Engineering emergence: applied theory for game design
Dormans, J.

Link to publication

Citation for published version (APA):
Dormans, J. (2012). Engineering emergence: applied theory for game design Amsterdam: Creative Commons

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
The challenge for game designers who want to create rich, open game worlds and tell interesting stories at the same time, is to move beyond the constraints of unicursal corridors or multicursal hub structures while keeping the player’s attention on a storyline. And it is no easy task.

Espen J. Aarseth (2005, 11)

6 Integrating Progression and Emergence

This chapter focuses on the relation between mechanics and levels, which were the subjects of the previous two chapters. The Machinations framework and the Mission/Space framework formalize different perspectives on these two aspects of games. Machinations diagrams help visualize and understand a game’s internal economy. It focuses on the mechanics that govern the economy and contribute to emergence in games. The Mission/Space framework, on the other hand, focuses on the construction of levels, and the mechanics that control players progress through a game. These two aspects of game design: mechanics and emergence, versus level design and progression, are often understood as two opposing or conflicting ways of generating gameplay. It is the goal of this chapter to leverage both frameworks in order to integrate structures of emergence and progression more closely and bridge the gap between game mechanics and level design. Ultimately, a close integration between emergence and progression would lead to games where progression through levels, and perhaps stories, emerges from the rules.

In section 6.1 I will discuss how professional designers typically create mechanics before they start focus on level design. In the two sections that follow, I will explore how the relation between mechanics and levels is typically approached in games that traditionally rely on structures of progression, and sandbox games that allow players to build and structure the internal economy themselves, respectively.

Integrating levels and mechanics is not easy, and the number of games that have successfully done so in the past is low. In section 6.4 I will discuss the limitations that have been encountered in designing game levels that display more dynamic and adaptive behavior. The problem often resides in a conceptual mismatch between relatively crude mechanisms for level and story on the one hand and quite sophisticated and expressive mechanisms for physics simulations and/or economy on the other. In section 6.5 I will explore how feedback mechanisms might be applied to mechanics to control progression. This explo-

1This chapter is an extended version of a paper presented at the DiGRA Conference in 2011 (Dormans, 2011b).
ration will lead to two new techniques to create progression through emergence: 1) feedback mechanisms for lock and keys, and 2) modeling progress as an economic resource. These techniques are discussed and illustrated in the final two sections of this chapter.

6.1 From Toys to Playgrounds

In general, mechanics are designed before levels, and this means that in most games, level design is dependent on the game mechanics. Particular mechanics often serve as a starting point for levels in a game: a level might introduce or focus on particular mechanics. For this reason, it is usually not possible to finalize a game’s level design before finalizing the mechanics. After all, the challenge posed by a platform carefully placed at a distance that is just less than the maximum jump distance completely changes when the jumping distance is increased. Worse, what was intended as an impossible jump, might change into a possible jump or the other way round. It is also one of the reasons why the MDA-framework stresses that game designers look at games starting with rules and mechanics (see section 3.2).

Kyle Gabbler summarizes the dependency of level design on game mechanics as one of seven tips in his video keynote for the first Global Game Jam in 2009 (Gabler, 2009). He advises game designers to “make the toy first”: he urges the participants not to create any art assets until the basic mechanics of the game are in place. These basic mechanics should already deliver an interaction that is fun, even without goals or well designed levels. For example, if one sets out to build a racing game, the car should be built first, it is only when the car is fun to drive around, that one should start work on graphics and on tracks. From my own experience as a Global Game Jam participant in 2009, 2010 and 2011 I can testify that this tip is very useful.

Gabler’s tips are not restricted to game jams and games built under extreme time constraints only: they are valuable for any game development project. Games are developed by creating many prototypes during short iterative steps. The earlier prototypes tend to resemble Gabler’s ‘toys’: they often implement the basic mechanics and interactions, without spending much attention to goals or levels. In most cases, the levels that exist at this stage, serve as testing grounds to illustrate the basic game mechanics. Designers typically throw in as many game objects as they can for play-testers to play around with. The goal of these prototypes usually is to explore the limits of the mechanics, and to look for opportunities to create interesting interactions, or explore how game objects might be combined into interesting puzzles and challenges. It is only when the designers are convinced that the basic mechanics are solid that the virtual space is organized into a level set up to deliver a concise and well-structured experience; when the individual toys are organized into a playground.

The emergent simulation game Spore was developed using a large number of small prototypes, and serves as an excellent illustration of this process. What

---

2The Global Game Jam is an annual event in which teams at hundreds of locations all over the world enter a competition to build a game based on a set theme within 48 hours.
is unique for a large commercial game, such as Spore, is that many of these prototypes are available for download.\(^3\) These prototypes were built for various reasons. Some were experiments with the procedural techniques used to create terrains, worlds and even entire galaxies. While others are gameplay experiments designed to try out various rules and behaviors. One of these, SPUG (figure 6.1), is a prototype for the “creature stage” in the published game (figure 6.2). According to accompanying text on the website in this prototype “no limitations are placed on leveling up or cheating stats. This tool was intended to give designers the opportunity to explore different economies for the creature game, so limitations on power-ups and level ups are self-imposed.”\(^4\) Thus it was used to experiment with the same type of game mechanics that is the focus of the Machinations framework.

The design strategy that gives mechanics dominance over level design has a number of implications. It provides certain types of games with a natural structure for progression, as will be explored in the next section. Other games, that focus more on dynamic gameplay created by sophisticated mechanics (as explored in section 6.3), use levels only to set some rough boundaries that provide some color and perhaps some constraints on the emergent gameplay.

### 6.2 Progression through Structured Learning Curve

One of the many considerations of designing levels, is to train the player in the required gameplay skills necessary to complete the game. In general, players do not want to read manuals in order to play a game; they expect that learning the mechanics is a natural result of playing. This means that levels need to be structured in such ways that players actually learn the game while playing it. This type of structure fits well with games of progression that frequently model their structure after the hero’s journey. A sense of growth and accomplishment drives these games and sets up a particular relation between game mechanics.

---

\(^3\)See [http://eu.spore.com/extra/?id=10488](http://eu.spore.com/extra/?id=10488) (last visited July 17, 2011).

and level design that is common among action-adventure games, first-person shooters and genres that typically include structures of progression.

Daniel Cook's skill atoms constitute one of the most concrete theoretical perspectives on this aspect of level design (Cook, 2007). He analyses the individual steps a player goes through in learning a new game skill, and the way individual skills are hooked up into chains. Once the design team has decided on the final mechanics to be included in the game, levels can be structured in such way that the player is taught these mechanics. The most straightforward approach is to spread out the chain of skills over the level and to organize the level accordingly. In this case, the chain of skills is integrated into the level’s mission structure, which is then related to the level’s spatial layout in a similar way as described in the previous chapter. However, levels are not there to teach the player the required skills only; there is usually more to a level than just a tutorial. Levels are also structured to facilitate exploration. Once the player has learned the basics of playing a particular game, levels provide the player with opportunities to display their mastery. During this stage, the mechanics become a means towards the goal of exploring the level or completing an interactive story.

The way the player is taught to use new items and strategies through level design also resembles the learning stages found in martial arts training (Kohler, 2005; Dormans, 2005). This structure plays an important role in the level’s mission and is a common feature in many action games, in particular action games produced by Nintendo. It is an excellent technique to learn the player the required actions to finish the game. It is an easily recognizable pattern of a game’s mission, although to use it to the best effects, the space needs to be designed to support this structure. In both action games and martial arts, students need to master a whole vocabulary of different moves and combinations of moves. In martial arts learning progresses through four stages in which techniques are learned, practiced, combined and tested. These stages, called kihon, kihon-kata, kata and kumite, are all present in the Forest Temple level of The Legend of Zelda: Twilight Princess (see section 5.6). The kihon stage, in which a player learns an individual technique in isolation and relative safety, is present in both the earlier tasks involving the ‘bomblings’ and the tasks for which the player needs the ‘gale boomerang’. These tasks are repeated a few times (kihon-kata) before the player is taught how to use the boomerang to pick up the bomblings to deliver them over greater distances before he can reach the final two monkeys (kata). Finally the player needs this combined technique in order to defeat the level-boss (kumite). The order and logic of these stages is dictated by the mission structure. The player cannot encounter the level-boss before she successfully demonstrated the techniques needed to defeat it (see figure 6.3).

This structure of introducing game mechanics gradually and having them act as locks and keys can be found in a very pure form in certain smaller independent

---

5Although I like to point out that the development process for games is highly iterative, and that work on level design is often started before all mechanics are complete. It is not uncommon that mechanics are changed because of insights gained during the design of levels. However, the design process usually starts with mechanics, and mechanics are usually made definitive before all levels are designed.
games. *Knytt Stories* and *Robot Wants Kitty* are good examples of such games. Both of these games are platform games where the player’s goal is to reach a particular location (even though these games’ story might frame it a little bit differently). Both basically consist of one large level where the player gathers a number of power-ups that act as locks and keys. But also the challenges the player meets get progressively more difficult. Where, for example, the double jumping ability allows the player to jump longer distances in both games, the gaps the player needs to jump across do get wider, and the penalty for failing a jump increases from, having to replay a little part of the level in order to try again, to dying and/or replaying longer parts of the level.\[^6\] Table 6.1 lists the power-ups in *Knytt Stories* in the same order they are encountered. Note that some of these power-ups combine in even more powerful combinations. For example, once players have found both the double jumping and wall climbing abilities, they can use a double-jump after jumping from a wall. Likewise when players have the double jump and the high jump, they can use both to jump even further.

Larger games, such as *The Legend of Zelda: Twilight Princess*, take more time to introduce their mechanics. As we have seen in the detailed analyses of this game’s Forest Temple level in section 5.6, only two important mechanics/power-ups were introduced in this level: the ‘gale boomerang’ and the use of ‘bombling’ creatures. *The Legend of Zelda: Twilight Princess* typically takes more time to train the player in the use of these mechanics, and explores more possible combinations of the two, allowing it to design an entire level around these two mechanics. Although I estimate the average time a player

\[^6\]A double jump is a common mechanism found in many platform games. A double jump allows player controlled characters to jump once while in mid-air, effectively allowing the player character to jump twice. It can be used to jump higher or further and emphasizes the timing involved in performing jumps.
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Associated lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>The player can now move faster.</td>
<td>Jump across wider gaps, it also becomes easier to jump past enemy creatures.</td>
</tr>
<tr>
<td>Climb</td>
<td>The player can now climb and jump from vertical surfaces.</td>
<td>Tall vertical structures too high to jump.</td>
</tr>
<tr>
<td>Detector</td>
<td>The player glows red when certain types of enemies are close. The player grows green when a secret passage is near.</td>
<td>None, this power up is optional (but very useful).</td>
</tr>
<tr>
<td>High Jump</td>
<td>The player can jump higher.</td>
<td>Wider gaps; creatures that jump low when the player jumps; jump and climb to higher locations.</td>
</tr>
<tr>
<td>Double Jump</td>
<td>The player can jump a second time in mid air before landing.</td>
<td>Wider gaps; creatures that jump low when the player jumps; jump and climb to higher locations.</td>
</tr>
<tr>
<td>Eye</td>
<td>The player can now see invisible creatures and platforms, which curiously did not affect the player before.</td>
<td>Previously invisible platforms allow the player to get to places that were impossible to get to earlier.</td>
</tr>
<tr>
<td>Umbrella</td>
<td>The player can use the umbrella to glide and even float up in areas that have upward drift. The Umbrella also protects from enemies that shoot 'bullets' from above.</td>
<td>Upward drift, extreme long jump (especially when combined with a high, double jump).</td>
</tr>
<tr>
<td>Hologram</td>
<td>Player leaves a holographic projection fooling enemy creatures.</td>
<td>Creatures that otherwise block your passage when the player gets close.</td>
</tr>
</tbody>
</table>

Table 6.1: The power-ups of Knytt Stories in the order they are encountered and their associated locks.
Figure 6.4: Game elements as they are introduced in *The Legend of Zelda: Twilight Princess*. The data for this figure was collected from playing the game and consulting a printed game guide (Hodgson & Stratton, 2006).
will spend in the forest temple level is roughly half of the time spent to complete *Knytt Stories*, *The Legend of Zelda: Twilight Princess* still takes more time to introduce each power-up and explore its potential. As players explore more levels in *The Legend of Zelda: Twilight Princess* more and more mechanics are introduced, although every level focuses on only a few new mechanics and a few reused mechanics. Figure 6.4 provides an overview of this game’s most important mechanics, when they are introduced and in what levels they are reused. Mechanics that only act as locks and keys (such as ‘fused shadows’ and ‘mirror shards’) without being weapons or affecting the gameplay are omitted in this figure. Many of the mechanics combine in interesting ways. For example, wearing the ‘Zora armor’ allows the player to breathe underwater, when also wearing the ‘iron boots’ will make the player sink allowing the exploration of deep water caverns. In addition, the enemies and traps players encounter require an increasing mastery of the game’s mechanics and their combinations to defeat or avoid them.

What both *Knytt Stories* and *The Legend of Zelda: Twilight Princess* do well, is to integrate the mechanics that control the progression through the game world with the emergent mechanics that determine the core gameplay. This is an effective and proven way to combine emergence and progression in a game. This is testified by the success of the Zelda games and the frequency with which mechanics or power-ups are used in this way. However, whether the two modes are truly integrated remains doubtful. Progression in these games still relies on a fairly low number of game state changes, and needs to be planned in detail to work. Over the past few decades, game designers have learned to forge the two into pretty smooth game experiences. But it has not led to a true synthesis of emergent and progressive gameplay.

### 6.3 Economy Building Games

Strategy games and management simulation games usually follow a different design strategy to integrate their mechanics into level design. In stead of creating levels to dictate a structure of progression, these games allow players to build and shape an emergent, internal economy within the constraints set up by the level’s space or mission. This design strategy can be found in computer games as *Civilization*, *Caesar III* or *StarCraft*, but also for modern, management simulation board games such as *Puerto Rico*, *Caylus* or *Agricola*. In these games an internal economy is built during play. Game spaces constrain the way the economy might be built, but rarely accommodate a mission in the sense that was discussed earlier. Instead of traveling towards a particular goal, the goal of the game is to build up an effective game space. Missions, insofar they exist in these games, often constitute of the single task to build an economy according to some sort of specification. These economies usually start out quite simple with the production of basic resources, but tend to get more complicated quickly. For example in *Civilization* players build cities to produce food, wealth, knowledge, buildings and units. The location of the city affects the production rates: building a city on fertile grasslands will increase the food production, rivers will
Chapter 6 | Integrating Progression and Emergence

143

Figure 6.5: In the game CAESAR III players build cities in the Roman era.

increase trade and wealth, while hills and mountains offer the opportunity to build mines to increase production of buildings and units. Players need to find locations that best suit their needs. A player that is going for a strong military will need more production, while building close to rivers can speed up trade, wealth and scientific advancements. There are both long term and short term considerations: cities that grow fast will eventually output more resources than cities that are located close to interesting resources but far from fertile soil. The default mode of playing CIVILIZATION generates new maps for every game the player starts. Players will thus have to make the most out of the land they have discovered.

Modeling a game like CIVILIZATION with the use of the Machinations framework presents us with a problem. Although many of the individual mechanics can be easily captured using machinations diagrams, different sessions require different diagrams as players effectively hook up game elements differently every time. By building and changing elements in the game’s space, players also set up their own game economy. These economies usually start out quite simple with the production of basic resources, but they tend to quickly get more complicated. For example, in the Roman city simulation game CAESAR III (see figure 6.5) players build cities in the Roman era. Players need to build infrastructure for traffic and water, buildings to produce food and other basic resources. To build up the city’s economy the player needs residences, workshops, markets and warehouses. Citizens will demand temples, schools and theaters, and at the same time the player must provide security against different types of threats by building prefectures, city walls and hospitals. Finally the player must train soldiers to protect the city from invading barbarians. The cities economy is dominated by a
Figure 6.6: The basic economic relations between the elements in *Caesar III*.

multitude of resources. Farms produce wheat, fruits, or olives, clay pits produce clay that can be converted into pottery in special workshops. Other workshops convert olives into oil, or metal into arms. The residences the players build are in constant need of these and other goods. The better the player can supply these residences the wealthier their inhabitants become, and that will in turn improve tax income needed to build more farms, workshops and residences or pay for other needs such as prefectures to fight crime and fire or military structure to protect the city from invading barbarians. In the meantime players will need to build granaries, markets and warehouses to distribute all these needs effectively over the growing city. Figure 6.6 represents the basic economic relations between some of these elements. The consumption of trade goods in residences triggers the production of wealth. More wealth has a positive effect on the amount of money generated through taxes. At the same time, wealth drains quickly creating an ever increasing need to supply residences with high quality trade goods. In the game the actual connections between all these elements are flexible: a farm might deliver its crops to a granary, warehouse or workshop depending on the needs and the distances to these locations (see figure 6.7). The challenge of *Caesar III* is to utilize space effectively and build a smooth running economy. Players gradually build this economy as they see fit, but it will invariably be dominated by the positive feedback loop that involves production, consumption by citizens and tax income. This positive feedback is balanced by the negative feedback provided by the dynamic friction built into the citizens mechanism (see figure 6.8). The more effective players are in utilizing space and building up their city, the more effective their economic engine will run.\(^7\)

Games where players build the game economy in this way, very clearly fall into

\(^7\) *Caesar III* includes a few more elements that I have left out for the sake of discussion. In the real game players also need to manage a number of hazards such as crime and fire by building special buildings to counter these effects. They serve to complicate the production mechanisms further. Apart from nutrition and wealth the citizens also demand entertainment, culture, education and religion which are produced and consumed in similar ways. Finally, in most levels of *Caesar III* players also need to deal with demands of the Roman emperor and fight invading barbarians, they act as extra (periodic) friction to the economic engine.
Figure 6.7: A possible spatial configuration for some elements of CAESAR III.

Figure 6.8: The dominant economic structure of CAESAR III.
Jesper Juul’s category of games of emergence: it is almost impossible to write a walkthrough for Civilization or Caesar III, but there are plenty of strategy guides available online. Still, playing these games does offer something of an experience of progression. Caesar III, for example, has a series of scenarios, each with its own particular challenges and goals, and within each scenario there are a number of scripted events. But even without these events, the process of building a city goes through a number of stages, from the initial planning stage, when players still have much money to build anything they might need, to the managing of a crisis or the city’s defense, to the tweaking and fine-tuning of the city’s economy in order to reach very hard economic goals during the end stages of the later levels. Caesar III, as many other games of emergence, has its own rhythm and progression that partly emerges from its dynamic game economy, and partly from the scripted events that are unique for every scenario.

To design mechanics and to plan scenarios for these games is not easy. It requires a thorough understanding of the game’s potential economy. Most of the game elements the player can add to the game will somehow expand the game’s economy, but there are only so many ways these elements can combine. Most of these games will have a dominant economic structure as the one presented in figure 6.8: these structures are the most effective way of setting up the economy and players will eventually veer towards building similar structures. In some cases, a game might contain a few alternative dominant structures. Most of the time players will be able to familiarize themselves with only a few elements at a time, learning the particularities of the game’s potential economy over multiple sessions. This requires that players should be able to combine individual elements in different ways, and preferably in such ways that these combinations offer different strategic options.

At the same time, to keep players on their toes, a few scripted events, or a rhythm that is dictated by a mechanism that is not very emergent at all, can be used to structure scenarios and can give the game clear goals or closure. Caesar III’s scripted events are one example of these, but similar mechanics can be found in many other games, including board games (where one perhaps least expected such mechanics). In the board game Caylus, for example, players compete for the favor of king Charlemagne by building him a castle and surrounding town. The game progresses through three stages after which a part of the castle is finished and the players are rewarded for their contribution (or punished for the lack thereof). Every round, the timeline that keeps track of the progress is advanced one or two steps based on the players’ actions. Once the timeline is advanced at least six steps, the first stage is completed; after at least twelve steps the second stage is completed and after at least eighteen steps the third stage is completed and the game is finished. This timeline drives the game towards a conclusion, but does not constitute what I would call a dynamic mechanism.

6.4 A Mismatch Between Mission and Space

Although the design strategies to integrate mechanics and progression as described in the previous two sections are successful in their own way, it seems
that the development of design strategies that integrate the two more closely has hit a barrier. This barrier seems to come from an imbalance in the advances that have been made with creating sophisticated mechanics and game spaces on the one hand, and the lack thereof in the development of missions and interactive stories (Wardrip-Fruin, 2009). This mismatch between the mechanics of mission and space has held back further integration of emergence and progression.

Over the past few decades the game development community has accumulated much experience with creating compelling game spaces with interesting rules. From the early limited spaces from the seventies and eighties to the vast virtual areas found in modern games, game spaces have grown into highly detailed constructions with near analogue qualities. Traversing the space of a contemporary game is no trivial task, especially for those games that involve movement in their core mechanics, as is the case with most action games. But also for strategic games this evolution has been fast. One needs to simply compare the open free world of STARCRAFT II to the tile based combat found in CIVILIZATION or indeed classic boardgames such as CHESS, to appreciate the strategic depth allowed by freely positioned units and more continuous terrain features. Seeing these huge strides in the development of game spaces towards structures with a high granularity, I agree with Noah Wardrip-Fruin who observes that it is curious that game stories and quests have not grown as much; game missions usually work with a very limited set of possible states, all of which are known before play (Wardrip-Fruin, 2009).

One cause for the mismatch in granularity between mission and space, Noah Wardrip-Fruin points out, is the popular quest logic and dialog trees common to most games. The player’s progression through a mission is simply tracked by setting up a few bottlenecks or gates to act as milestones in a story. Once the player fulfilled the task associated with a milestone, the story is advanced. The implementation is as simple as keeping track of a few simple Boolean story flags that control the visible entries for the in-game journal that records the game story (Wardrip-Fruin, 2009).

For example, the story-driven first-person shooter game Deus Ex has detailed rules to simulate combat in urban environments, where players can use different strategies to overcome obstacles. An important aspect of the gameplay is that players can choose between various upgrades of their character, allowing them to specialize in stealth or direct combat, hacking computers or bypassing locks. Most of the levels are set up to accommodate different playing styles. It is even possible to complete the game without using lethal force against human opponents. Deus Ex accommodates a fairly continuous space of viable strategies and tactics to overcome obstacles. On the other hand, the options presented to players during narrative cutscenes and dialogs are always presented as discrete choices between a handful of options (see figure 6.9), which have little effect on the development of the narrative events. The progression of the levels is always the same, sometimes dialogs and cutscenes are different, but the overall outcome is usually the same. Only during the very final level, the actions of the player will determine which one of three endings is selected.

The common implementation of dialog trees can further serve as an example
of the problems faced by designers of interactive missions and stories in games. Dialogs feature in many games, and while certain games do not even bother to make their dialog interactive, those that do often resort to using dialog trees. A dialog tree offers players a few optional lines to advance the dialog. Reaching certain leaves in a dialog tree might change the story state. Many dialog trees are not really trees, but are more akin to directed graphs as often different paths through the tree will take players to the same node in the dialog. One problem commonly associated with these tree-like structures is that they quickly become inefficient and overly complex; the number of options that need to be created is much larger than the average player will ever see, and without proper editing tools the writer of a dialog tree might quickly lose track of all the options. Worse, dialog trees do a rather poor job of really creating the illusion of freedom or agency. With only a few options available at a time, chances are that players will feel constrained in their options (Wardrip-Fruin, 2009, 56). In all likelihood players will recognize the tree-like structure, and it is not uncommon for them to traverse the entire tree in order to explore all possible gameplay options, which mostly is a trivial yet tedious task. In short, at the micro-level of the dialog, these tree-like structures often constitute poor gameplay (Dormans, 2006c). Still, at the macro-level of mission or story, they are quite common.

It is probably one reason why many story-driven games do not offer any variation in the way the story unfolds. Games such as \textit{Half-Life 2}, \textit{The Legend of Zelda: Twilight Princess}, or \textit{StarCraft}, all use their levels and challenges as simple bottlenecks that players must pass in order to advance the story. Failure to overcome an obstacle simply means that players will have to try again. Sometimes there is little room for variation as is the case with \textit{StarCraft II}, but that rarely goes beyond the option to choose in which order levels are completed and the occasional choice between two alternative options. In the ten years that lie between the release of \textit{Deus Ex} and \textit{StarCraft II} there seems to be little changes.
For Noah Wardrip-Fruin the problem ultimately lies in the shape of the underlying processes: the processes that underly both the dialog tree and larger interactive story/mission implementations are rather uninteresting. He suggests that a new approach to game fiction is warranted and that this approach should be fundamentally different from the quest flags and dialog trees that govern most missions in games. (Wardrip-Fruin, 2009, 76). I propose that a closer inspection of the mechanics that control game missions offers plenty opportunities to arrive at a better shape for interactive missions.

6.5 Mechanics to Control Progression

The mechanics that govern progression through a level can be represented as Machinations diagrams. These diagrams can be added to a space graph in order to provide more detail. For example, figure 6.10 represents a rough representation of the Forest Temple space graph as was discussed in the previous chapter. In this representation fights and tests of skills are omitted, the focus is on the items that must be collected to finish the level: on the mechanics that control progression. However, where before keys were connected directly to a lock, keys now activate mechanics represented by Machinations pools and resources, which in turn operate locks in the space graph. Extending space graphs with Machinations diagrams allows the level to be represented in more detail.

The diagram might seem a bit overwhelming at first. The nodes that represent places in the level’s space all have a black outline, whereas the nodes representing mechanics all have a colored outline. The translation of a lock and key mechanism into a Machinations diagram is done with two pools and one resource representing the key. This lock and key mechanism is isolated in figure 6.11. In this diagram the key element in the space graph triggers the pool below it causing the resource representing the key to be transferred to the pool on the right and thereby activating (unlocking) the lock. Similar mechanics are used to represent those locks that are opened with multiple keys, as is the case with the monkeys in the Forest Temple level. In this case multiple pools and resources represent those keys which must be collected on a single pool to unlock new areas (see figure 6.12). In the forest temple level there are more lock and key mechanisms that are omitted from figure 6.10 because they operate only on a local scale. One of these mechanisms involves the bombing creatures that spawn at particular locations and which explode a few seconds after they have been picked up. The creatures can be used to destroy certain walls. This mechanism is different from the other lock and key mechanics presented thus far. However, it can easily be captured with a mixed graph representing both space and mechanics (see figure 6.13). In this diagram, the bombing element activates a source that produces a bombing resource that the player can use to trigger a drain on a destructible wall, another drain is activated once per five seconds representing the time the player has to use the bombing before it blows automatically. This also means that player might actually fail to perform the task to open the lock. In The Legend of Zelda: Twilight Princess this failure poses no problem, as the bomblings respawn quickly after they have been
Figure 6.10: A simplified space graph of the Forest Temple level extended with Machinations to represent mechanics. The space nodes and edges have a black outline, the Machinations nodes and edges are all colored. In this figure: b = boomerang, e = entrance, g = goal, k = master key, m = monkey.

used, allowing the player to try again. While the player has a bombing, the source producing new once is deactivated to make sure there is only one active bombing in play. In addition, the time constraint (bomblings explode a few seconds after the player has picked them up) constrains this mechanism to locks that can be reached within a few seconds after grabbing a bombing. Hence the use of a source and the additional drain in figure 6.13.\footnote{Although the diagram does not show that players might loose health when they fail to use the bombing in time.}

In the remainder of this chapter I will focus on the theoretical possibilities and difficulties of relating levels and mechanics using Machinations diagrams to extend mission and space graphs. From the discussion in chapter 4 it became clear that game mechanics benefit from having feedback loops. The lock and key mechanics discussed above do not involve any feedback. Feedback needs a closed circuit that consists of at least one state connection that is not an activator or event; none of the mechanics above fulfill those requirements. There are two design strategies to include more feedback in the game mechanics that control progression through a level: 1) designers could develop lock and key mechanics that involve feedback directly, or 2) progress itself could function more like an abstract resource that can be gained (and might be lost) through mechanics that operate on a fairly large scale. Roughly speaking the first strategy boils down...
Figure 6.11: The lock and key mechanism in isolation. In this figure: e = entrance, g = goal, k = key.

Figure 6.12: The lock and multiple keys mechanism. In this figure: e = entrance, g = goal, k = key.

Figure 6.13: The bombling lock and key mechanism. In this figure: b = bombling, e = entrance, g = goal.
to adding interesting feedback mechanics to control locks to a space graph, and the second strategy boils down to replacing missions graphs with more sophisticated mechanics. Both strategies are explored in the following two sections, respectively.

Feedback in the game mechanics that interact with level structures constitute an important and interesting category. As was mentioned in the general discussion of the relation between feedback and emergence in section 1.5, feedback that traverses between different levels of organization within the system contributes to stronger types of emergent behavior. Internal economy and level design constitute such levels of organization. Games that are set up so that their internal economy and level design affect one and other during and as a result of play, have a greater potential for emergent behavior.

6.6 Feedback Mechanisms for Locks and Keys

To create lock and key mechanisms that involve feedback, a good starting point is treating the keys more as a resource that can be produced and consumed. For example, figure 6.14 represents a mechanism where players need to “harvest” ten keys before they can open the lock. Feedback is implemented through the application of dynamic friction on the number of keys players have collected. The more keys that are collected, the quicker the keys are drained. This makes it somewhat harder to estimate how many keys need to be harvested to get past the lock. Obviously, this gets even more difficult as the distance between the location where keys can be harvested and the lock increases. However, the mechanism is not very interesting in itself: it boils down to harvesting enough keys and then make a run for the door, there is little strategy involved.

In an attempt to create a more interesting mechanism, we can apply the dynamic engine pattern (see chapter 4 and appendix B) to a lock and key mechanism. Figure 6.15 represents such a mechanism. This time players need to collect more than 25 keys in order to proceed, but now they have the option to invest 7 keys to increase the harvest rate by 0.5. However, this mechanism is probably too simple, too. It is not very difficult to find out what number of upgrades is ideal for for this scenario. But even that is not necessary, play-

\[9\]

As it turns out, the ideal number is actually one or two upgrades, both arrive at 26 keys.

\[\text{Figure 6.14: Simple feedback applied to a lock and key mechanism. In this figure: e = entrance, g = goal, k = key.}\]
ers can achieve the goal without needing to upgrade at all. These weaknesses should not come as a surprise: as I argued in chapter 4 one feedback loop is generally not enough to create an interesting dynamic mechanism. The particular strategies are the direct result of the use of the dynamic engine pattern. Games that do mostly rely on just a dynamic engine as their sole, or single-most important, feedback loop, such as MONOPOLY, usually include random factors to make it more interesting and unpredictable, but that is not the direction I want to explore here.

To create a more interesting lock and key mechanism, we can complement the dynamic engine pattern by some form of dynamic friction (see figure 6.16). In this case enemies spawn that will consume the harvested keys. Now players have to balance between three tasks: harvesting, upgrading and fighting the enemies to keep their numbers down. This is not a trivial task, playing the interactive version of the Machinations diagram\(^\text{10}\) is already a fairly interesting challenge. Simply harvesting will probably not bring the player very far, and although it is possible to achieve the goal by switching between harvesting and fighting, this requires players to maintain a delicate rhythm of switching between the two for a long time; it is very hard to accomplish. Players need to find a balance between the three actions in order to reach the goal. When the fighting is made skill-based, then the most effective balance can actually vary between individual players. In its essence the mechanism is very similar to the basic gameplay mechanism of the real-time strategy game I have discussed earlier: players need to balance between harvesting raw materials, fighting, and upgrading.


6.7 Progress as a Resource

In many games that integrate emergent gameplay with progressive level design, the goal is to reach a certain location. This goal can be represented abstractly with a very simple diagram (see figure 6.17). In its essence, the ‘core mechanics’ of The Game of Goose are similar. The main elaborations this game implements are the use of dice to determine how much progress the players are making each turn and the chance that a player might gain extra progress, lose turns or lose all progress. More advanced games elaborate more: the most common strategy for action-adventure type games is to make the production of progress non-trivial and interesting in itself. To a certain extent, the experiments with lock and key mechanisms that involve feedback, discussed in the previous section, fall into the same category. This section seeks to go one step further, it explores the possibility to involve progress itself in a mechanism to make the gameplay more dynamic. Put differently, this section explores how feedback mechanics might replace typical missions found in most games of progression.

An interesting and fairly abstract implementation of a progress mechanism can be found in the latest edition of Warhammer Fantasy Roleplay. The rules of this tabletop role-playing game include the concept of a ‘progress tracker’ as a generic tool to manage the players’ progress towards a single goal, competition for conflicting goals by multiple parties, or even the players’ party’s internal tension and friction. The progress tracker takes the form of a track that can be built from individual track pieces (see figure 6.18). This allows the ‘game master’ to build tracks with lengths that suit the current situation.\footnote{In a tabletop role-playing game one player takes the role of the game master (also known as}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure616.png}
\caption{The dynamic engine pattern and dynamic friction applied to a lock and key mechanism. In this figure: e = entrance, f = fight enemies, g = goal, k = key, u = upgrade.}
\end{figure}
the track indicate the progress of individual parties. The rules suggest a number of ways a progress tracker can be used to facilitate scenes that involve races, chases, investigations. The tracker can also be used to represent a time limit by forcing the players to complete a certain task before a marker on a progress track reaches the end, or to create tension by using it to track the build-up of some “looming danger” unknown to the players.

Crucially, progress tracks in WARHAMMER FANTASY ROLEPLAY do not only track progress towards some goal (or danger), they might affect gameplay as well. For example, in the scenario that is published with the rules, the progress track to represent the players’ investigation into some secret cult includes a special position. Once the players’ marker reaches this position, the game master should provide the players with an extra hint in order to speed up their progress. This occurrence creates a one-off, positive feedback loop. Similar events on the party tension meter, a progress track that is part of the core rules, can cause the players’ characters to suffer additional stress, fatigue or wounds, causing destructive feedback.

As WARHAMMER FANTASY ROLEPLAY illustrates, progress mechanics can be used to cause feedback, and this is an excellent way to involve progress in the dynamic behavior of the game. However, more sophisticated forms of feedback can be used to evolve this further. A suitable pattern to accomplish this is the escalating complications pattern (see figure 6.19). With this pattern, which is found in simple, emergent games like PAC-MAN or SPACE INVADERS, the player’s goal is to complete a number of tasks. In PAC-MAN this task is to eat all available dots; in SPACE INVADERS it is to destroy all alien invaders. The task is getting progressively harder as the player is making progress. The dots get harder to reach in PAC-MAN while the alien invaders start to move faster and faster as their numbers decrease. A slightly more complex variation of the pattern, called escalating complexity can be found in TETRIS. In this variation some form of complexity is created and it is the player’s task to keep this complexity under control. However as complexity increases this task gets progressively more difficult, usually another mechanism ensures that complexity is produced at an increasing rate (see figure 6.20). In TETRIS the blocks cause dungeon master or storyteller). Where normal players usually control one character, it is this players responsibility to set the scenes and take care of all other characters in the game. The game master and the players are not opponents; they collaborate in creating an entertaining game experience.
the complexity, and the game speeds up every time the player reaches a new level.\footnote{For a more detailed discussion of escalating complications and escalating complexity see these patterns’ description in appendix B.}

Applied to progression, it is possible to model the progress towards a certain goal and have that progress affect the mechanics. In a sense, games of progression have always mimicked this effect by ordering a fixed sequence of challenges roughly from easiest to the most difficult. Nonlinear missions with alternative branches can be built using a similar principle in order to create dynamic levels with more replay-value, but as has been argued before, this strategy is not very effective. Most of the time many more branches and challenges need to be created than an average player is ever going to see. By creating a system where story-like progression emerges directly from the game mechanics, endless possibilities can be created efficiently. When the mechanics are set up to produce enough variety, this could lead to games where interactive experiences, and perhaps stories gain a whole new dimension.

In order to illustrate how a game like this might work, I will discuss an independent Flash game I designed and developed myself.\footnote{The reality is that I could not find a published game that could serve as an example. Games where progression truly emerges from mechanics are quite rare, and the examples I know of do not lean towards storytelling. SEASONS has the advantage of being a game that follows the tradition of games of progression, while it does treat progression a little different than most games in this tradition. While developing SEASONS I focused more on the story structure of the game than I explored the emergent properties of the game’s ecosystem. Originally it was an experiment with a frame story structure for games first and foremost. At the same time, the contrast between the highly linear stories and the open sandbox-like frame story was deliberate. As was the reflection of the player’s progress through the state of the environment in the main quest.} In this game, SEASONS (see figure 6.21), the player controls a few characters in a 2D platform game constructed as a frame story. In the frame story itself, the player controls a girl that lives in a land that lost all color and is covered in perpetual winter. She travels to her grandfather who tells her stories of the past, when the world still...
had colors and seasons. These stories are also playable levels during which the player controls other characters. By listening to her grandfather’s stories the girl learns how the land lost its color and why the cycle of the seasons stopped. The player is encouraged to use that knowledge to slowly restore the land back to life. The girl needs color magic to do so, utilizing color magic, she can restore elements back to their original state. Certain elements, such as trees and living creatures, in return produce new color magic for the player to use. She also needs color magic to perform double jumps, to fly, or to control the wind, amongst other things, all of which are necessary actions to reach the places where she can restore crucial elements or creatures and to ultimately locate the evil villain. In a way, her progress is reflected in the current state of the environment, the more color is restored the closer the girl is to reaching her goal; in order to reach the villain, the girl needs to restore all colors to the land, and it is through this restoration that the villain is defeated in the end.

When I originally designed SEASONS I was aware of the powerful dynamic engine that is created by its main mechanism: by restoring plants and creatures the player can easily get more color magic. I countered this by adding static friction: restoring plants only works for a limited amount of time. Figure 6.22 represents this mechanism in a Machinations diagram. Apart from the dynamic engine, the most important sources of color magic in SEASONS are reaching certain locations which would unlock and supply the player with a new type of color magic and the stories themselves, as the secret stars the player can find in these stories supply the player with extra color magic in the frame stories.

In hindsight, it would have been more interesting to use a mechanism that would have implemented the escalating complications pattern. For example, by
having restored elements also feed the aggression of those elements and creatures that have not been restored, requiring the player to spend more color magic in order to avoid them while traveling the world or even to actively oppose them (see figure 6.23). This would in effect aggravate the contrast between those parts of the game world that have been restored and those that have been not restored and cause a gradual increase in difficulty that is an emergent result of the mechanics, and not of careful, progressive level design.

A game set up like this, still relies on triggers to advance the storyline. In SEASONS reaching certain locations in the frame story will unlock new stories to be told by the girl’s grandfather and a few other characters in the game. Completing these stories will sometimes unlock a new type of color magic or direct the player to those locations in the game’s frame story where they can be unlocked. This set-up works, but much progression is still designed in advance. There are only a few alternative routes the player can take within this structure. This could also be changed. In SEASONS there is a rudimentary ecosystem that produces color magic. This economy could be elaborated so that instead of only one or two restored elements providing the girl with all color magic she needs, the player needs to structure many more elements. By relocating creatures and activating plants the player could build and tune an economy not unlike the city-building in CAESAR III (see above). By making a few changes the ecosystem in SEASONS could be rewired by the player to produce the color magic that is needed for the present task. Reaching certain objectives in this economy could still trigger story events, but the control of these events is far less linear, as it would be fairly easy to design the system in such a way that the player has many options in respect to what objectives to go for first. As with CAESAR III or SIMCITY the game would be able to accommodate a multitude of player strategies, and the game’s world would reflect a player’s preferences. It would require a multi-part ecosystem that could function in roughly the same way as the basic economy of CAESAR III (see figure 6.6).
6.8 Conclusions

In games, mechanics and levels cannot be separated. Games use level structures to teach the mechanics to players, and use special mechanics to control players’ progress through a level. In strategy and management simulation games, players typically have much control over the game economy that is built from the individual mechanics. However, games that find a good balance between the two are rare. One would expect missions to be the place where levels and mechanics converge. But as games have evolved over the past few decades, and ways have been found to create detailed spaces and articulate mechanics, the evolution of missions and mechanics to control progression has mostly stood still. As a result it is rare to find a game that truly integrates emergent gameplay and progressive level design.

In this chapter I have explored the relation between mechanics and levels in order to find a better balance between these two elements of game design. Leveraging the formalisms developed in the previous two chapters, I have explored how mechanics that control level progress could benefit from implementing feedback loops, but also how progress itself can be integrated better in the general economy of a game. These suggestions are preliminary; apart from some promising prototypes they have not been implemented and have not been thoroughly tested. My intention in this chapter was to illustrate how games might integrate these two important elements of game design. The possibilities for the convergence of progression and emergence in games are far greater than the illustrations in this chapter.

The main point, however, is that emergence and progression are not two separate dimensions of game design. Using the right tools designers can shape
emergent mechanics to produce progressive experiences, and by having a clear perspective on a game’s internal economy and mechanics designers can structure levels that go beyond a structured learning curve. The internal economy of a game and its level designs constitute different levels of organization. General theory on emergence suggests that feedback that traverses these different levels creates a greater potential dynamic behavior than feedback that operates only within one of these levels. In this way mechanics that drive emergence and progression can be combined to create compelling game experiences that offer great freedom to the player.