Engineering emergence: applied theory for game design

Dormans, J.

Citation for published version (APA):
Storytellers in all media and all cultures are, at least partially, in the business of creating worlds.

Scott McCloud (2000, 211)

All games have rules and mechanics, but different types of games utilize different types of mechanics. As was discussed in Chapter 1 mechanics to govern physics, economy or levels are of a different nature and therefore must be treated differently. The previous chapter discussed the Machinations framework, diagrams and tool to represent discrete game mechanics that govern a game’s internal economy, and to correlate their structure to emergent gameplay. It is a framework that works well for certain types of games. As was pointed out, the Machinations framework does not incorporate level design. Yet for level-driven games, such as most action-adventure games and most first-person shooter games, quality level design is critical. Unfortunately, level design has been studied less extensively than game mechanics dealing with physics or economy.

Level design is one of the aspects of game design where the different disciplines of art, design and technology converge. Level design combines all of these aspects as it works with core mechanics and art to create a spatial and temporal experience within the technological boundaries of the game’s software-architecture. Level designers are usually responsible for turning gameplay elements and art assets into game spaces that create a concise and compelling experience. Often this requires scripting of simple behavior to create mechanics to control player progress. The specialized role of the level designer was one of the last roles to emerge from the growing game industry (Byrne, 2005; Fullerton, 2008). As a result there is no definite set of principles guiding level design yet, although a few common patterns and strategies can be found in different sources.

In this chapter I will present the Mission/Space framework. This framework provides a structural perspective on levels and players’ progression through them. However, this chapter starts with investigating a few existing structural models for level design that are suggested by others in the game development community. The first section presents an overview and a discussion of common typologies for level layouts. The second section explores the models that focus on levels as a set or series of challenges to be overcome by players. The Mission/Space framework builds on these two different perspectives and includes separate graph representations for both the mission and space that comprise a level. With a detailed analysis of a game level from The Legend of Zelda: Twilight Princess I will illustrate and discuss the relation between the missions and
spaces. This discussion extends into the final sections of this chapter where I will show how the Mission/Space framework can be leveraged and expanded to get a better grip on mission, spaces and their relation. In chapters 6 and 7 I will combine the Mission/Space framework with the Machinations framework in order to integrate structures of emergence and progression more closely, and to unify both in an formal approach to game design that is based on model transformations, respectively.

5.1 Level Layouts

Level design has not been studied extensively. Yet, it is generally acknowledged that levels benefit from having a relatively simple gestalt or purpose, especially when this gestalt matches the intended gameplay, rhythm and pacing. To this end, a number of scholars of games and interactive storytelling categorized spatial structures frequently found in games. An overview and comparison of four of these typologies is presented in figure 5.1. This figure lists all the categories in each typology horizontally. Their relative position and size on the horizontal axis suggests overlap between the categories of each typology. Figure 5.2 illustrates each category with a schematic representation of their structure.

Of these typologies, Marie-Laure Ryan’s specifically concerns itself with interactive story structure, while the others concern themselves with game levels. Still, the similarities between many of the structures they describe is striking. In fact, it is a common observation that in games, stories are, at least partially, structured spatially instead of temporarily (Jenkins, 2004). This causes some confusion whether these categories concern themselves with level geography or topology. As Ryan focuses on interactive storytelling, her categories are clearly topological, but the other three typologies are much more geographical in nature. From these typologies it appears that in level design topological structures and geographical structures are frequently isomorphic or at least often treated as such.

The different categories in the typologies in figure 5.1 can be grouped into three super-categories. The categories on the left all concern tightly controlled structures that are fairly linear and directed; players usually cannot go back in these structures. Categories that describe railroading and branching techniques fall in this group. The categories in the center are all networks of some sort. These categories offer players more freedom to explore the world but still restrict players’ positions to a restricted set. The categories on the right feature more open world that impose few restrictions on player movement.

One observation that can be made from figure 5.1 is that, although there definitely is some overlap between the different typologies, their differences are also quite prevalent. What is a main category in one typology might be a special case of a category in another. For example, ‘bottlenecking’ with Byrne can be regarded as a special instance of ‘parallel structures’ with Adams and Rollings. Categories present in most typologies are omitted in another: Adams and Rollings have no category that covers the ‘tree/branching structures’ with
Ryan and Byrne. Other categories, such as ‘dynamic levels’ with Byrne and “grids” with Schell, probably have no place in the typology at all, as they are separated by their mode of creation and not their topological structure.

The categories within some typologies do not appear to be on the same level as some other categories within the same typology. For example the ‘hub-and-spoke’ structure can be regarded as a special case of ‘network layout’ in Adams and Rollings typology. It might be a quite common structure in games, but it does not capture nearly as many different structures as the network layout does. At the same time, other specialized structures can be found that do not feature in any of these typologies. For example, a variation of the ‘hub-and-spokes’ layout: the ‘hub-with-loops’ layout, in which the player progresses through longer linear and directed level sections before returning to the hub can also frequently be encountered in games (see figure 5.3). Three of the nine main dungeons in THE LEGEND OF ZELDA: TWILIGHT PRINCESS\(^1\) have this type of layout: Link, the game’s protagonist, can often quickly return to a central location after activating the trigger that unlocks a new section. The other six main dungeons have a normal hub-and-spoke layout. The point is, that from the typologies presented in figure 5.1, one cannot be certain to classify the hub-and-loops structure as a special combination of hub-and-spokes and linear structures or as a category of its own. The typologies do not provide enough guidance to safely argue one way or the other.

These problems indicate that understanding levels in games is not as straightforward as it may seem. The reason for this may be, as is one of the main points of this chapter, that levels do not constitute a single structure: they contain

---

\(^1\)This includes the ‘Hyrule Castle’ level that builds up to the final conflict with the game’s main antagonist.
Figure 5.2: Illustrations of the level typologies found in (Ryan, 2001; Byrne, 2005; Adams & Rollings, 2007; Schell, 2008)

Figure 5.3: A hub with loops structure
at least a geography and a topological structure. Separating a level’s topology from its geography will help to create much clearer perspective on game levels. In this respect, I consider the layouts discussed above to be geographies. A level’s topology would focus more on the logical structure of the player’s tasks.

5.2 Tasks and Challenges

A level’s topology concerns itself with the tasks and challenges players must perform in order to complete a level. These tasks or challenges are usually fairly straightforward tests of the player’s abilities. They might take the form of puzzles, fights, traps or hidden objects that need to be collected. Ed Byrne suggests structuring these tasks in cell-diagrams outlining the game’s flow and highlighting the different player tasks (see figure 5.4). These cell diagrams are simple informal structures that read almost like a storybook that help design a level’s layout or rhythm. Cell diagrams focus on a game’s logical and temporal structure instead of its spatial layout (Byrne, 2005).

The temporal dimension of games and the players experience is also emphasized by the approach of Ben Cousins (2004). Cousins introduces the idea of a hierarchy to order the gameplay experience. In a detailed analysis of the game SUPER MARIO SUNSHINE he divides the experience in five layers, where the top layer constitutes the whole game. The subsequent layers describe the individual missions, mission elements, input elements, and primary elements. The middle layers intuitively correspond to player plans and intentions. In cognitive science they would have been called basic level categories for the games actions; these actions are most accessible to players and would typically be the actions players would use when describing the gameplay. The lowest layer represents the actions made possible by the game mechanics and interface; they correspond to the buttons pressed by a player and the resulting actions of the avatar, such as jumps or simple moves. Cousins stresses the importance of the quality of these
low level actions as the player spends much time performing them. Using simple metrics Cousins shows that 55 percent of the primary elements in a four minute session of Super Mario Sunshine consisted of running forward and changing direction.

The ‘hierarchy of challenges’ described by Ernest Adams and Andrew Rollings was directly inspired by Cousins’ analysis. However, where Cousins focuses on individual sessions, with the hierarchy of challenges Adams and Rollings abstract away from individual sessions and represent all possible trajectories in a single image. In this hierarchy all the game’s challenges are ordered into a layered structure representing what a player needs to accomplish to complete the game (see figure 5.5). To build a hierarchy of challenges, inside knowledge of the level’s design is required. Some challenges can be performed in different order or even simultaneously. At the lowest level in the hierarchy are the atomic challenges: the micro actions the player needs to perform to get ahead. At higher levels in the hierarchy there are goals of individual sections and missions. At the highest level the game’s ultimate goal resides. Adams and Rollings discuss the different needs of visibility among the levels: games need to be very clear in their high level and atomic level challenges while they can be less clear for intermediate challenges. They also stress that presenting a player with simultaneous atomic challenges considerably increases the game’s difficulty (Adams & Rollings, 2007).

Creating simultaneous or alternative options in the hierarchy of challenges leads to a potentially more varied gameplay. However, this is more difficult in the higher levels of the hierarchy than it is in the lower levels (see also Arinbjarnar & Kudenko, 2009). Deus Ex is a good illustration of this. In this game the player has to get through a series of connected missions. At most low level challenges the player is presented with options: to deal with a guard the player might use stealth or violence. As players progress they choose how the game’s protagonist develops his skills: players can choose to specialize in different strategies to
overcome common challenges, such as the the use of force or stealth. Yet, at the high end of the challenge structure there are only few options and branches. It is only at the very end of the game that the player gets to choose between one of three alternative endings.

5.3 The Mission/Space Framework

The two different approaches, focusing on the geographic layout of a level on the one hand and on the sequence of tasks on the other, suggest that in a level both structures exist at the same time. These structures are superimposed onto each other and as a result it is all too easy to confuse one with the other or to take their interrelation for granted. The Mission/Space framework formalizes both perspectives and foregrounds that spaces and missions coexist within levels; a level has a particular geometric layout and a series of tasks that need to be performed in that space. At a first glance this observation seems obvious, and in fact, the level layouts and the hierarchy of challenges as discussed above seem to acknowledge a similar distinction. At the same time, many designers seem to forget about this distinction: for many games the mapping between the mission and the space is quite direct and their structures often are quite similar, even isomorphic. This section formalizes the structures of missions and spaces in games into a single framework. The advantages of this framework, where missions and spaces are treated as separate but related structures, is that their individual structures, but also their relations, can be investigated with much more clarity. The relation between mission and space is more sophisticated than a superficial survey would suggest. Games might reuse the same space for different missions, as is the case in SYSTEM SHOCK 2 where the player traverses the same areas of a spaceship multiple times. SYSTEM SHOCK 2 shows that the same space can accommodate multiple missions (assuming that the individual mission structures do not resemble each other too closely). Reuse of game space in this way is often economic: the developer does not have to create a new space for every mission in the game. It has gameplay benefits as well. For example, players can use previous knowledge of the space to their advantage, adding to the their sense of agency and the depth of the gameplay.\footnote{Agency is commonly used to describe a player’s power to affect and influence a game world. Agency is a core concept in Janet Murray’s book *Hamlet on the Holodeck* where she defines it as follows: “the satisfying power to take meaningful action and see the results of our decisions and choices” (1997, 126).}

A designer of a level works with both missions and spaces, just novelist work with plots (the sequence of events) and their narration (the way these events are told).\footnote{With plot and narration I refer to the similar notions of “story” and “narration” used by Gérard Genette (cf. 1980, 29). However to avoid confusion between the common sense use of the word “story” and Genette’s more technical use I use replaced it with “plot”. To add to the general confusion, Foster uses the terms “plot” and “story” to indicate a similar distinction, but here “story” means narration, and is completely opposite to Genette’s use of the word (Foster, 1962).} During the design process, the designer is likely to loop between mission constraints and spatial affordances while setting up a delicate balance between
the two that ultimately facilitates the target gameplay. For players, the space is the access point: in space, through play, they realize a mission.

Mikhail Bakhtin’s notion of the ‘chronotope’ provides another interesting parallel for the combined mission and spaces in games. The chronotope refers to the artistic relation between time and space in literature. Bakhtin observes that in literature time and space are “fused into one carefully thought-out, concrete whole” (Bakhtin, 1981, 84), and that their particular artistic intersection is important to the literary form. He also observes that particular chronotopes correlate to particular literary genres. The first genre Bakhtin describes is that of Greek Romance. The plot typically involves two lovers who have to overcome many obstacles in order to marry. This requires many adventures that take place in several far-away countries. The intersection of plot-structure and the journey through hostile environments is surprisingly uniform across many stories from this genre. For games it is the artistic intersection between mission and space that has a similar, large impact on the experience. It also seems that particular configurations of missions and spaces correspond to particular game genres. For example, the mission/space configurations found in action-adventure games are usually far more linear and restricted than the configuration of an open-world with multiple quests that is typically found in computer role-playing games.

Often the complete, combined mission/space structure of a level is set up to generate a narrative experience. For example, the Forest Temple level in THE LEGEND OF ZELDA: TWILIGHT PRINCESS is set up to generate play trajectories that resemble elements of the hero’s journey as described by Vogler (2007). As Link enters the temple, his mission is set up as he finds the first monkey of eight monkeys he needs to liberate. Shortly after this he encounters a large spider guarding the first hub in the level. Defeating this spider grants access to many locations in the first part of the level; the spider acts as a threshold guardian and marks the transition into the dungeon realm of adventure. What follows are many tests and obstacles, during which our hero Link meets many new friends and enemies. Halfway through the level, Link’s confrontation with the monkey king fulfills the same role as “approach to the inmost cave” where the hero confronts and fights the adversary or an important henchman for the first time. The hero escapes with the “elixir” that ultimately helps him to defeat the main adversary in a final climactic confrontation. In this case the elixir is the ‘gale boomerang’: a magic device that acts as a weapon and a key at the same time. With the gale boomerang Link can activate triggers that unlock the second part of the dungeon that hides the final level-boss. Just as the same structure never seems to grow stale for fairy tales and adventure films, it is a recipe that can be found in many of Link’s adventures in this game or any game within the series. It is also found in many Mario games, Nintendo games, and many other games.

More importantly, the Forest Temple level illustrates how both mission and space are used to shape the play experience. The relation between the mission and the space is less straightforward than it may seem. In section 5.6 I will investigate the structure of the Forest Temple in more detail. First, I will discuss how the Mission/Space framework uses two kinds of graphs to represent missions and spaces respectively. These graphs have their own language and structures...
that are discussed in the next two sections.

## 5.4 Mission Graphs

Mission graphs represent the players’ progress towards a goal not by tracking their physical location, but by tracking the tasks they must perform to finish a level. A mission graph is a directed graph that represents a sort of to-do list with each node representing a task that might or must be executed by the players. Nodes can be in one of three different states; the current state of a mission is determined by the respective states of its tasks:

1. A task can be **available**. When a task is available the player can execute it.
2. A task can be **unavailable**. When a task is unavailable the player cannot execute it.
3. A task is **completed** when the player has successfully executed the task.

Edges in the mission graph determine how changes in tasks’ states affect the state of other, subsequent tasks. There are three types of edges between tasks (also see figure 5.6):

1. A **strong requirement** indicates that the completion of the source task (called a strong prerequisite) is a necessary condition for the availability of the target task. A strong requirement is represented as a double arrow directed towards the target task.
2. A **weak requirement** indicates that the state of a target task is changed from unavailable to available when the source task (called a weak prerequisite) is completed or available, unless this is barred by the incompleted state of a strong prerequisite of the target task, or when this is barred by explicit inhibition (see below). A weak requirement is represented as a single arrow directed towards the target task.
3. **Inhibition** indicates that the state of a target task is set to unavailable when at least one the source task (called an inhibitor) is completed, unless the state of the target task is completed. Inhibition is represented as a solid line that ends in an open circle at the side of the target task.
When a task has multiple predecessors, its state is determined as follows:

If it is completed, then its state no longer changes.\(^4\)

Else, if it has inhibitors, then its state is set to unavailable when \textit{at least one} of the inhibitors is completed.

Else, if it has strong prerequisites, then its state is set to available when \textit{all} strong prerequisites are completed.

Else, if it has weak prerequisites, then its state is set to available when \textit{at least one} of the weak prerequisites is completed.

Else, its state is set to unavailable.

There are three classes of tasks in a mission graph (also see figure 5.6):

1. \textit{Regular tasks} are represented as circles. Different colors and letters can be used to represent specific (types of) tasks.

2. An \textit{entrance} is a task that serves as an entry point for a player. A mission graph should have at least one entrance, but might have several entrances representing different starting points for the mission. Entrances cannot have prerequisites and one entrance starts in the available state to represent the player’s point of entry.\(^5\) Entrances are represented as circles that are marked with an arrow head pointing towards it.

3. A \textit{goal} is a task that finishes the mission when it is completed. A goal cannot be a prerequisite or an inhibitor for another tasks. A mission can have multiple goals in which case it is finished when one of these goals is completed. Goals are represented as circles that have a double outline.

\(^4\)Curiously, in games it is fairly uncommon to be able to undo a task, which might make sense when that task inhibits other tasks. Allowing a special undo edge that could revert the completed state of a task would allow mission graphs to incorporate the option of allowing players to overcome problems they created for themselves. This notion, tantalizing as it may be, is left unexplored in this dissertation.

\(^5\)In the case of multiple entrances, what entrance is activated depends on the player’s or the game’s actions prior to that mission.
Static mission graphs cannot display changes to the states of the nodes. However, in a digital version of mission graphs this can be done. For example, figure 5.7 displays a mission graph that does represent the state of the tasks: unavailable tasks have a gray outline and fill, available task have a solid outline and completed tasks are checked. Figure 5.7 represents a mission state where a player has progressed through half the entire mission. A dynamic representation of mission graphs displaying states in this way can be developed into a useful design and analytics tool. This notion is explored further in section 5.8.

At a first glance, when a task has only one predecessor, it seems to make little difference for a mission whether that predecessor is a weak or strong prerequisite. However, as there is no need to execute and complete a task that is a weak prerequisite in order to proceed, players might simply ignore weak requirements. In the case when a fight is a weak prerequisite to finding a key, players might ignore the fight and simply pick up the key. Would the fight be a strong prerequisite for finding the key, for example because the enemy is actually carrying it, players are forced to fight.

Weak prerequisites are quite common in games. For example, the two-dimensional, vertical scrolling shooter game Star Defender launches a sequence of enemies against the player (see figure 5.8). In this type of game, players are usually not required to actually kill those enemies; they can simply ignore them and proceed to the end of the level without firing a single shot.\(^6\) The structure of a game like that might be represented as figure 5.9. In a way, this structure can be replaced by a wide branching mission (see figure 5.10). After all, none of the tasks in the mission is a strong prerequisite for another; the tasks might be executed in any order. The player might even choose to complete the goal immediately. Once players start the mission, all other tasks become available. However, in this case, the order in which players will encounter tasks is lost. This order might be important, as increasing difficulty, pacing and task variation are all considerations that impact mission design. Therefore, the mission structures in figures 5.9 and 5.10 are not the same and specifying the order in which players will (likely) encounter weak prerequisites does matter.

\(^6\)Although this would make the game very, very difficult. In addition, the final enemy of a level usually is a level-boss that must be defeated to finish the mission.
Figure 5.8: Many of the tasks in Star Defender are not required to finish a level.

Figure 5.9: A linear mission with weak requirements.

Figure 5.10: A branching mission with strong requirements.

Figure 5.11: A linear mission with one solution.

Figure 5.12: A linear mission with six solutions.
In addition, there is an important difference in the way strong and weak requirements map to game spaces. It makes sense to place all tasks with a weak prerequisite closely after that prerequisite. That way the order in which the player encounters them is preserved. On the other hand, because strong requirements dictate the order in which players can execute tasks more strictly, a task can be placed much further from a strong prerequisite. One might think of a weak prerequisite as a task players must encounter in order to get to the task that requires it. This difference will be explored in more detail later in this chapter and in chapter 7.

Games with a linear structure of many strong requirements restrict the number of play trajectories considerably. For example, the mission in figure 5.11 only has one solution: players must simply execute all tasks in order to finish it. Although structures like this can be found in games, they offer little variation and replay value. Many games would do well to include weak requirements and branches of strong requirements in order to create more interesting missions. For example, the mission in figure 5.12 does not dictate in what order the player must perform tasks A and B, but both must be completed before the player can proceed. In addition, the player needs to perform at least C or D to be able to complete the mission. Even old and linear games such as Super Mario Bros. often offered alternative paths with different challenges to get to the end of a level. Frequently, their mission structure is more like the one in figure 5.12 than the one in figure 5.11.

### 5.5 Space Graphs

In the Mission/Space framework space is also represented as graphs consisting of nodes and edges. In contrast to mission graphs, space graphs represent space directly, most nodes represent places the player can be in. A space graph has three different types of nodes. Any node in a space graph can be further specified by using colors and letters to indicate different types. The three types of space nodes are (also see figure 5.13):

1. A **place** is the elemental unit of space. What constitutes a place can vary from game to game. In a game with discrete spaces such as a text adventure or a tile-based board game, places correspond with the positions the player can occupy. In games with continuous spaces a place might be a room, a platform or a zone. Usually mechanics help define places in these games: if a particular platform gives the player different options to travel onwards than another platform in a gravity based platform game, then both platforms are considered to be individual places. Places are represented as open circles, whose size can be used to indicate the place's

---

7It might also suggest a second type of strong requirement, one that suggests a prerequisite must be completed and must be immediately precede the task that requires it in space. However, the precise nature of the tasks can be used to determine how far apart a task and its prerequisite can be placed: if the prerequisite is finding a key, they can be placed far apart; when the prerequisite is jumping across a wide gap, it must be placed immediately before the task.
Figure 5.13: The basic nodes and edges of a Space Graph

relative dimension.

2. A lock is a special type of place that is not accessible until the player has activated or acquired the correct key or keys. A lock can be in two states: locked and unlocked. When a lock is locked it is represented as a black circle with a white lock symbol, when it is unlocked it is represented as a white circle with a black outline and a black opened lock symbol.

3. A game element is an item, character, creature or feature that can be found in a particular place. Game elements are represented as small colored shapes that are embedded within the circle of the place they are located in. Game elements often have letters indicating their particular type. Game elements can only be located in places; locks cannot contain game elements. Typical game elements that are frequently found in the examples that follow are entrances, goals, keys and triggers (marked with ‘e’, ‘g’, ‘k’ and ‘t’, respectively). These elements represent game objects that implement mission logic.

Space graphs can have many different relations represented by different types of edges. Some of these depend on a game’s particular implementation. The more common relations include:

1. A path indicates that players can move freely between two places. A path can be traversed in two directions, but as it is usually important to know which place the player is likely to encounter first, a path is represented as an arrow pointing towards the space that is further away from the player’s point of entrance.

2. A valve connects two places or a game element and a place. It can be traversed only in one direction. A valve is represented as a solid arrow

\[\text{In addition, this difference will play an important role when performing transformations on game spaces, certain transformations will allow game spaces to be reorganized in such way that game element or connections are moved towards the starting location, but not the other way round.}\]
with a different arrow head (→) indicating the direction of travel. A valve between two places is only traversed when a player chooses to do so. A valve that starts from a game element automatically transports a player when the player activates the element (voluntarily or involuntarily). This can be used to represent teleporters or traps.

3. A *window* indicates that the player can see a place from another place, but cannot directly travel between these places. A window is not necessarily literally a window: any place the player can see but not immediately reach is considered to be connected by a window. Windows can prepare the player for challenges to come. They can also point the player at areas that are otherwise hidden or difficult to reach and the rewards that getting there might yield (see figure 5.14). In general, players assume that a visible yet inaccessible place does have a path leading to it. Windows are represented as dotted arrows connecting two places. The arrow points towards the place that can be observed.

4. An *unlock* relation connects a game element with a lock, or it connects two game elements. When connected to a lock, the source game element acts as a key to open it. Locks can have multiple keys in which case all keys must be acquired or activated before the player gains access to the lock. An unlock between two game elements indicates that the first element is required to use, obtain or activate the second element, or it might indicate that an element is protected by another element. Unlock relations are represented as dotted arrows pointing towards the lock or the element being activated.

5. A *lock* relation connects a game element with a lock. The game element acts as a trigger to close the lock. Lock relations are represented as dotted lines that end in an open circle at the side of the lock.

As with mission graphs, the basic elements of space graphs can be used to create a number of elementary constructions. These include, but are not

\[\text{Figure 5.14:} \text{ A window in PRINCE OF PERSIA showing a potion that cannot be reached from this direction, prompting the player to paths that are not yet discovered.}\]
A graph that connects two places with paths and valves is called a **pathway**. The pathway includes the two places it connects, and any places in between (see figure 5.15). The shortest possible pathway simply connects two places directly with a single path.

A pathway that does not contain any closed locks or valves is called an **open space** (see figure 5.15). How the open space is constructed does not really matter; even if the place is constructed from a set of connected rooms in a textual adventure it is still considered to be an open space, as long as the player has the ability to travel to all places in the open space freely without having to activate game elements. A linear pathway through which the player can freely travel back and forth is also considered an open space.

A **locked short-cut** can be used as an alternative for a valve. It blocks player’s access from one direction but can be easily unlocked from another direction. Once it is unlocked it will stay open. Locked short-cuts are often implemented with doors that open only from one side (see example a in figure 5.16).

A **hub** is a central place frequently visited by the player from which several separate (usually four or more) pathways start (see example b in figure 5.16). Although some pathways might be initially closed and some pathways might lead back to the hub through a valve. A hub is a good place to locate entrances and ‘save points’ as it can minimize the amount of backtracking after a set-back.

A **set-back** forces players back to a previously visited place as a result of a failed challenge or through the use of traps (see example c in figure 5.16). It will take time and perhaps resources to return to an earlier place. Games with ‘save points’, and in which the player frequently dies, automatically
Figure 5.16: Example illustrating a locked short-cut (a), a hub (b) and a set-back (c).

have set-backs. Set-backs are also frequent in platform games where failing crucial jumps can force the players to lose much altitude that might not be easily regained (also see Compton & Mateas, 2006).

- In contrast to the set-back, a force-ahead pushes players into previously unexplored areas for which they might be ill-prepared or lack the proper equipment. In general, chances of player death and failure increase, but the player should be allowed a chance to return to the previous stage, leading to a survival challenge. The old dungeon crawler Eye of the Beholder II includes trapdoors that drops the player into lower, more difficult part of the dungeon from which the player would struggle to return. A force-ahead can be realized with a valve to a new and very dangerous place.

5.6 Level Analysis: The Forest Temple

The differences between mission and space, and their relation, become more clear from the analysis of the ‘Forest Temple’ level in The Legend of Zelda: Twilight Princess (see figure 5.17). The The Legend of Zelda series of games are well known for their quality game and level design and this game is no exception. In this level, the player, controlling the game’s main character Link, sets out to rescue monkeys from an evil presence that has infested an old temple in the forest. The mission consists of the player freeing a total of eight monkeys, the defeat of the mini-boss (the misguided monkey king Ook), finding and mastering of the ‘gale boomerang’ before finally defeating the level-boss (the ‘Twilit Parasite Diabara’). Figure 5.18 displays the forest temple level map as it is published in the official game guide. Figure 5.19 presents a mission graph and a space graph of the level. In order to reach the goal, Link needs to confront the level-boss in a final fight. In order to get to that fight, Link needs to find a key and he needs to rescue four monkeys, for which he needs the gale boomerang, for which he needs to defeat the monkey king, etcetera. Some tasks can be executed in different order: it does not really matter in what order Link liberates the
Figure 5.17: Confronting the misguided monkey king Ook to obtain the gale boomerang in The Legend of Zelda: Twilight Princess.

Figure 5.18: The map of the Forest Temple from the official game guide (Hodgson & Stratton, 2006, 68).
Figure 5.19: Mission graph and space graph for the “Forest Temple” level from The Legend of Zelda: Twilight Princess. The gray lines connect the corresponding fights in both graphs.
monkeys two, three four. Other tasks are optional, but lead to useful rewards.

The mission structure for the Forest Temple level has a few striking features. One is the bottleneck formed by the fight with the mini-boss and the retrieval of the boomerang halfway through the structure and the two sections of relative nonlinearity before and after the bottleneck. This structure corresponds directly to the learning curve of the level in which the player needs to train with the new weapon before using it to defeat the level-boss. It is a structure that is frequently found in Nintendo games.

The game space features a hub-and-spoke layout (see section 5.1) that supports the parallel tasks of the mission structure. From the central hall (where the first guardian is fought) the player can go into three directions. The pathway that leads to the right, quickly branches into three more pathways. Three pathways lead to captured monkeys and one to the mini-boss. The last pathway is only open to Link after he has freed the first four monkeys. After the player has retrieved the boomerang, additional spaces in the first hub-and-spoke structure and a new hub can be reached. This structure is not the only possible space to accommodate the mission. It is not very hard to imagine different types of spaces that accommodate the same mission structure. Even a linear layout, in which the rooms are aligned in a long row with different tasks in each of them would be a possibility.

When playing the game, a trajectory through the space is generated by the player. This trajectory can be represented as a linear string of visited nodes. Trajectories that lead to the end goal of the level might be called solutions. The individual trajectory is constrained by the structure of the map. The player has to start at the entrance and the only way to progress is to find the first monkey. From there on an infinite set of trajectories becomes possible (infinite, because the space allows the player to travel back to earlier locations). Yet, this does not mean that all combinations are possible. Players must first defeat the mini-boss before retrieving the boomerang, and they must have retrieved the boomerang before they can reach the master key. The layout of space constrains the possible play-trajectories and solutions.

From this analysis it should become clear that both mission and space have their own individual structure. There is some natural affinity between the hub-and-spoke space and the required parallel branches in the mission, but the two are not directly linked. As mentioned above the same mission structure could be mapped to a linear structure in space, whereas a linear mission could also be mapped to a hub-and-spoke layout (one where the order of the spokes to be visited is fixed).

The gale boomerang itself is a good example of the lock and key mechanisms typical for the series and indeed this is used in many action-adventure games (cf. Ashmore & Nietsche, 2007). Lock and key mechanisms are one way to translate strong prerequisites in a mission into spatial constructions that enforce that relation between tasks. The boomerang is both a weapon and a key that can be used in different ways. It has the capability to activate switches operated by wind. Link needs to operate these switches to control a few turning bridges to give him access to new areas. In order to get to the master key that unlocks
the door to the final room with the level-boss, he needs to use the boomerang to activate four switches in the correct order. At the same time the boomerang can be used to collect distant objects (it has the power to pick up small items and creatures), and can be used as a weapon. This allows the designer to place elements of the second half of the mission (after the mini-boss has been defeated) in the same space that is used for the first half of the mission. This means players will initially run into obstacles they cannot overcome until they have found the right ‘key’.

5.7 Mission-Space Morphology

The structuralist tradition in narratology understands the part of the art of a well-told story in terms of the structural relations between the different elements in plot and narration. In a well-told story plot and narration are rarely the same. Narration speeds up through, or omits uninteresting events. Narrations that use flashbacks or flash-forwards present the events in the plot in a different order than they have occurred. A common trick is to start a narration “in medias res”, where the narration starts in the middle of or just the main action, only to step back to relate the events that led up to the starting scene. Differences between plot and narration are powerful narrative devices that are exploited by master storytellers in any medium. What is more, a plot is not necessarily linear, it might contain events that occurred simultaneously; when the story is narrated in a linear medium (as most traditional media for storytelling are), the storyteller must use these devices to be able to tell the story at all. Traditional structures and templates exist that have proven their value in the past and are still used in books and films alike. From the structuralist analysis of Russian fairy tales by Propp (1968) and theory of the monomyth of Campbell (1947) to more modern interpretations used in contemporary cinema by Vogler (2007). Many of these structures facilitate anticipation and involvement of the reader with the narrated events. Differences between plot and narration are employed to foreshadow, to change perspectives, to shock and to educate. The possibilities are endless.

Similar, commonly used structures do exist for game spaces and missions. In fact, the structures in figure 5.1 are examples of this type of structures, even though mission and space are somewhat tangled up in them. Currently much level design depends on a direct mapping between mission and space: all too often, mission and space are isomorphic structures. A strong indicator for this dependency is the popularity of the quest in many adventure games, a genre that typically features strongly articulated spaces and missions. In the trope of the quest, in which the journey of the hero indicates personal growth, mission and space are isomorphic; they share the same structure. In many ways this is very convenient. Dungeons & Dragons designer Dave Arneson is attributed to have once remarked that Dungeons & Dragons levels (or dungeons) are intentionally designed as flowcharts. It could be argued that a certain level of isomorphism between mission and space guarantees a smooth player experience.

10 From personal correspondence with David Ethan Kennerly (2009).
In its most extreme case where a player is both railroaded through space and mission the play experience can be tailored to that one single trajectory. *Half-Life* and *Half-Life 2* have refined this strategy to perfection, and to great commercial and critical success (see section 1.6). Strong isomorphism between space and mission allows the use of many spatial metaphors or other spatial narrative devices. The symmetrical pathways mentioned above, could for example be used to suggest a similarity between the tasks that must be completed along those pathways.

On the other hand, in the ‘Forest Temple’ level, mission and space are not isomorphic (see figure 5.19 above). During the first half of the level, the mission with its parallel chains maps fairly directly to the hub-and-spoke layout of the space. But once the gale boomerang is obtained the player has to traverse back through the dungeon to gain access to previously inaccessible places. Would mission and space be more isomorphic, Link would have to travel further into a new section of the dungeon. Of course, as conventional level design wisdom dictates, there is little point in using a lock and key system where Link encounters all the keys first. In general, it is better to have the player find the lock before the key in this way for three reasons.

1. When keys are encountered first, players will simply be forced to collect everything they encounter without discrimination, which makes rather simplistic gameplay.

2. With obstacles and items that act as locks and keys but are represented as something else, it is easier to recognize the key if players know what the lock is; players then usually realize where they can proceed; they will actively formulate the intention to return to the lock.

3. When players can negotiate obstacles they were unable to get past earlier, they will experience progress and accomplishment. Thus, forcing Link go back to previously visited places foregrounds his growth as a character. Where there were obstacles he could first not overcome, he now can conquer that space. It sets up a natural, clever contrast between Link’s state before and after the defeat of the mini-boss and marks his progress along the hero’s journey.

There is no reason why in a game mission and space should be isomorphic. In fact, just as stories benefit from different morphologies for plot and narration, so do games benefit from having different morphologies for mission and space.\footnote{On a side note, I once experimented with an ‘in medias res’ structure for a tabletop role-playing session, where I had players play through the events that lead up to the opening scene they played through earlier. ‘In medias res’ is also used in *Prince of Persia: Sands of Time* where most of the game is framed as a long flashback. In both cases the structure translates well to games and actually contributes positively to the gameplay experience.}

### 5.8 A Software Tool for Level Design

As is the case for the Machinations framework, the Mission/Space framework is formal enough to be implemented as an automated design tool. A tool like
that, with the name *Ludoscope*, was created as part of this research. Ludoscope allows a designer to draw mission and space graphs efficiently. In fact, the mission and space graphs that feature in this chapter were created with Ludoscope. However, Ludoscope is more than just a drawing aid. It also implements the logic that governs mission and spaces. As a result, it can be used to simulate levels at an early stage and help developers identify design flaws.

The implementation of mission graphs in Ludoscope allows designers to hook up a network of tasks. Designers can indicate which tasks are entrances and which tasks are goals. Based on that information, it determines which tasks need never be encountered and marks them as such. By selecting a task, the designer can immediately see what other tasks can be reached from that point in the graph, these tasks are given a green outline. When a selected task inhibits another task it displays the inhibited task with a red outline. Other tasks that have become unreachable because of this are also displayed with a red outline (see figure 5.20). This allows designers to select tasks one at a time in simulation of a player completing tasks in the game and checking if the mission still can be completed. Ludoscope can easily be extended to include more checks for inconsistencies or warn the designer for potential deadlocks. In addition many types of heuristics for games might be implemented to assist designers, such as space syntax measures described by Paul Martin (2011).

The implementation of space graphs in Ludoscope allows designers to do something similar with spaces representing missions as well. By selecting a place in a space graph, all places that can be reached from that location are rendered in with a green outline. In addition, designers can select game elements to unlock (or lock) locks. Again, this allows them to simulate a player’s progress through a level (see figure 5.21).

Ludoscope is still work in progress, and as will be discussed in Chapter 7 has much more functionality than drawing Mission and Space Graphs only. In particular, Ludoscope initially evolved from prototypes for procedural level generation tools, and it contains support to represent and involve mechanics through Machination diagrams as well. This latter aspect allows designers to zoom in on the relation between their game’s mechanics and level design, which, as will be argued in the next chapter, is a critical aspect of designing games where

![Figure 5.20: Selecting task B in Ludoscope reveals that task A is inhibited making it impossible to reach goal C. Goal D can still be reached.](attachment:image.png)
Figure 5.21: Simulating the Forest Temple level. The light blue outline indicates selected places and activated game elements. By selecting the place containing the entrance and various keys, player progress can be simulated. In this case, the simulated player is in the early stages of the level having made some progress in the lower left and right corners.
structures of emergence and progression are tightly integrated.

5.9 Conclusions

As is the case with game mechanics, level design benefits from a more structural approach. A game level is a complicated construction consisting of many different parts. Different models and perspectives can help level designers to keep their focus and to structure their labor. To this end, this chapter introduced the Mission/Space framework. This framework includes specialized graphs to represent missions and spaces separately.

Mission graphs focus on the tasks and challenges that players need to complete in order to finish a level. There are three types of relations between the different tasks in a mission: weak requirements, strong requirements and inhibition. Mission structure foregrounds the interdependencies between tasks and can be used to identify potential deadlocks or other inconsistencies, as well as a balanced learning curve and training structure. Space graphs represent the topographical structure of the level. They are indicative of how different places are connected, a level’s flow, pacing, and what game elements can be used to unlock new areas.

Mission and space are related, creating spaces to accommodate missions or creating missions to suit a particular space is an important aspect of level design. Although, sometimes games have levels where missions and spaces are almost isomorphic, the structures of mission and space are independent. A detailed analysis of the Forest Temple level in The Legend of Zelda: Twilight Princess showed that games benefit from more sophisticated relation between missions and spaces.

In order to aid level designers, the Mission/Space framework can be used to create level design tools that help designers create complex but consistent levels. The Ludoscope prototype implements some of these ideas. It allows designers to create mission and space graphs and subsequently simulate players progress through them.

The Mission/Space framework provides a solid structure that can be used as a stepping stone in the development of further theory for game design. In the next chapter, the Machinations and Mission/Space framework will be used to explore how structures of emergence and progression can be integrated. In Chapter 7 both frameworks are leveraged to create a framework and prototype that allows the automation of certain task in order to support the creative process of designing games.