a graphical theme, are also important elements in design .A game can be seen as an instance of a game design, in which the parameters of the rules and of the goal have been set. A game design can refer to game content of all the types, including the recursive reference to other game design content. Game design can be complemented by (semi-)automatic game generation or by providing tools that help the designer convert ideas into game design content.

* The System Design of a game entails “the creation of mathematical patterns underlying the game and game rules”. An example of generating game rules is the generator by which generates symmetric, chess-like games. One of the main challenges in system design generation is ensuring that the rules are balanced between all players.
* The World Design of a game is “the design of a setting, story, and theme”. An example includes the generation of novel games based on beforehand unknown story structures by [Hartsook et al. 2011].
* Derived Content is defined as content that is created as a side-product of the game world. Generation of derived content can greatly increase the feeling of immersion the player has with the game world, as players record their in-game experiences for review inside or outside the game; this “game beyond the game” [Garfield 2000] was the basis of the “metagame” notion (ibid).
* News and Broadcasts A game may show its players news items based on their actions and other changes in the game’s universe; the same news items could then be presented as television broadcasts and newspaper articles. Similarly, game sessions may be broadcasted—a popular part of television schedules in Asia and, more recently, US and Europe.
* Leaderboards player ranking tables are popular for a variety of game genres and are used by fan-sites to serve millions of players.