Iteration and Procedurality in Game Studies

Gábor Zoltán Kiss

Abstract

There is a conceptual difference between traditional media and the video game form: while the former is characterized by technological and formal standardization, the latter stays for forty years in a specific experimental and cumulative state. Although we are aware of its technical, formal, and genre changes, we cannot see traditional media as cumulative. Video games, on the other hand, are iterative: most of them exist in a continuous technical and mechanical development, which gets in the way of conceptual standardization.

Iteration is driven by procedurality, which is not the programmer's privilege. It is a general cultural competence, and we can see it in gaming's best. Last year's *SpaceChem* (2011), a design-based puzzle game is a creative laboratory for experimentation, prototyping and manufacturing. It demonstrates the procedural operations within, and the iterative process in the making of games. *SpaceChem*'s gameplay can be summarized in one sentence: As an engineer, our aim is to solve a particular problem with the most apt and most rational program. Our duty is to analyze engineering systems, various contraptions and substances, and to shape puzzles from them. Traditionally puzzle games introduce elegant and sophisticated riddles, and we play them for their Eureka moments. *SpaceChem* goes further. It encourages us to break down the problems and create new ones, to prototype and manufacture new solutions, to iterate on the game's mechanical and our mental processes. *SpaceChem* is a possibility space in the literary sense, a framework which allows us to plan, analyze, break down and assemble our own puzzles. Video games are complex iterative and procedural systems, and we should approach them iteratively. The paper will analyze how *SpaceChem* helps us to recognize their inner workings, so we can build new conceptual prototypes for them.

Key Words: Iteration, procedurality, prototyping, game studies, conceptualization, puzzle, programming game, *SpaceChem*.

*****

There is a conceptual difference between traditional media and the video game form. While the former is characterized by technological and formal standardization, the latter stays for forty years in an experimental and cumulative state. Although we are aware of its technical, formal, and genre changes, we cannot see traditional media as cumulative. Video games, on the other hand, are advancing through iterative phases: Most of them exist in a continuous technical and mechanical development, which gets in the way of conceptual standardization. To sum up the difference between traditional and new media, it is evident that games do not recount the history of old media. We are not witnessing the birth of a new medium with videogames, as we have seen it in the case of cinema, as the ground to make such comparisons simply does not exist anymore. The problem with the "old" and "new" media comparisions is that we can't really see the difference between the two (for historical and perceptual reasons), and we need good examples to illuminate the dissimiliarities. What seems to be certain is that the new iterative media does not allow us to reach back to old tropes or viewpoints, and it gives rise to an alternative and *iterative* concept of "the canon," or the object of preservation. Video games, as the technological vanguard of new media, are complex iterative systems, and we should approach them iteratively. In the following paper I will argue that puzzle games can help us recognize the inner workings of games, so we can build new conceptual prototypes for them.

Iteration is driven by *procedurality*, a term which is not granted as a privilege to the programmer. Procedural thinking is a general cultural competence, and we can see it in gaming's best. It is not the expertise of writing in one or more programming languages, but the ability to recognize procedural logics in diverse artifacts (Mateas 2005). There are several various tools available for us to master proderural thinking, from high level programming languages to educational programming, from plain scripts to graphical programming environments and complex level editors. Programming has always been introduced by information science and education as objectified mathematics, even though it has an artisan, creative, and iterative element to it. Traditional informatics conceals the fact that programming is an inspired, often unpredictable activity, that coding has always been characterized by the same kind of playfulness that is typical to all creative endeavours. As Ira Greenberg puts it, creative coding "allows you much greater freedom to build in levels of control [to any system], and also levels of randomization and even irrational processes that don't always work but can lead to exciting, unexpected results" (Greenberg 2007: 1). Procedurality supports creativity and the aesthetics of accidents, even if they do not belong to the traditional idea of programming.

Procedural thinking manifests itself in many ways, even in playful forms, and it is a phenomenon worthy of serious analysis. Last year's *SpaceChem* (Zachtronics Industries, 2011), a design-based puzzle game is an exceptional case of playful procedurality. It is an inspired testing room for experimentation, prototyping, and manufacturing: It demonstrates the procedural operations *within*, and the iterative process *in the making* of games. *SpaceChem*'s gameplay can be summarized in one sentence: As an engineer on a distant space colony, our aim is to solve specific problems with the most apt and most rational programs. Our duty is to analyze engineering systems, various contraptions and substances, and to shape puzzles from them. Through the game the player has to reply to particular problems through the game's interface with creative and accidental results (Armitage 2011).

One can argue that *SpaceChem* belongs to the sub-genre of programming games, a specific type of puzzle that allows the player to program visually or type in actual code to solve problems. The genre can be traced back to such renowned titles as *ChipWits* (Doug Sharp, Mike Johnston, 1984), *Robot Odyssey* (Mike Wallace, Dr. Leslie Grimm, 1984), *Carnage Heart* (Artdink, 1995), and *Corewar* (1984). *SpaceChem*, as the aforementioned titles, is a programming game: it transforms the iterative process into gameplay. In other words, it makes us play through a process (on an abstract level) that is analogous to the game's development. *SpaceChem* underwent several changes and iterations until it reached its final form, and it replays the same process in its puzzles. The developer's former games also deal with a similar problem from various angles, and arguably they are preliminary prototypes to *SpaceChem*. In *Kohctpyktop: Engineer of the People* (2009) "you play as an engineer working in a semiconductor factory designing integrated circuits based on specifications provided to you"; in *The Bureau of Steam Engineering* (2009) "you play as an American steam engineer at the beginning of an alternate civil war"; in *Ruckingenur II* (2008) our task is to "reverse engineer electrical circuits"; while *Codex of Alchemical Engineering* (2008) is about "programming 'manipulators' to move, transmute, and bind alchemical 'atoms' into complex compounds" (quotes from the developer's website). They share the common idea of transforming engineering systems to modular and combinatory puzzles, to analyze and manipulate various substances and mechanical parts.

In *SpaceChem* the player construct "elaborate factories to transform raw materials into valuable chemical products," in the required quantity and during a convenient timeframe. Elements and compounds are carried by virtual production lines and so-called "waldos" to their destinations, while various manipulators are dealing with their dissociation, rotation, slowing down, or assemblage. The two waldos are the key componens of the mechanical process: They execute a specific set of processes, predefined by the player. They are following a well-defined path, activating elements and manipulators, grabbing and dropping compounds, rotating and synchronizing them, while creating and breaking chemical bonds. Manipulators are the visual equivalents of the game's rules and possibilities (or, in an abstract sense, the program's instructions), while the two waldos are the executors of the temporary rules prescribed by the player to solve actual puzzles.

The game could be described as a rather simple visual programming language with some base rules and instructions, although they provide infinite creative solutions to the game's finite number of problems. The complex transportation networks of the virtual production lines are carrying and transforming the elements and compounds at hypnotic speed, through a recurring process. Every puzzle contains a set of compounds or elements that needs to be produced from a given number of elements or other compounds, through several cycles, in the least amount of time – and from the least number of mechanical components possible. The actual task ends when the appropriate amount of compounds has been produced, followed by a quick statistical calculation concerning the number of manipulators, reactors (sets of manipulators), and cycles. Also, the game compares the actual results with other engineering solutions, which adds the element of replayability and social dimension to an otherwise solitary experience. It urges the player to reprogram the existing visual scripts, and to seek for better, more efficient solutions to the same problems. The game saves all solutions, so the player can break down, revise, and substitute them with more efficient and more rational ones at a later point. Players can also compare their results at SolutionNet (http://spacechem.net/), "a site devoted to sharing and comparing solutions" of the game, and they can examine other solutions for the same puzzles.

The game's procedural workings manifest themselves on various levels through the sci-fi theme. While none of them have any significant effect on the gameplay, they add much to the overall feel of the game. *SpaceChem*'s visual design is twofold: the menu suggests a cartoony science fiction style, while the actual puzzles are abstract and informative. (The game's sound and music fit well into both styles, as the game has a lush audio palette from orchestrated heroic pieces to indusstrial noises.) The story itself, as subsidiary it may be, is present all the time: It is built upon the game's paratexts (mainy the menu and the tutorials), and it stems from the descriptions of the official website. According to the game's flavour text, as a deep space reactor engineer, our task is to produce and assemble valuable chemical compounds on faraway space colonies, and to shed light on some menacing occurrences inside and outside the titular firm. The chemical references and metaphorics are revealing in regard of the theme: The game presents each puzzle through specific chemical elements and compounds, and it calls for appropriate "reactor" setups in every situation. However, the chemical theme is present solely for expositional reasons, and it does not restrict the "engineering" gameplay. The game's chemical terms (like fusion, bonding) are mere metaphors, as they have much more to do with the rules than actual chemistry. The point at issue here is that the game's core system does not refer to chemistry but to various programming concepts.

The game's graphical programming language is complicated enough to allow the player to give clever, diverse, and inspired solutions to every puzzle. Puzzles are divided to simpler tasks, independent elements or simple compounds with real or fictitious chemical equivalents. Basic levels ask us to make relatively simple compounds like carbon dioxide ($O=C=O$) or water ($H–O–H$), while a complicated task makes us produce hydrogen peroxide ($H–O–O–H$) from hydrogen gas ($H2$) and oxygen ($O2$). Puzzles can be solved successively or simultaneously, and while the latter reduces the number of cycles needed, it increases the possibility of collisions between atoms, which leads to reaction error immediately. Parallel execution recalls the processes of multithreaded programs, where simultaneous threads use the same resources, running independently. *SpaceChem* illustrates other programming concepts as well, like synchronization, debugging, and optimalization, and it makes us think in terms of cycles of synchronous or asynchronous processes. Puzzles break down to simpler, parallel, rationalized tasks, to simultaneous objectives and processes – represented by the increasingly complicated compounds and conveyed by the two waldos in the game.

According to Steven Johnson, “the mental labor of managing all these simultaneous objectives” can be seen as a telescope – hence his term, telescoping – “because of the way the objectives nest inside one another” (Johnson 2005: 54). Johnson argues that

> the closest analog to the way gamers are thinking is the way programmers think when they write code: a nested series of instructions with multiple layers, some focused on the basic tasks of getting information in and out of memory, some focused on higher-level functions like how to represent the program's activity to the user. (55)

SpaceChem emulates the mental processes demanded by the code on two levels, through the game design and its gameplay.

The game is divided to increasingly complicated problems, to progressively difficult “planets” and “levels”. The second level of the third planet (“It Takes Three”) asks us to produce ammonia (NH3) via the combination of three units of hydrogen and one nitrogen atom (nitrogen gas). SolutionNet offers several solutions to the puzzle, through various reactor parts. One of the most inventive solutions transforms the puzzle to a twofold rebounding process, by using 142 cycles and 14 reactor parts. The first waldo activates and prearranges the hydrogen atoms, while the second waldo transponts the nitrogen to the hydrogens’ arrival site: The second waldo activates the chemical bond, then the first one grabs the final molecule and carries it to its destination. The idea here is to let the reactor carry the atoms back and forth to solve the same repetitive process with an extremely limited set of items. (There is a significant drawback, though: the program needs numerous cycles to execute, which is unfavorable by all means.) Another paradigmatic solution (102 cycles, 34 reactor parts) offers a circular system: it parallelizes the task, producing several molecules during one cycle. (Again, the solution involves the same problem as before, the other way around: While the process lasts for a relatively short time, it involves much more reactor parts.)

SpaceChem’s main merit is that it makes us realize programming patterns through its gameplay. It grants us the tools to build increasingly efficient solutions to progressively complex problems, and it introduces the principles of efficient programming as puzzles where time savings and spare parts mean better results (the same way as cycle time optimization means actual efficiency in hardware programming). Traditionally, puzzle games introduce elegant and sophisticated riddles, and we play them for their Eureka moments. *SpaceChem* goes further: it encourages us to break down the problems and create new ones, to prototype and manufacture new solutions, to iterate on the game's systematic processes, and to understand the inner mechanical workings underneath it. According to Zachary Barth (the game's designer) puzzle development try to direct the player toward a predesigned solution. *SpaceChem* reverses the usual process: it gives the player a frame and a theme with assignments, not knowing whether they are solveable or not (Barth 2011). The game is a possibility space in the literal sense, where players plan for, analyze, break down, assemble, and test their own puzzles: It is a creative laboratory to play with prototypes, to experiment with creative solutions.

Programming games like *SpaceChem* seems to be specifically relevant from a proceduralist standpoint, as they refer to the inner workings of the medium. Proceduralists believe "that computer games present a technological and cultural exception that deserves to be analyzed through the ontological particularities that make computer games unique" (Sicart 2011). According to proceduralism, computational quality is a creative, multi-layered attribute, based on "the computer's ability to construct complex behaviours driven by many variables that the user can manipulate and explore" (Ian Bogost at NYUGC). The wording here is misleading to some extent, and it leads to a tautological definition. To define computer games by their computational quality seems evident; something along the lines of "the uniqueness of film is its cinematic quality" or "literature's uniqueness lies in its written nature". More importantly, the computer is not the "constructor" of complex behaviours: It executes absurdly sophisticated processes with many variables, designed and selected by the programmer, the designer of the code. The designer is the actual agent in the process, not the computer, as *he* defines the rules and the logics, *he* gives the framework and the variables for the computer to "construct," and *he* sets the rules and the possibilities for the user to "manipulate and explore". The computer definitely has an initial effect on the rules and logics of the game. However, the designer will take the leading action by playing with and selecting from the platform's offerings (not to mention the fact that in most cases the brightest programmers will find a way to work around the constraints of the system). This is a somewhat sobering realization: in order to see the inner workings of the game, we need to demistify the technological processes of processing and talk about the systems *above* the hood. *SpaceChem* helps us in this regard; through its gameplay, it makes us reconcile the developer's business with the player's task.

In his most recent book Ian Bogost declares that "unlike playground games or board games, videogames are computational, so the model worlds and sets of rules they produce can be far more complex" (Bogost 2011). This could be an overemphasizing and enthusiastic definition, carried away by the “computer” in computer games. According to Bogost, “the ontological particularity” of the computer is its capacity to sophisticate the processes beyond recognition, to behave like an automatic, self-propelled system, having an end in itself. Again, the question can be addressed from the designer’s standpoint. Should we see emergent gameplay or interactive storytelling as possibilities provided by the computer, or as possibilities prepared by the designer and given to the computer to calculate? Usually, we think about emergence as those moments when the player feels that he made something beyond the system's plausibilities, exploiting or circumventing the pre-planned possibilities. In those moments, the system’s computational capabilities and procedural virtues would not suffice without iteration, without letting the player feel he actually made something unforeseen within the game's imaginable. *SpaceChem* is brimming with these moments: it lets us think in terms we never thought we could comprehend.

What can we learn from *SpaceChem*? How does it make us recognize the inner workings of the medium, and more importantly, what does it signal in an industry so obsessed with simple tasks and instant gratification? Videogames motivate us “through the desire to know what happens next [or rather] […] what the next puzzle is” (Poole 2010). They don’t motivate us, however, through *solving* these puzzles or streamlining our results. *SpaceChem* and its breed of programming games pursue the iterative pleasures of the medium: they are not interested in futile computational exercises or narrative enigmas, but in maintaining an iterative system or service to (re)solve problems. Game studies’ endless quest to define “games” is a somewhat fruitless effort: not because of the form’s complexity, but because its ever-changing nature. The medium’s inherent historical and productional logic presumes that the medium is incapable of turning back, that its iterative logic entail functionalism and irreversible accumulation. *SpaceChem* supports the latter idea, and it leaves us as rather bemused problem solvers: not by the computer’s ability or its narrative merits, but by the creative synergy of the system and our engineering capabilities.

Bibliography

Armitage, Tom. “Programming Games,” *Hide & Seek*, May 25, 2011, accessed May 10, 2012, http://www.hideandseek.net/2011/05/25/programming-games/.

Barth, Zach. “IndieGames.com Podcast 17: Zach Barth on SpaceChem and Infiniminer,” by Michael Rose, March 8, 2011, accessed May 10, 2012, http://indiegames.com/2011/03/indiegamescom_podcast_17_zach.html.

Bogost, Ian. *How to do things with videogames*. Minneapolis: University Of Minnesota Press, 2011. Kindle edition.

Greenberg, Ira. *Processing: Creative Coding and Computational Art*. Berkeley, California: Apress, 2007.

Johnson, Steven. *Everything Bad Is Good for You. How Popular Culture Is Making Us Smarter.* Penguin Books, 2006.

Mateas, Michael. "Procedural Literacy: Educating the New Media Practitioner," *On the Horizon* 13 (2005): 101-11. Accessed May 10, 2012. http://users.soe.ucsc.edu/~michaelm/publications/mateas-oth-2005.pdf.

Poole, Steven. "(Can't Get No) Satisfaction," *stevenpoole.net*, January 25, 2010, accessed May 10, 2012, http://stevenpoole.net/trigger-happy/cant-get-no-satisfiction/#more-497.

Sicart, Miguel. "Against Procedurality," *Game Studies*, volume 11, issue 3, December 2011, accessed May 10, 2012, http://gamestudies.org/1103/articles/sicart_ap.

Gabor Zoltan Kiss is assistant lecturer of literature at the University of Pecs. He earned a Ph.D. in media theory from PTE in 2007. Gabor Kiss is a scholar of literature and media studies, film history, and popular culture. Currently he is working on an introductory coursebook on videogames.