System Thinking in Game Design

Advanced game design for entertainment games, serious games and gamification

Anders Nordby (anders.nordby@inn.no), Håvard Vibeto, Harald U. Sverdrup

Department of Game Development – The Game School, Inland Norway University of Applied Sciences, Holsetgaten 31, NO-2301 Hamar, NORWAY

Abstract

This article describes how system thinking can be used in advanced game design to make design drawings and simulate the game design before the game coding starts. This can save time and simulation code can be extracted and imported to the game engines (unity, unreal etc.) The article shows examples from an advanced game design course taught in the Game School at the Inland Norway University of Applied Sciences which demonstrate which elements of game development can benefit from this methodology. We discuss the course design and how students have used system analyses and system dynamics to design games and to balance game assets and resources by simulating game outcomes.

Keywords: game design, game development, serious games and gamification, system thinking, system dynamic simulation

1. Introduction

Salen and Zimmerman (2004) point out that all games are intrinsically systemic, meaning that all games can be understood as systems. Gee (2007) argues that people learn skills, strategies, and ideas best when they see how they fit into a larger system. Games systematic core are therefore good at helping players see and understand how subsystems fit into an overall system and thus facilitate learning and system thinking (Gee, 2007). Ernst Adams emphasizes that games are complex systems that consist of many parts, and that when these parts are put together, games can display unpredictable behaviour, resulting in emergence (Adams, 2014). Systems in games provide context for interaction, rules and behaviours that players can explore, manipulate and immerse themselves in. Although one can argue that games have an inherently robust systemic quality, especially modern videogames, this is not often discussed explicitly in the game design literature. Game design in practice tends to focus on how to design the ludic elements of a game -- its core gameplay mechanics, rules and goals, and the corresponding aesthetic qualities that make a successful game experience (Schell, 2008). Many modern videogame genres have become complex systems of varying rule sets and modes that the player needs to interact with and understand to successfully play the game. The player needs to understand and predict how the rules will react to their input and then develop different strategies to beat the game or human opponents (Tekinbas et al., 2014). Some genres offer massive, simulated worlds inhabited by NPCs (non-player characters) that each offer many different goals and interactions to the player. An understanding of system thinking when playing these games has in many ways become essential if players are to get the most from their game sessions. Indeed, today players can find videos, wikis, forum posts and other materials online that break down the many systems in videogames in order to successfully master them (Wallach, 2019).

Certain games and genres require metagaming -- the player must understand how the different inner workings of game systems function to play them successfully. Metagaming involves many different elements that affect the game, but are not in the game (Mora-Cantallops & Sicilia, 2018). This use of out-of-game information or resources to affect the player's in-game decisions may include community standards, statistics or datamining. Metagaming often focuses on the strongest, most effective strategies currently available in a game because today's videogames are in constant flux, and each patch or new downloadable content (DLC) can change a game's rules and systems (Blake, 2017; Peterson, 2016).

Designing and playing videogames both demand that participants develop a systemic understanding of the game with regard to not only the behaviour of the individual components of a system, but also the way the system operates and interacts with other systems within the game. As videogames are interactive, players must not only understand a game's systems, they must also decide the best way to intervene to change and manipulate these systems as well as understanding the connections between systems. This means that game designers must make game systems intelligible and interactive, while also ensuring players can have agency over the systems without breaking them (Tekinbas et al., 2014).

As videogames have become increasingly diverse and advanced, design trends have emerged

that system thinking can help explain. The growth in sandbox and open world videogames that create massive, interactive worlds that players can use for their own pleasure gives rise to a number of design problems. Many of these games showcase advanced systems on both macro and micro levels that are interwoven and which are supposed to give the player a feeling of meaningful play and agency. As sandbox and open world design become more widespread in the videogame industry, the task of balancing all these systems to create consistently good gameplay and experiences becomes more difficult (Doyle, 2015; Pramath, 2016). The same balancing problems can be seen in another dominant videogame trend, namely, the big multiplayer games that focus on a "games-as-a-service" business model. Games-as-a-service means that a game is constantly updated and monetized with new content that encourages players to keep playing and paying. Games-as-a-service can be found inside the MOBA (multiplayer online battle arena), FPS (first-person shooter), and MMOG (massively multiplayer online games) genres. All create online competitive modes where players compete yet also need to cooperate within teams. A lot of these games have many different heroes, abilities, and weapons that must be balanced against one another. Examples of these kind of games are League of Legends, Dota 2, Overwatch, The Division 2, Destiny 2, and World of Warcraft. Since many of these games are constantly being updated and new content introduced, there is always a chance that changes can alter the game in ways the game designers did not intend – and infuriate the player base. System dynamics simulations can help avoid these problems.

Finally, yet importantly, traditional game design often uses a linear approach when designing games (A causes B causes C causes D, etc.). This is understandable and what our brain wants to do, if not forced to think differently. System thinking, with its causal loop diagrams, does just that -- it forces game designers to discover non-linear feedback loops in games systems. (A causes B but B also has a feedback loop back to A, etc.). The real world is rarely linear. Feedback loops are everywhere, and good game design requires the game-world to be as complex and interactive as the real world.

Seen in this context, it is only natural that the game design process uses system thinking to develop the game. System thinking is a methodology used to understand how causal relationships and feedback loops function in everyday problems. It is comprised of two parts: system analyses and system dynamics. System analysis uses language and a set of diagrams to describe connections and casualties in a system. System dynamics is the numerical simulation of a system.

2. Scope and Objectives

We are lecturers at the Game School at the Inland Norway University of Applied Sciences, Norway. In this article, we discuss how we have changed the focus of a course in system thinking for game students from traditional system thinking to advanced game design. Students are given assignments where they must use system thinking to balance and further understand games and game design. The course, *Games and System Thinking*, is the follow up course to a more traditional course in game design and has been taught in the technical bachelor of the Game School since 2012. The article will discuss two examples of assignments given to the students during the course. Although the implementation of system thinking in the course is a work in progress, we will try to shed some light on two central questions in our current research: First, does system thinking improve students' understanding of how games work? Second, can system thinking in general improve game design?

3. Methodology

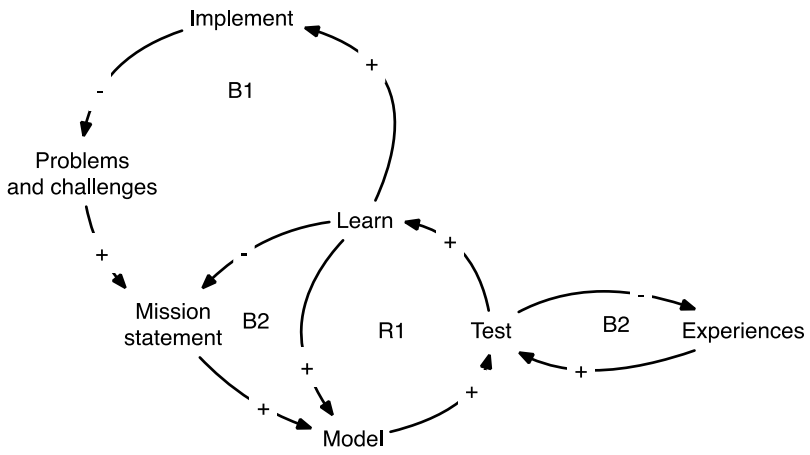
3.1 System Thinking

The main tool in system analysis is the causal loop diagram (CLD) as defined by Senge (1990), Sterman (2000), Haraldsson and Sverdrup (2004), and Sverdrup et al. (2020). A CLD describe how variables in a system are connected and influence each other. These variables can be causes or effects and are connected by arrows and a label that describes how the effect develops in relation to the cause. A plus symbol indicates that both cause and effect develop in the same direction (e.g., more cause creates more effect). A minus symbol indicates that they diverge from one another (e.g., more cause creates less

effect). Arrows can also indicate feedback – that is, the effect can result in changes to the cause. In addition to CLDs, system analysis uses flow charts to show what and how things flow through the system, and reference behaviour patterns (RPB) that show how the system develops over time (not shown here). Figure 1 shows an example of a CLD that describes how the system analysis works. All start with problems and challenges that are brainstormed and developed into a mission statement. The mission statement is then developed further into a model, which is tested. During the testing students learn and gain experience that is used to create better tests (the B2 loop). Learning often leads to corrections in the mission statement or model (the B2 branch). When learning is good enough, the game developers can implement the system, which usually leads to fewer problems and challenges to solve. The R indicates that the loop is reinforcing or always increasing, while the B indicates a balanced loop, meaning it will stabilize on a particular value.

Figure 1

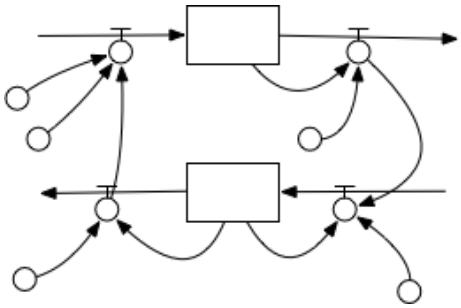
CLD Learning Loop



During the system dynamics component of system thinking, the numerical simulation of the system is developed. This is usually done with simulation software and involves two steps. First, a translation of the CLD using a simulation software, often through a stock and flow diagram (Figure 2), and then research to find appropriate and realistic numerical values or ranges for all variables in the system. The stock-flow diagram in Figure 1 shows how the CLD can be implemented through stocks and flows in a simulation. In a large systems simulation, finding the correct values can often be difficult and time-consuming. The translation from the CLD to the simulation software involves defining time variables and steps, establishing which values should be accumulated, which values should be used as input and output, and how mathematical functions should be defined in order to affect the system in the desired way. When the model is finished, different scenarios are simulated to confirm the model works (often by checking it against the real world). When the simulation aligns with the real-world scenario, game designers can use it to simulate how the system will behave with different inputs. In game design, it can be useful to randomize input values to simulate different playstyles and to run many simulations to learn which one results in the best gameplay.

Figure 2

Stock and Flow Diagram.



Note: The boxes are stocks, and the arrows indicate a flow.

Simulation data, mathematical equations, or pseudo code can be exported from the simulator (Stella Architect). Although data from simulations can be imported directly into the game engine (Unreal or Unity) and used when needed, pseudo code offers more flexibility if it is translated into the game engine programming language (C, C++, Java, etc.) and run directly in the game engine because it allows the students to rerun simulations in the game engine whenever the conditions of the original simulations change. Mathematical equations are the basis for the game design and can be used directly in the game programming.

3.2 Learning Methodology

The *Games and Systems Thinking* course uses a problem-based (PBL) learning approach (Nordby & Karlsen, 2014). Students are divided into groups with a tutor and the development process of the game is divided into steps. In each step, the student groups decide what is necessary to study in order to solve the challenge. Each step prepares the player for the next step until the game is done or they go on to the next game. Problem-based learning was first implemented in medical schools in the late 1960s by Howard Barrows and his colleagues (Barrows, 1980), and has since established itself as an independent learning method, especially in higher education (Pettersen, 2005), (Dochy et al., 2003), (Maastericht,2013).

The game development process is also seen as a "community of practice" where relevant school topics pop up as students try to create a game. This suggests a closer look at learning methodologies such as situated learning (Lave & Wenger, 1991; Wenger, 1998), "learning by doing" (Dewey, 1916), and "learning just in time" (Gee, 2007), but these are not discussed further in this article.

3.3 Research Methodology

Course designers and instructors use of system thinking as a tool for game design is quite new. Because of this we expect to have to work further to refine the course design. We see the course as a research project with action research as the research methodology. Action research is an interactive inquiry process that balances problem-solving actions implemented in a collaborative context with data-driven collaborative analysis or research to understand underlying causes, thereby enabling future predictions about personal and organizational change (Reason, 2001). Throughout the course, we collect notes, feedback from students, and reports and results from each assignment. Every year, we examine this data to see how the course can be improved and what information it provides about how system thinking functions in game design situations.

4. Theory

4.1 Introduction to State-of-the-Art Game Design

The need to have textbooks on game design and to foster an academic understanding of how to develop a good game became important as the videogame industry has grown. In its infancy, videogame production was the domain of engineers, programmers, and self-taught enthusiasts with no formal training in game design (Donovan, 2010; Kent, 2001). At the end of the 1990s, this changed. The videogame industry was

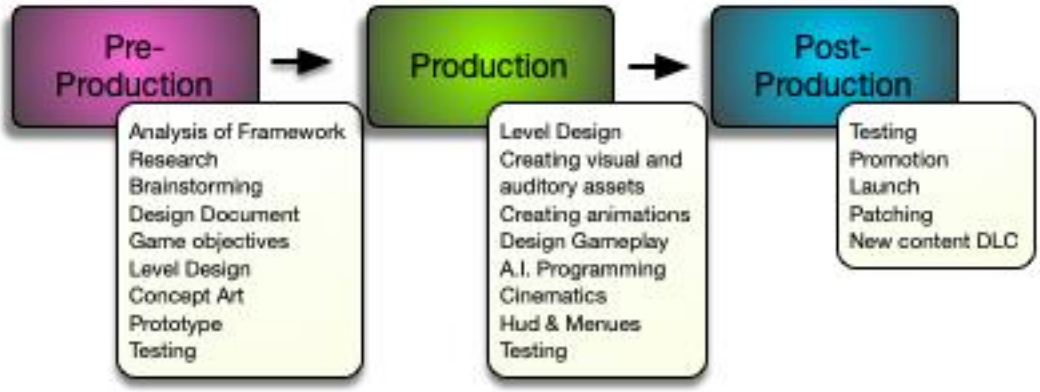
growing internationally, production costs were rising, and there was a need to professionalize game development (Egenfeldt-Nielsen et al., 2008). Videogame development grew from being one-person projects to large-scale productions that employed hundreds. Budgets increased, development phases were standardized, and specialized roles related to work tasks such as sound, programming, animation, and game design were developed. Videogames had become so big, expensive, and specialized that the gaming industry needed qualified professionals with formal education, and higher education institutions needed to educate professionals in a number of fields in game development. In response, game design literature emerged, with a goal of explaining how to create high quality videogames. One of the first books on videogame design was published in 1984 (Salen & Zimmerman, 2004). As the years passed and industry grew, more and more books on game design – with the main focus on videogames – appeared. Game design literature clearly emphasized the ludic aspects of videogames. Attention was given to rules, interactivity, core mechanics, gameplay, the development cycle, and commercial questions (Shell, 2008; Salen & Zimmerman, 2004; Adams, 2014; Rouse, 2005).

For the most part, this literature highlighted gameplay as a key element unique to videogames and that also needed to be the main focus of games. Gameplay is often viewed as the secret ingredient that makes games fun to play. But what characterizes gameplay? Richard Rouse, for example, has argued: "A game's gameplay is the degree and nature of the interactivity that the game includes, i.e., how players are able to interact with the game-world and how that game-world reacts to the choices players make." (Rouse, 2005). Rouse links gameplay to a videogame's interactivity. Accordingly, game design must focus on determining the choices available to players and considering the ramifications of the preferred alternative (Rouse, 2005). This means that videogames need to produce systems that can handle the input and output of the players' choices. As videogame engines become more advanced and hardware more powerful, videogames need to develop scientific methods to guide and help the designer in making good design choices.

However, many game design publications are written by veteran game designers who do not necessarily have a research or academic background. This often limits the game design literature because it is mostly based on subjective experiences and includes little research or methods from other disciplines. As a result, in this literature, practical, technical, and ludic questions in game design easily eclipse more theoretical examinations of how to use tools and methods to help balance games (see Figure 3). This is something the Game School at the Inland Norway University of Applied Sciences has tried to remedy – by introducing system thinking into game design education. Even though system thinking has not been addressed in game design books, some game designers and game researchers have used Unified Modeling Language (UML) to map specific gameplay elements with the goal of assisting the balancing and testing of features and systems. Adams and Dormans (2012) have developed a form of UML called Machinations to help designers and students of game design create, document, simulate, and test (primarily) the internal economy of a game (Adams & Dormans, 2012).

Figure 3

Traditional Game Design Development Structure



4.2 Where Does System Thinking Fit in the Game Design Process?

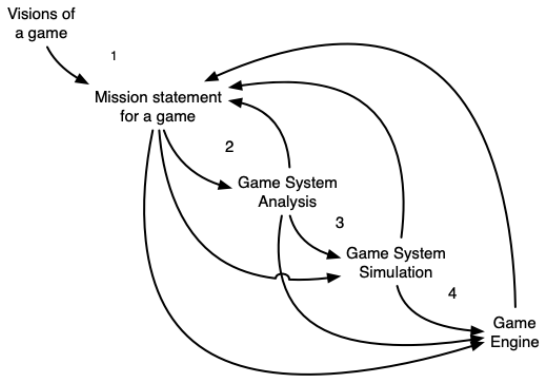
When using system thinking in game design, the process is approximately as follows:

0. Vision of a game
1. Mission statement: brainstorm and describe the game idea
2. System analysis -> Complete system analysis of the game idea and design a qualitative model of the game
3. System dynamics -> Simulate the model, test scenarios and export the code to the game engine
4. Game engine -> Develop programming and art (and maybe art system analysis) within the game engine

Unlike the traditional gaming model illustrated in Figure 3, in a system thinking developement process, all steps take place in preproduction. However, because production and postproduction might lead to changes in system analysis, simulations, or even the game idea or mission statement, feedback is an integral component of preproduction. This is shown in Figure 4 below.

Figure 4

Overview of the Game Design Process Using System Thinking



The process of adding system thinking tools to the game design process has usually been divided into the four steps shown in Figure 4. In this process the first step is brainstorming the game idea outside the constraints of any structuration tools. It is important to make sure creativity flows freely and is unhindered in this phase. As there are many descriptions in the literature on how to do this, we will not discuss this step here.

The second step in a system thinking approach is system analysis. This process, where elements from brainstorming the game idea are structured logically and qualitatively, uses the tools described in Section 3.1 introduction, such as CLDs, flow charts, and RBPs. An analysis component does exist in traditional game design, but only proven system analysis tools provide a solid scientific fundament on which to build the game.

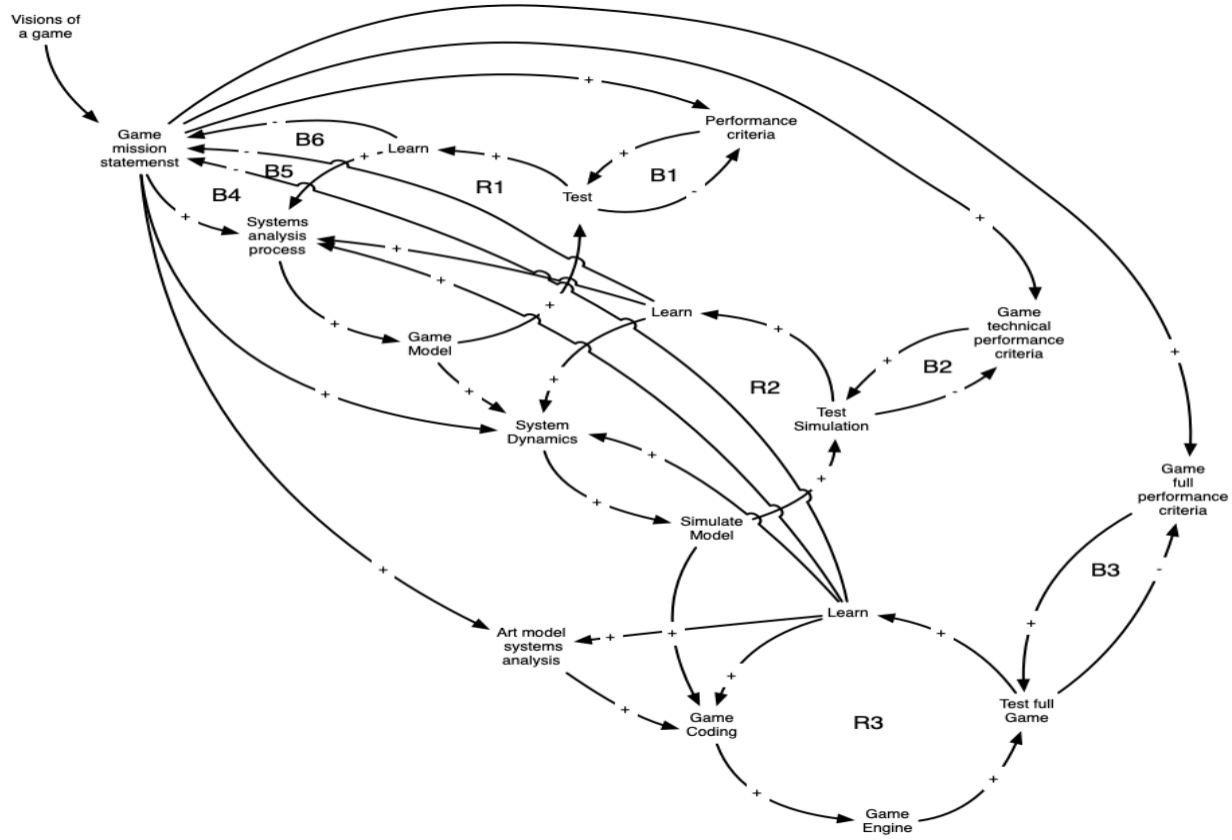
The third step, system dynamics, is where the game is simulated. This process involves designing stock-flow diagrams, finding values for the variables, and implementing the model in the simulation software. The simulation is then tested using scenario-based datasets. When it works as intended, the code is exported into the game engine (see more details on this below), creating a prototype of the gameplay. This step, too, happens in the preproduction phase. Assigning numbers can be an exacting process; often the developers must make qualified guesses based on the system analysis of the game idea. However, the process forces them to dig deeply into their game idea and game design documents to develop a quantitative understanding of how their game systems should work.

The fourth step is to make everything work within the game engine and, in the production phase, to add graphics. As this step is not very different from traditional game design, it will not be dealt with here.

Figure 5 illustrates this process in more detail. Note that all steps have feedback loops on all levels, which can entail revisiting and making changes at any level of the design. It is important that all steps are continually updated throughout the design process for the system analysis to always reflect the current version of the game engine.

Figure 5

The Game Design Process Using System Thinking



5. Case Studies

The game design and system thinking course at The Game School is centred on system thinking and includes game balancing and game design assignments. As the course is part of a broader gaming curriculum, as educators we want students to see the relevance of system thinking to advanced game design. During the course, students solve four assignments of increasing difficulty out of a total pool of 15 assignments/tasks. The first assignment is solved individually, but the next assignments are solved in groups that increase in size with each subsequent assignment. Since there are many different tasks and the students do not solve all of them, all finished assignments are presented in class so all students can discuss them and learn from them.

The task development follows the model given in the theory chapter. For the system dynamics simulations, the students use ISEE Systems' Stella Architect. When they have finished their designs and simulated a few scenarios, they present them to the class and teachers for feedback. This process usually forces them to correct their designs and run new simulations. The students then submit their revised designs along with a case report in which they present the final design and discuss the development process, the different scenarios they explored, and their progress and challenges along the way. As the assignments are folder-based, students can alter and improve their case solutions throughout the semester based on other cases they have solved. All these iterations force them to dig deeply into using system thinking in their game designs.

The next section of this paper presents two examples and two possible solutions to game design cases or assignments. The examples are from level 2 and level 4 in the class, where students are working in groups and cases have increased in difficulty. The first examples cover only one system, but in the second, several systems must be connected and work together in-game.

5.1 Example 1. Tank, Healer, Damage Dealer

The first example is a dungeon crawler game where the objective is to simulate different combinations of healers, tanks, and damage dealers. The assignment can be solved by making just one system and it serves well as an assignment at a moderately difficult level. It is open-ended in the sense that gameplay values – see Table X – can be interpreted in several ways. Aggro is one such example. Aggro is a term that describes how an NPC determines which player to attack first in an encounter. A player that has lost aggro during gameplay often cannot withstand attacks from the NPC, meaning that the game is often lost. In this case, different choices in how aggro is implemented will result in very different solutions – and the first time students were assigned this problem, every group interpreted and implemented aggro differently. The example in Table X shows only the simplest solution, based on World of Warcraft's aggro implementation, where only the player with the highest aggro receives damage. A dungeon party of 10 players (the maximum amount) is about to encounter the boss fight. Each of the classes have the following statistics.

Table X

Table Title Goes Here

	Tank	Healer	Damage Dealer
Damage rate	10	0	30
Healing rate	0	30	0
Health	30	10	20
Aggro rate	30	10	20
Damage reduction*	25%	0%	10%

The asterisk indicates the percentage of damage subtracted from the total damage dealt to the character. The boss has the following statistics.

Table Y

Table Title Goes Here

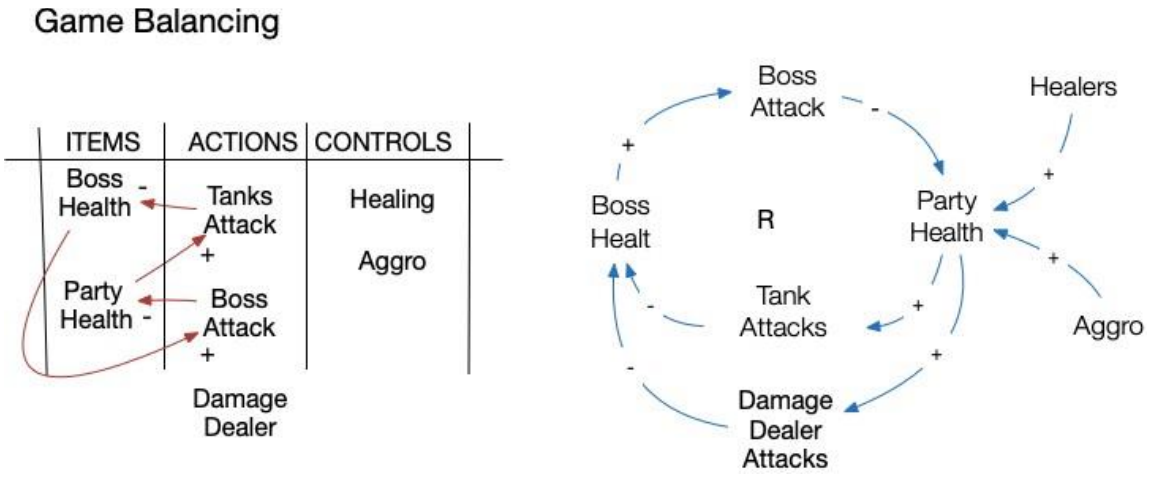
Damage	100
Health	1000
Healing rate	0

The assignment gives the students the following list of tasks:

1. Perform system analysis and create CLDs, flow charts and RBP for the system
2. Create simulations of the current balancing of the dungeon
3. Assign a limited random range of damage and healing instead of the constant shown in the table. Run the simulation 100 times and tweak the ranges to bring the probability for survival to 60%-80%
4. Create a more difficult version of the dungeon, where the survival rate is 10%-30%, varying the number of players and the boss damage reduction

Figure 7 shows the stage of system analysis at which students have sorted the variables in a table with items, actions, and controls and have developed this further into a CLD. Figure 8 shows how the simulation can be solved. In this example, there are two stocks (the square boxes with curves inside) – player health, which represent party health for all players, and boss health – the enemy's health. Inflows in both stocks represent how much they heal per time unit, and outflow represents how much health they lose per time unit. Player health also has another inflow that represents how much players heal during the fight, which again is dependent on the number of Healers in the party and the healing resources available. Party DPS calculates how much damage the Boss receives and is dependent on the number of players in each class and random factors set in the damage dealer DPS, healer DPS, and tank DPS. The damage dealt by the Boss is also random and is set in the boss DPS and multiplied with a value set in damage reduction (Dred). Damage reduction moderates the damage before it is applied to parties or the boss.

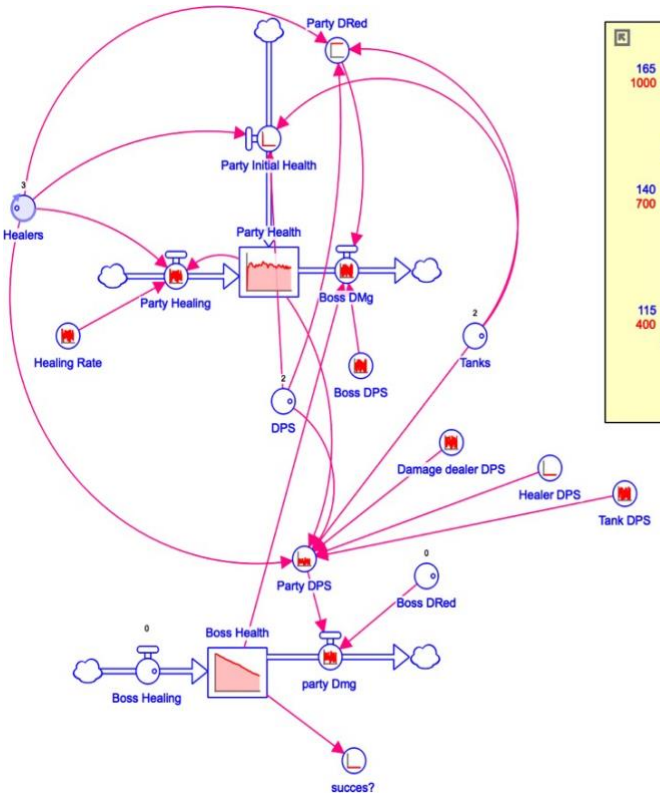
Figure 7

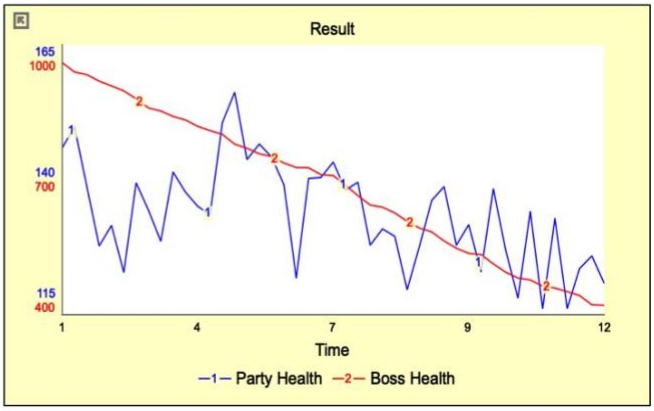


Note: The figure shows the CLD (to the left) and the resources used in the CLD sorted by items, actions, and controls.

Figure 8

The Stella System Dynamic Simulation





Note: Results from this example will be further discussed in 5.3 and the discussion section (part 6).

5.2 Example 2. Trade Route Logistics (Level 3 or 4)

The second example is a part of a bigger simulation game with several systems that must be connected. The game's goal is maximizing profit from cotton production for a whole town (Cottonbog) and choosing the right combination of trade routes to minimize loss from bandit attacks. In this example, town traders sell cotton to three nearby villages, Slagtown, Coalpeak, and Goldford, and bring back gold, food, and other commodities. Currently, the town of Cottonbog has 35 active traders. The road to Slagtown is short and safe. It is guarded by the local garrison and has a very low chance of bandit raids. The road to Coalpeak is long and tiresome, but relatively safe. The road to Goldford is medium in length but there is a greater chance of raids. Each route presents different task values, outlined in Table Z.

Table Z

Task Values and Trade Logistics in Cottonbog Scenario

Town	Travel time (weeks)	Raid chance	Toll price per trip	Trader income per week	Trader income per trip	Gold earned per kg cotton
Slagtown	2	low	500	1000	1500	109.5
Coalpeak	8	medium	0	1000	8000	584
Goldford	4	high	0	1500	6000	438

The assignment asks the students to do the following:

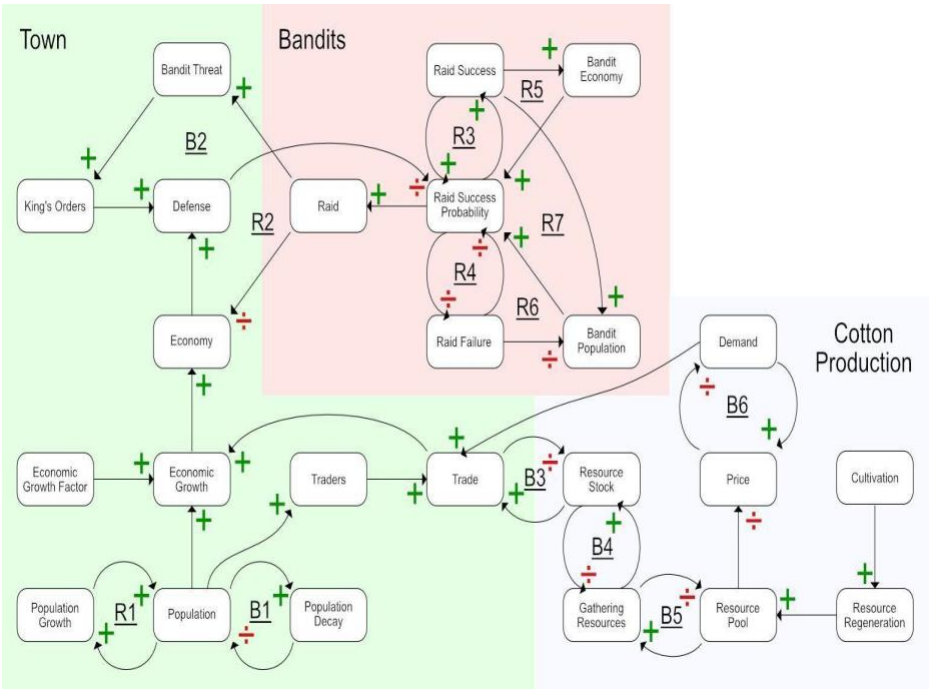
1. Create CLDs, flow charts, and RBPs
2. Create a model of Cottonbog's population, economy dynamics and trading, and cotton production. Run the simulation a number of times (e.g., 100) to establish "the best" trading routes
3. Introduce the bandits as a new system and connect it to the Cottonbog system
4. The king wants to make the roads to Coalpeak and Goldford safer and introduces a tax. Simulate

the tax rate and time needed to do this

5. The king wants to further increase the trade in the region and decrees that Cottonbog needs to trade for 10000 gold per week. Find a solution to this problem and implement it in the system. In this task, the students create a model with three systems: one for cotton production, one for the bandits that raid the traders, and one for Cottonbog. Two things flow through the system: cotton and gold.

The CLD and flow charts are shown in figures 9 and 10. The Stella simulation contains a standard population system for Cottonbog that is dependent on cotton production and trade.

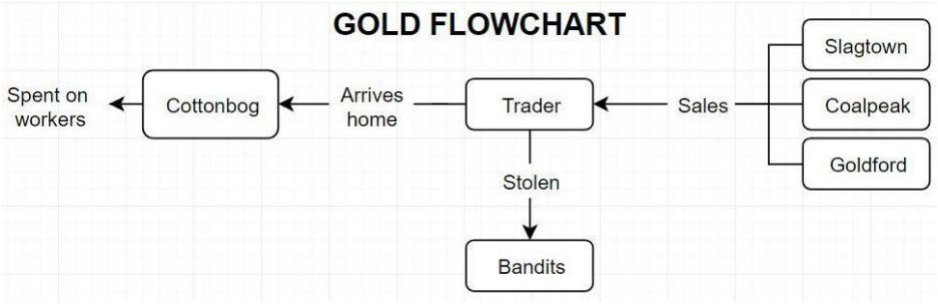
Figure 9 *CLD From System Analysis and Three Interconnected Systems*

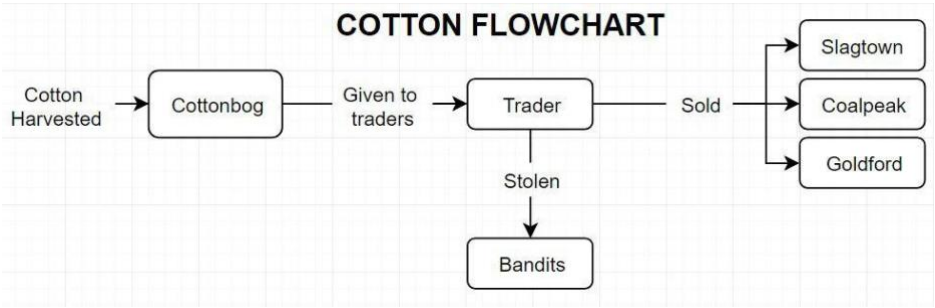


Note: Green (Town) shows the Cottonbog system, pink shows the bandits, and light blue shows cotton production. For example, bandit raids take money from Cottonbog and, consequently, the town's defense is weakened. An increase in raids also creates a bigger threat and increases the likelihood the king will give orders to secure the road.

Figure 10

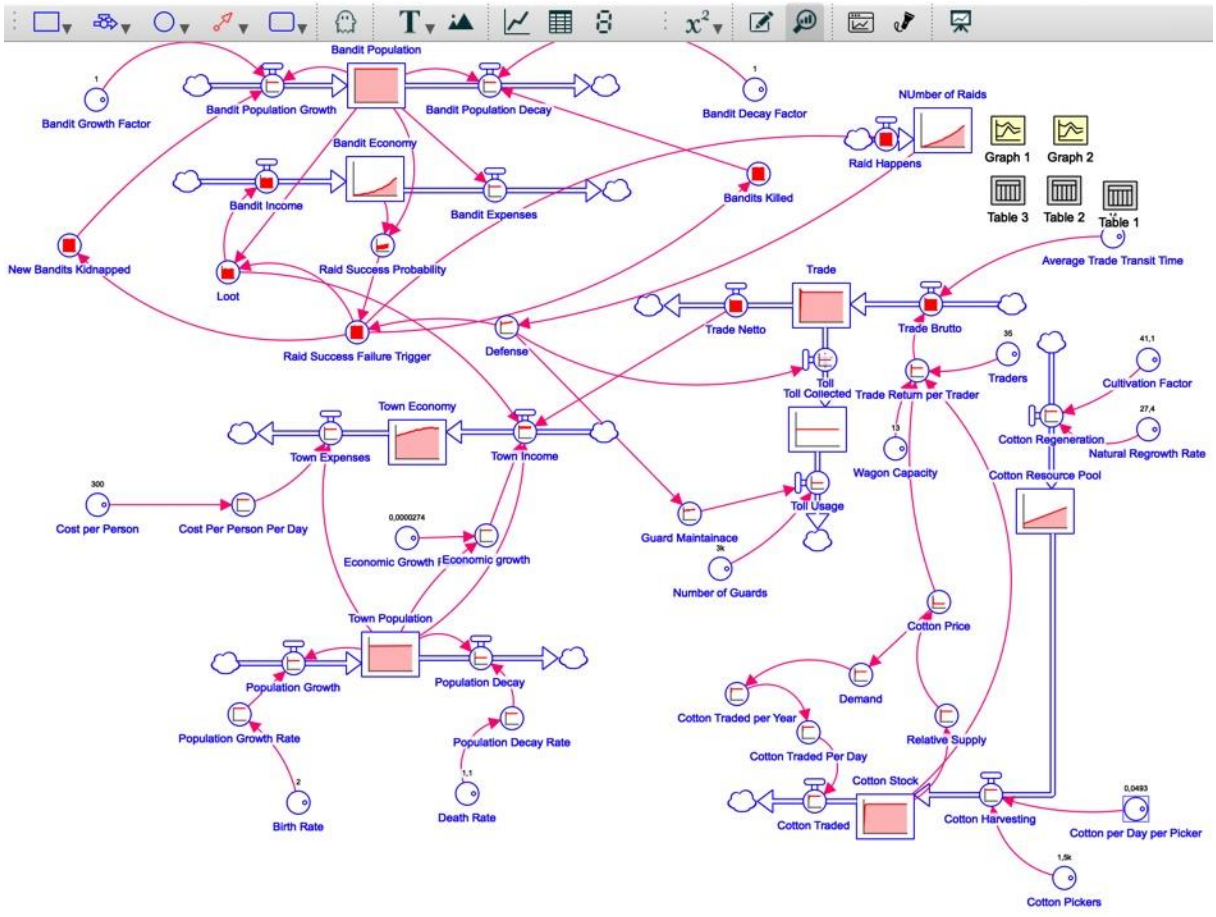
Resource Flowcharts From System Analysis





Note: The flow charts show what flows through the system – namely, cotton and gold.

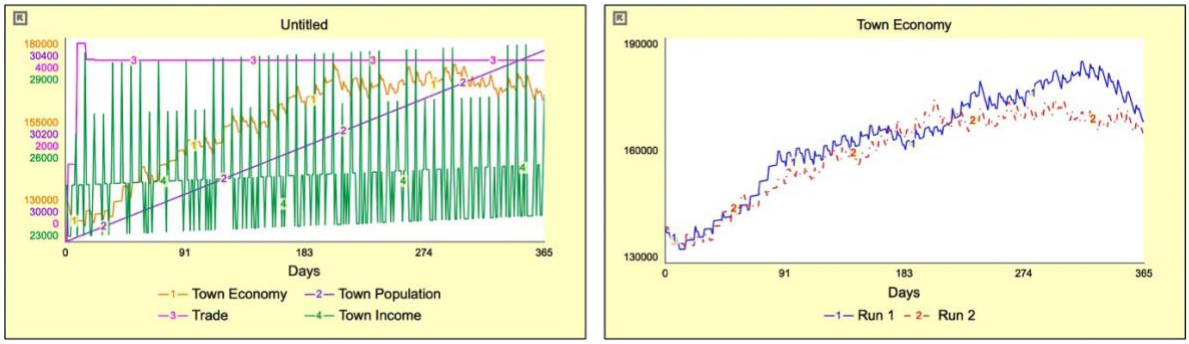
The Bandits system (on the upper left in the Stella model in Figure 11) differs from the Bandit population, and the economy improves only when a raid is successful. However, raids are risky because players often lose bandits during a raid. The number of raids and the success rate are random functions influenced by the number of bandits, the danger of the route, and a defense factor. This assignment is quite large and in many ways resembles how a game idea is developed and its difficulty balanced in creating good gameplay experience. As with the other example, the system analysis students apply is all about understanding how the game idea can be divided into systems, what each system will do, its boundaries, and how the systems can be connected. RBP allows students to analytically predict how the systems will behave over time, and what to expect from the simulations. The model, and the system dynamics simulation, make it possible to simulate different scenarios to build an understanding of how a game behaves under different conditions. The simulations enable the game designer to see how the game design functions when it is played by showing how player agency affects every system in the game and how each system reacts, and if these reactions create the desired gameplay experiences.



Note: In all accumulators (square boxes), curves indicate how the accumulator develops over time.

Figure 11

Stella Model of all Three Systems



Note: The Stella model contains all the three systems in the CLD. The yellow diagrams show system parameters and how they develop over time.

5.3 Critical Evaluation of Student Performance

The system thinking used in the first assignment highlights the usefulness in balancing the game mechanics in a dungeon crawler where different players, with different attributes, encounter an NCP. These encounters can be advanced, with many different systems connected to different player classes, abilities, items, and equipment. Because the simulations in Stella are well-suited to simulating the full range of variables, designers can scale up and down scenarios according to how advanced a game they want to run. This can be done in the initial game design process in the preproduction phase, or in the production or postproduction phase after feedback from playtesting.

In general, system analysis contributes to the students' general understanding of both system thinking and game design. Further, system dynamics and its simulations force students to reflect on how the values in a given task influence gameplay and even design (such as the aggro in example 1, which can be defined in many ways). It also helps students balance the game and provides mathematical equations they can code and use in the game engine. Value of system thinking of course increases with the complexity of the systems.

The three systems in example 2 train students to see how systems influence each other. This assignment extends students' thinking around system dynamics because it involves two completely different societies (town and bandits) that behave differently yet influence each other.

In our experience as educators, introducing system thinking into the game development process forces students to consider more thoroughly what type of gameplay experiences they want in their games and how to implement them, and how a game behaves and changes when parameters are varied. In addition, forcing students to update both system analysis and system dynamics each time changes in the game engine are implemented gives them a complete set of tools to explain their game to others, such as other team members, investors, and sub-contractors.

6. Discussion and Lessons Learned

In the two assignments presented in this article, system thinking is used as a proven academic tool to further understand and develop advanced game design. System analysis and system dynamics methodologies allow students to get a good overview of the systems underpinning their game concept. With these tools, the game concept can be broken into different systems, making it easier to see how the game difficulties will fluctuate and behave, how different systems will interact with each other, and if the gameplay loops create the desired player experience. The students can therefore tweak different aspects of the game idea until it approaches the concept they envisioned. They can take the data that

their system thinking as a whole has produced and improve the game engine prototype they will playtest, rather than iterate on the game engine in the production phase.

By using system dynamics simulations, students can explore a scenario many times while varying the parameters without developing the game – and gain a more thorough and detailed understanding of how different solutions behave in their games. Over our three years of instruction in game design using system thinking, the class discussions, presentations, and reports have shown that these methodologies provide students a more comprehensive understanding of the problems involved in game design. In the first example, we found this evident in the students' deeper and more thorough discussion and understanding of aggro, healing, and health in a boss encounter and the flows and outcomes it produces. As simulations run in Stella can be stored, students can examine prior results, and therefore can more easily balance the solution they choose.

Class reports also indicate that students using system thinking tools developed a more comprehensive understanding of game balance, as their simulations showed them how small changes in game parameters can have a sizeable impact on game systems -- and ultimately affect player experience. This leads us to the conclusion that the system dynamics simulations students used in their assignments resulted in their more thoroughly understanding how to balance systems in a game and make meaningful gameplay encounters. As a result, students acquired an overall fuller understanding of how games work. Students do still need a basic understanding of traditional game design as a foundation, but system dynamics helps them dive deeper into the game design. The best student groups took the assignment further than needed and experimented by making "smarter healers" through implementing a more sophisticated aggro system. As educators, we learned that it is important to make the cases or assignments open-ended so students can take them as far as they can.

The second example presented above teaches students the same lessons and knowledge as the first, but with a more complicated game design; in this case, they must calibrate – and understand – three systems. While this second assignment is small when compared to a fully developed commercial game, approaching it through systems thinking can help provide an overview of the game (or parts of the game) that make it easier for students to grasp the overall behaviour of the gameplay experience without having to consider every detail of the game at once (Gee, 2007).

Normally, game design calls for a lot of playtesting. Many game design books assert that playtesting by professionals and amateurs alike is the best way for developer to get information about a game's different components (for example, gameplay and meaningful play). In our classroom assignments, system dynamic simulations saved time otherwise spent on playtesting by letting students "test" and document the balance of the game prior to user tests.

System dynamic simulations in Stella also give game designers differential equation-based pseudo code that can be implemented into the game engine with little effort. This, too, makes it easier for the game designers to quickly build a more balanced core version of the game in a game engine that can be playtested.

In general, all students enjoyed the assignments and found them relevant for their education and their game designs (explicitly pointed out by the in the mid-way evaluation). Adams and Dormans (2012) argue that game mechanics, meaning the actions and effects a player can make in a game, are harder to talk about than any other aspect of game design. They are difficult to prototype and test because they often are systemic in nature, meaning the actions, resources, and reactions must be coded, or the designer must use a spreadsheet to get a feel of how each function and relates to the other. This process is not particularly fast or intuitive and neither does it give designers a good understanding of how the game mechanics interact or scale with rising difficulty and increasing challenges. Using system thinking in game design helps solve these difficulties.

So, does system thinking improve the understanding of game design? In general, we think it does. System analysis helps designers understand how cause and effect and feedback loops are connected in their game, and RBP allows them to reflect on how their systems and loops behave over time. System dynamics greatly improves the ability to conduct thorough, detailed testing and balancing of the game before playtesting is carried out. It also makes it possible for the developer to transfer the simulations over to the game engine and execute them whenever the game requires, down to each frame.

7. Conclusion

System thinking is a well-tested and proven tool for analysing, understanding, and testing systems. As Salen and Zimmermann (2004), Gee (2007), and Adams (2014) have pointed out, games are systems; system thinking and game design would appear to be a good match. As game design educators, we have discovered that system thinking encourages students to dig deeper into their designs, which always is a good thing. Further use and testing will, we believe, reveal new areas where system thinking can be used in game design. As we discuss, game development can also be a great learning environment for traditional school subjects. All in all, we can recommend using system thinking for game design and believe it could become the predominant methodology in future game and gamification designs.

8 References

Adams, E. (2014). Fundamentals of game design (3 ed.). Berkeley, Calif.: New Riders.

Adams, E. and Dormans, J. (2012). Game mechanics: advanced game design. Berkeley, Calif: New Riders.

Barrows, H. S., & Tamblyn, R. (1980). *Problem-based learning: An approach to medical education.* New York: Springer.

Blake, B. (2017). The Future of Gaming: Free DLCs and Paid Microtransactions. Retrieved from https://gamerant.com/future-gaming-dlc-microtransactions/

Dewey, J. (1916). *Democracy and education: An introduction to the philosophy of education.* New York: Macmillan.

Dochy, F., Segers, M., van den Bossche, P., & Gijbels, D. (2003). Effects of problem based learning: A meta-analysis. *Learning and Instruction*, *13*(5), 533-568.

Donovan, T. (2010). *Replay: The history of video games*. East Sussex, England: Yellow Ant.

Doyle, N. (2015). Why more open world games is a bad thing. Retrieved from http://gamemoir.com/more-open-world-games-bad-thing/

Egenfeldt-Nielsen, S., Smith, J. H., & Tosca, S. P. (2008). Understanding video games: the essential introduction. New York: Routledge.

Gee, J.P. 2007. Good video games and good learning: Collected essays on video games, learning and literacy. New York: Peter Lang International Academic Publishers.

Haraldsson, H.V. 2004. Introduction to systems thinking and causal loops diagrams. Reports in ecology and environmental engineering. Report 1:2004. 49pp

Haraldsson, H. and Sverdrup, H., 2005, On aspects of systems analysis and dynamics workflow. Proceedings of the systems dynamics society, July 17-21, 2005 International Conference on systems dynamics, Boston, United States of America. 1-10 pages. http://www.systemdynamics.org/conferences/2005/proceed/papers/HARAL310.pdf

Reason, 2001, Handbook of action research: participative inquiry and practice. Reason, Peter, Bradbury, Hilary. London: SAGE. 2001. ISBN 978-0761966456. OCLC 50303325.

Kent, S. L. (2001). The ultimate history of video games: from Pong to Pokémon and beyond--the story behind the craze that touched our lives and changed the world (1st ed. ed.). Roseville, Calif.: Prima.

Lave, J., and Wenger, E. 1991. *Situated learning. Legitimate peripheral participation.* Cambridge: Cambridge University Press.

Maastricht University 2013. *Problem based learning preparatory website.* Retrieved from http://www.umpblprep.nl/

Mora-Cantallops, M., and Sicilia, M.-Á. (2018). MOBA games: A literature review. *Entertainment Computing, 26*, 128-138. doi:10.1016/j.entcom.2018.02.005

Nordby, A., and Karlsen, S. 2014. Teaching 'hardcore science' to arts and design students: Reflections on the development of a basic programming course. *InFormation - Nordic Journal of Art and Research* 2014; Volume 3.(2) p. 129-142

Peterson, S. 2016 The Marketing Power Of DLC. Retrieved from https://www.alistdaily.com/digital/marketing-power-dlc/

Pettersen, R. C. (2005). *Kvalitetslæring i høgere utdanning: Innføring i problem- og praksisbasert didaktikk.* Oslo: Universitetsforlaget.

Pramath. 2016 How Do Developers Create Massive Open Worlds For Exploration? Retrieved from https://gamingbolt.com/how-do-developers-create-massive-open-worlds-for-exploration

Rouse, R. (2005). *Game design: theory & practice*. Plano, Tex.: Wordware publishing.

Salen, K., and Zimmerman, E. (2004). Rules of play: Game design fundamentals. Cambridge, Mass.: MIT Press.
Schell, J. (2008). *The art of game design: A book of lenses*. Amsterdam: Elsevier.
Schiefele, U. (1991). Interest, learning and motivation. Educational Psychologist, 26(3 – 4), 299 – 323.
Senge, P. The Fifth Discipline. The Art and Practice of the Learning Organisation. Century Business, New York. 1990
Sterman, J.D., 2000. Business Dynamics, System Thinking and Modelling for a Complex World. Irwin McGraw-Hill: New York
Sverdrup H. (Ed.), Haraldsson, H., Olafsdottir, A.H., Belyazid, S. Svensson, M., Nordby, A., 2022 System Thinking, System Analysis and System Dynamics: Find out how the world works and then simulate what would happen. 7th revised and redesigned edition. Oplandske Bokforlaget, Hamar, Norge. 330pp. ISBN 978-82-7518-281-2
Tekinbas, K.S., Gresalfi, M., Peppler, K., Santo, R. 2014. *Gaming the system: designing with Gamestar mechanic*. London: The MIT Press.
Wenger, E. (1998). *Communities of practice: Learning, meaning, and identity.* Cambridge: Cambridge University Press.
Wallach, O. (2019). Who Builds Video Game Wikis? Retrieved from https://www.fanbyte.com/features/who-builds-video-game-wikis/