Patterns of Recreating Reality in Games

Branislava Vranić and Valentino Vranić
branislava.vranic@gmail.com, vranic@stuba.sk
Institute of Informatics, Information Systems and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava

There are various ways of how reality is recreated in games. These patterns recur in within a single game and throughout many games. There are contradictory forces that lead to a solution that can be reused. Clearly, these ways of recreating reality in games exhibit the properties of patterns in the Alexandrian sense, that heavily influenced the area of software development. Identifying the patterns of recreating reality in games is important in order to develop entertaining and interesting games that people will enjoy playing. Implementing them can greatly improve the quality of games. Furthermore, the patterns are flexible, thus they can be used in many creative ways. For games to be enjoyable and plausible, the reality within them needs to be recreated. Good practices in achieving this can be captured as patterns in the Alexandrian sense. This paper brings in the idea of patterns of recreating reality in games, the connections between them, how they are implemented, and how to apply them. Five patterns mined in Minecraft and other successful games are presented—*The Presence of an Enjoyable Challenge*, *Preservation of Progress*, *Flexibility of the Laws of Nature*, *Expansion of Inventory Limits*, and *Fast Travel*—along with the relationships between them and examples of where they can be found. A story describing gameplay of Minecraft from a player's perspective has been used to show exact examples of these patterns.

Categories and Subject Descriptors: [**Software and its engineering**] Patterns

General Terms: Patterns

Additional Key Words and Phrases: game patterns, game design, laws of nature, Minecraft, drama patterns

ACM Reference Format:

Vranić, B. and Vranić, V. 2022. Patterns of Recreating Reality in Games. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 29 (October 2022), 12 pages.

1. INTRODUCTION

Games have been around for decades and most of them have been becoming significantly more realistic over the years. The graphics have definitely improved, but so have the game mechanics, providing the players with a more realistic experience. However, games do not perfectly resemble the real world. This is one of the main reasons that games are so widely popular—they let people escape their everyday life, immersing them into a completely different world.

The virtual environment in games is modified in ways that makes it more entertaining. However, it is essential to keep some realistic elements in games in order to make the virtual environment plausible. It is important to note that the word *realistic* is used for demonstrating the fictional reality, and not for removing fantasy, science fiction, or other elements that do not belong to reality. Therefore, the aforementioned *reality* may include beings or objects that cannot be found in the real world, such as monsters or objects with magical powers.

It is important to mention that the reason that games are not entirely realistic is not the difficulty to fully recreate reality. The actual reason is that by being fully realistic, games would end up containing all the tedious and unskippable parts of life, thus defying the very point of playing games—entertainment.

Games are comparable to movies or books because all of them have a setting (place), characters, problem, goal, and plot. The only difference is that in games the player contributes to the story by being able to modify or even create certain parts by playing the game. Movies and books remain unchangeable, while games are interactive.

There are various ways of how reality is recreated in games. These *game patterns* recur both within a single game and throughout many games. There are contradictory forces that lead to a solution that can be reused. Clearly, these ways of recreating reality in games exhibit the properties of patterns in the Alexandrian sense [Alexander(1979)], that heavily influenced the area of software development.

Reality is recreated in a way that makes the games feel as real as possible, but without making them boring to play. As a consequence, some rather unrealistic elements remain in every game. Despite this, most people that play games do not seem to be bothered by this; certain unrealistic elements in games have become “normal.”

Identifying the patterns of recreating reality in games is important in order to develop entertaining and interesting games that people will enjoy playing. Implementing them can greatly improve the quality of games. Furthermore, the patterns are flexible, thus they can be used in many creative ways.

The rest of the paper is structured as follows. Section 2 briefly positions patterns in game development. Section 3 introduces the actual game patterns that have been mined in the work described in this paper. Section 4 analyzes the connection between the reality in games and real life and the connections the patterns have to the real world. Section 5 discusses some further aspects of applying the patterns of recreating reality in games. Section 6 relates the finding presented here to what others have done. Section 7 concludes the paper.

2. MAKING GAMES WITH PATTERNS

Several types of patterns found in games have already been identified, focusing on specific areas. Kreimeier recognized patterns in game design [Kreimeier(2002)]. These patterns mostly focus on making gameplay more enjoyable, preventing the players from making monotone choices in games, while also ensuring the design is not too complicated for the designer. The patterns address common problems that arise in games, for example the *Weenie* and *Weenie Chain* patterns describe the problem of players losing their sense of direction in the virtual world. Then the *Paper-Rock-Scissors* pattern resolves the issue of the presence of a clearly dominant strategy that would lead to the players practically having only one choice. The *Filter* pattern, on the other hand, focuses on ensuring that the game is not too complex for the designer by limiting how the player can create new objects or edit the already existing ones.

Barney also describes various game patterns in his pattern language for game design [Barney(2020a), Barney(2020b)]. However, the reader has to devise this from the examples as the pattern description does not include the solution. Furthermore, the real intent of the patterns is obscured by the choice of more metaphorical names. Consider *I Could Be Bounded in a Nutshell and Count Myself a King of Infinite Space Published*, *I See Where You Are Going with This Published*, or *It All Depends On How You Look At It*. Apart from the patterns themselves, Barney explained at length the process of devising these patterns, as his idea is that each designer or design team should devise their own pattern languages as a design tool [Barney(2020a)], which is in accordance with Alexander’s viewpoint [Alexander et al.(1977)].

Game programming patterns, identified by Nystrom [Nystrom(2014)], are meant to make coding more efficient and clear. They are also meant to make code flexible, resulting in the games being easier to modify or update. Nystrom distinguishes between sequencing, behavioral, decoupling, and optimization patterns. The nature of these patterns and even their classification is close to the famous, generally applicable Gang of Four (GoF) design patterns [Gamma et al.(1995)]. They are actually generally applicable as much as GoF patterns are, which is obvious from their names: *Double Buffer*, *Update Method*, *Type Object*, *Event Queue*, *Object Pool*, etc.

The only pattern name that appears to be game specific is *Game Loop*, but this pattern actually describes a generally applicable event loop.

Thus, while there has been some research done on patterns in games from several perspectives, it doesn't seem there has been any on how reality is recreated in games.

3. A NEW (MINECRAFT) WORLD

Picture a game—a new world in Minecraft, one of the most popular video games. A new survival world is created. The game difficulty is set to "normal" (see Section 3.1).

The player enters a seemingly normal world that's made entirely out of blocks. They look around and start hitting a tree chunk until it breaks. The block of wood breaks, but the rest of the tree remains floating above the ground. The player then breaks those as well and moves on to building a shelter—a wooden house. They place rows of blocks making walls and then place blocks above themselves creating a ceiling. Then they arrange six blocks of wooden planks onto a crafting table, creating a door (see Section 3.2).

The night comes and there are monsters outside—the player has to wait out the night and there is not much to do. However, as soon as the night ends, they can go outside again and manage to find a few sheep to kill in order to obtain some wool. They also come across a desert while exploring. The blocks of sand fall to the ground as they remove the blocks that were under the sand (see Section 3.2).

The sun is setting and the player is now at home melting the sand into glass to create windows. Having gathered wool, they are able to create a bed, thus they can sleep through the night, allowing them to skip through it as if it were seconds (see Section 3.2).

The player wakes up and goes exploring. They end up finding a big cave. They are able to gather a lot of resources from there, carrying them in their inventory. Soon, their inventory becomes filled with cobblestone and they need to drop some to be able to pick up more useful materials (see Section 3.4).

Eventually, however, the player gets surrounded by monsters in the cave and dies dropping all the gathered resources. Subsequently they respawn in their house next to their bed (see Section 3.3).

The player does not feel happy about the lost items. Despite this, they go back into the cave, finding their items. Eventually they manage to get better tools with which the monsters become easier to fight. Looking for a greater challenge, the player sets the game difficulty to "hard" (see Section 3.1).

A few days have passed and the player encounters a weird-looking tall monster with purple eyes—an enderman. After being stared at, the enderman attacks the player and keeps teleporting onto them. The player manages to kill the enderman, which drops an ender pearl. They throw it and are subsequently teleported to where the ender pearl landed (see Section 3.5).

Figure 1 shows the patterns of recreating reality in games and how they are related to each other. The main patterns point to the leaf patterns they are related to. These connections are described in detail in each section that covers the main patterns. The following parts of this section describe each pattern individually, explaining its connection to the story, the forces, resolution and examples of where else it can be found. The connections to other patterns are also described wherever applicable.

The rest of this section explains each of the patterns that occur in this pattern story. For this, the example based pattern format as proposed by Vranić et al. [Vranić and Vranić(2019)] was used. It starts with an example of the pattern, making it more understandable. Afterwards, the example is generalized. The forces and resolution are presented in general terms. They are followed by positive and negative consequences of the pattern application with an indication how to mitigate the negative ones. Where possible, particular patterns are mentioned. This is followed by further examples of the pattern. The description is concluded by the explanation of the connections to other patterns (not applicable to leaf patterns).

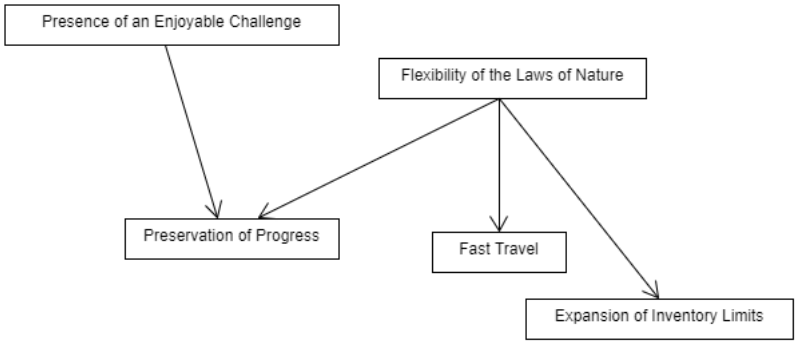

Fig. 1. Patterns of recreating reality in games and how they are related to each other.

3.1 The Presence of an Enjoyable Challenge

As mentioned earlier, Minecraft has a difficulty setting that the player can choose when creating a new world and change it at any point. Therefore if the player does not enjoy fighting monsters, for example, they can simply play the game in “peaceful” difficulty mode that stops the monsters from spawning. Alternatively, there are modes that change the difficulty, allowing the more experienced players to take on bigger challenges.

The player mentioned in the story has at some point fought an enderman and has encountered many monsters in a cave. While fighting the monsters may be a fun challenge for some, others may not like it. Minecraft is a sandbox game, therefore the player should decide for themselves what they want to do in the game.

The game should not be so difficult that players start feeling frustrated or get demotivated, but it also should not be easy enough to complete everything in the first try as this takes away the feeling of accomplishment and presence of a challenge. However, some players do prefer greater challenges than the average games tend to provide, but this would make the game enjoyable for only a very specific audience—for example, people who would rather play a game for its story prefer it not being as challenging to complete.

Forces:

> The game should pose an interesting challenge for the player
> *But* the challenge must not be too difficult to complete.

Resolution:. Let the players choose what game mode or difficulty setting they want to play on. Game modes change up certain aspects of the game, such as what happens if the player dies (i.e., do they lose their items? Is their progress reset completely, partially or not at all?) This allows for the application of the *(*Preservance of Progress) pattern. There may be a game mode that does not preserve the players’ progress, however it is optional. The “classic” or “default” game mode should preserve the players’ progress in accordance to the aforementioned pattern. Usually, three game difficulties (most commonly labeled as “easy,” “medium,” and “hard”) are enough to choose from and not too overwhelming for the players. These difficulties can change simple things only, such as how strong the enemies are or the frequency of their spawning. Even such small differences can offer versatility to the players.

Consequences:

Positive:. Players get to choose the difficulty they wish, making it very likely that thye will enjoy the game.

Negative:. —Certain game difficulties do not allow the players to obtain all the materials the game normally has. An example of this is players not having a way to gather loot obtained by killing monsters in peaceful mode, since they do not spawn. This prevents them from being able to craft certain items, resulting in a less

enjoyable gameplay. There should be an alternative way of accessing monster loot (or any other material that is not available in a certain difficulty mode). For example, loot that is normally gained by killing monsters can be found in some special places, such as desert temples, woodland mansions, etc.

—Players may also not be able to estimate their skill level properly, leading to them picking the wrong difficulty. This can be fixed by letting players change the difficulty even later on in the game.

Examples:

—Among Trees (a sandbox game where the player has to survive in a forest). The game has two modes—Standard, in which there are dangerous animals that the player has to be wary of, and Zen, in which no dangerous animals can spawn and therefore the player can freely explore any area without being in danger.

—Don't Starve Together (a multiplayer survival game). There are three modes, each of them fit for a different playstyle with various ways of preserving progress, leading to some modes being slightly challenging, while one is suited for a more relaxed playstyle.

—Terraria (a 2D sandbox game). There are several modes, for both the world and the player's character, ranging from Journey to Hardcore for the character. Journey mode is the most flexible and allows the player to do things they normally would not be able to, such as duplicating items and controlling the weather, time and enemy spawns. The rest of the modes change merely the difficulty of the game, impacting mostly what items the player drops on death and how strong the enemies are.

Connections to other patterns:. The exact way of moderating the difficulty can take different forms. One of them is described in the *Preservation of Progress* pattern (see Section 3.3).

3.2 Flexibility of the Laws of Nature

The player in Minecraft managed to build their house very easily without any real struggle because the blocks of wood do not fall down on the ground even though there is no block under them. However, sand blocks, on the other hand, do fall down when there is nothing under them.

The wood blocks, and most other blocks in the game, act as Defiers, defying the laws of nature—in this case, gravity. The sand blocks, along with gravel and a few other blocks, act as Compliers because they act in accordance with gravity.

Wood blocks and stone are the most used building blocks, therefore it is important for them to be easy to build with. However, sand or gravel are not important for building, so it is acceptable to use them as Compliers.

Forces:

Game objects should follow the laws of nature in order to be realistic
But the laws of nature can get in the way of doing certain things.

Resolution:. There are objects or elements in the game that follow the laws of nature and some that do not follow the laws of nature. The element that does not follow the laws of nature is a Defier and its role is to make gameplay more enjoyable by making certain things simpler or easier to do. A Defier removes the uninteresting parts by getting rid of realistic elements or by adding unrealistic ones. The element that does follow the laws of nature is a Complier and it is supposed to make the game seem realistic without seriously interfering with the gameplay. The Complier is a realistic element that does not make the gameplay any less enjoyable.

To apply the pattern, firstly, think of the world in the game if it was realistic. Focus on all the things the players would have to do—or even the things they would not be able to do. The Defier allows us to add or remove the part that would make the game uninteresting—or enhance it with an interesting feature. Now, find a Complier. The Complier needs to be in the same category as the Defier (i.e., if the Defier is a game object that defies gravity, the Complier is a game object that complies with gravity). The Complier is usually in a less significant part of the game, its role is only to create balance.

Consequences:

—Positive: The environment of the game that is created by applying this patterns allows players to express their creativity. It also allows for new, interesting elements to be in the game that would otherwise feel out of place. As a result, the game provides players with experiences that would often not be possible in the real world.

—Negative: The game may become less realistic or plausible or it may start feeling inconsistent due to the lack of balance with the rest of the virtual world. There needs to be balance within the entire game when it comes to the laws of nature in order for it to feel consistent; the Complier alone is not enough. A sufficiently acceptable explanation can be used to achieve this. Minecraft is a world made entirely out of blocks, therefore it is not realistic enough for players to question the unusual logic behind the gravity.

3.2.0.1 *Examples:*

—Gravity in *Terraria* (a 2D sandbox game). The trees fall down when they are cut down, thus they act as Compliers. Most other blocks, such as wood or stone blocks, that do not fall if the block under them is removed are Defiers. They allow for efficient building and mining.

—Health bars in many games, such as Minecraft, Terraria and Enter the Gungeon. The Complier is the fact that the player has a "health bar" that represents the health of the player's character. When the bar reaches zero, the character dies. The Defier is the fact that the character does not get weaker or limited from taking certain actions despite being at low health, unlike an injured person in real life.

—Crafting in many games, such as Minecraft, Terraria, Among Trees, Stardew Valley, Don't Starve. Where the players can craft items instantly if they have the materials needed. The fact that the players do not need to actually take time and create the items is the Defier, while the Complier is the fact that they need certain materials to create the items.

—Sleeping in Minecraft and some other survival games which allow the player to skip the night. The Defier is that the players can essentially skip time. The Complier is that the players need a bed in order to sleep.

Connections to other patterns:. The way that the laws of nature can be bent is also described in the *Expansion of Inventory Limits*, *Fast Travel*, and *Preservation of Progress* patterns. In Expansion of Inventory Limits, the Restrictor acts also as the Complier as it sets a consistent limit. The fact that the player can still carry around so many things is the Defier (see Section 3.4).

In *Fast Travel*, the fact that the player can quickly move from place to place acts as the Defier. The Enabler acts as the Complier because it gives an explanation as to why the player can travel or teleport, even though this explanation may sometimes be only sufficiently acceptable.

In *Preservation of Progress*, the Progress Preservers take the form of something that defies the laws of nature, such as setting checkpoints for respawning or resurrecting players. Therefore the Progress Preservers act as Defiers. While not all Progress Threateners are beings that exist in the real world, such as monsters, they comply with the laws of their own nature in that specific world, making them the Compliers.

3.3 Preservation of Progress

Many games tend to have some kind of missions—tasks for the player to complete. The player has to complete these missions in order to get closer to achieving the main goal. Each mission has some *Progress Threateners* that attempt to make the player fail, thus lose some progress. In case that happens, there are checkpoints, some chosen points in the game that the player has previously reached. It could for example be at the end of every mission or after arriving at a special destination. The checkpoints act as *Progress Preservers*.

Even though Minecraft is a sandbox game and therefore does not have any special missions the player must complete in order to advance in the game, it does have Progress Threateners and Progress Preservers that are present throughout all parts of gameplay. Beds act as Progress Preservers because they bring the player back

to their previous location if the player were to die. Monsters act as Progress Threateners due to the fact that they put the player's life in danger, potentially making them drop their items that they are not always able to retrieve.

Forces:

The player should have a way of failing a task,
But it is uninteresting to repeat already completed tasks.

Resolution:. The missions can fail (i.e., if the player dies, gets caught by an enemy, does not complete the task within a time limit, etc.). In that case, the player's progress is set back to a certain checkpoint that enables them to continue.

Consequences:

Positive:. Players do not get too frustrated if they fail and they do not need to worry about doing everything perfectly. This makes the gameplay relaxing enough to be enjoyable, while still being thrilling because players feel good when they perform well in a game.

Negative:. Players may start taking advantage of being able to repeat a certain task as many times as they wish. Dying in the game may also start feeling inconsequential since the players can respawn and keep playing. There needs to be a consequence to failing a task or dying that would discourage the players from taking advantage of the Progress Preservers. In Minecraft, the consequence is that the players lose their inventory whenever they die, discouraging them from taking advantage of the fact that they respawn, while also allowing them to keep playing the game.

Examples:

—Control (a third-person action-adventure game). The Progress Threateners aresecurity guards of an entity who try to stop the protagonist Jesse, that the player controls, from entering specific areas by shooting at the protagonist. The Progress Preservers are checkpoints—in case that the protagonist dies, they respawn and restart the mission, where the objective is to get past the security guards by shooting them.

—Don't Starve Together (a multiplayer survival game). The Progress Threateners are monsters, such as hounds or spiders, that will try to kill the player by attacking them. The Progress Preservers are structures or items that help revive a player—the player becomes a ghost after death and is able to move around the world, but is prevented from other actions in the game. Upon activating a certain structure or using a certain item, the player can be resurrected and keep playing the game.

—Stardew Valley (a role-playing game where the player maintains a farm). The Progress Threateners are monsters found in caves that attack the player. If the player's health bar reaches zero, they find themselves back at home, get a message saying someone found them unconscious and lose a few items and some of their coins. The player being rescued is the Progress Preserver as it allows the player to keep playing the game.

3.4 Expansion of Inventory Limits

In Minecraft, players always need to have various tools or materials at hand for efficiency and practicality. (which involves being able to easily store up to sixty-four blocks of the same type;) The inventory contains thirty-six slots that can store all sorts of in-game items, while one slot can store one stack (usually sixty-four pieces) of the same block. While this would be impossible and unimaginable in real life, Minecraft provides the players with inventory slots that can store virtually all they need.

Forces:

It is interesting and practical to have access to various items at once,
But it is unrealistic to be able to carry more than a couple objects at a time.

Resolution:. Many games provide the player with an inventory that lets them carry various tools, weapons or building materials at once. There are limits to the inventory set by the *Restrictor*. The limits can vary from game to game, but are usually based on limiting how many unique items the players can carry at once. The Restrictor should allow the players to carry enough items, while also making them accessible relatively easily. To achieve this, the inventory space needs to be smaller than the screen, allowing it to be viewed all at once without scrolling.

Consequences:

—Positive: Players have quick access to all the items they need to have at hand instead of having to continuously go back and forth to get their items.

—Negative: As the player progresses in the game, they gather more and more items and the inventory space that once seemed so large is starting to not be enough to let them easily carry all the items they want to have at hand. More advanced players may further expand their inventory either by exchanging it for an in-game currency or obtaining a special material or item that provides them with more inventory space. An example of this are shulker boxes in Minecraft, which take up one inventory slot and can be placed in the world to access their content. Shulker boxes are only obtainable in the End cities, which the player can only explore after having defeated the ender dragon. This ensures that only the more advanced players are able to get the inventory expansion.

Examples:

—Terraria (a 2D sandbox game). The inventory works very similarly to the one in Minecraft—the main difference is that the inventory in Terraria lets players store up to 999 blocks per slot, which is significantly more.

—Stardew Valley (a role-playing game where the player maintains a farm). The inventory starts off with a smaller number of slots, but the player is able to purchase a so-called Backpack, which increases the number of slots to a certain number.

—Kingdom Series games (a franchise of minimalist strategy games where the player builds their kingdom). The inventory is a pouch that only stores a certain amount of coins.

3.5 Fast Travel

While exploring is fun and there are many games that revolve around doing exactly that, there are some places that are simply boring and there is nothing interesting to see or explore. Having to constantly walk through boring areas is tedious. Thus, many open-world games have a way of avoiding this, such as giving the player the ability to teleport. This allows for more dynamic and interesting gameplay.

While it may not be as efficient as in most games, Minecraft also has its method of *Fast Travel*. It is using ender pearls. Although at first they may not seem that useful, they may be used in situations such as when the player needs to get to a place that is separated by a ravine, body of water or some other obstacle that is harder to cross.

Forces:

It is fun to explore,
But it is boring to have to walk through uninteresting places.

Resolution:. The player does not need to walk through the uninteresting places. There is a way to skip them, using an *Enabler* that allows for fast travel to be possible. This can be done either by giving the player the ability to teleport or modifying the uninteresting places. Modifying the places can be done in two ways:

—Make the uninteresting places smaller. This is acceptable in case some sort of transition border, such as a forest, grass field, or mountains between settlements, is present.

—Make the uninteresting places useful and interesting by adding some resources unique to those areas that the player can gather there.

Consequences:

—Positive: Players are provided with an efficient way of traveling. This provides them with a more dynamic and entertaining gameplay.

—Negative: Constantly teleporting from place to place may make the gameplay feel less realistic by making the player lose awareness of the actual distance of the places. Additionally, they will no longer have much of a reason to walk instead of teleporting. There needs to be a consequence for using teleporting that would discourage players from using it all the time, such as having a limited amount of uses within a certain period of time or needing to use a special item to teleport. For example, the player uses up the ender pearl when teleporting. Another example of this are special potions in Terraria that teleport the player to a certain location. These potions are consumed when used, making the player use teleporting more wisely.

3.5.0.1 *Examples:*

—Genshin Impact (an action role-playing game). There are certain waypoints in the form of structures that the player can teleport to. This is under the condition that the structure has been previously activated by the player.

—Stardew Valley (a role-playing game where the player maintains a farm). After completing a certain task, the player unlocks minecarts that allow the player to select a destination and instantly transport to that place.

—Skyrim (an action role-playing game). The distances between settlements and other interesting areas are smaller than they would be in a real world.

4. WHAT REALITY REALLY IS

Having analyzed the patterns of how reality is recreated in games, it becomes apparent that some of them may even be applicable to real life. It is also important to distinguish between reality and a person's perspective. For example, how fast time passes—sometimes we feel as if time was passing more slowly than it actually is. On the other hand, there are also situations in which we do not perceive time, such as when we are asleep or unconscious. There are nights when we fall asleep and feel as if the night had passed quite literally in the blink of an eye. This is also what happens when players sleep in Minecraft—the night passes within a second.

The *Expansion of Inventory Limits* pattern could be applied to real life as well. Most games do not display the player's character with a bag or backpack, yet it is the closest thing comparable to an in-game inventory. One of the games that provides the player with an inventory even refers to the inventory expansions it offers as a backpack, addressing the similarity. The forces that are present in this pattern that is applied to games are also present in real life. The resolution in real life is using a backpack to store the items we need. Meanwhile the resolution in this pattern is giving the player an inventory that also has certain limits. Despite working differently, there is a certain *Restrictor* to both of them—in real life, it is the volume of the items, while in games, it is mostly the number of unique objects and situationally the number of identical objects (see Section 3.4).

Fast Travel is another pattern that is applicable to real life. The world is large and humans love exploring not only in games, but also in real life. While there are many interesting places in both worlds, there are also many uninteresting ones that do not have anything amusing to explore. Even though teleporting is impossible in the real world, that does not mean patterns of fast travel cannot be found in real life. Humans have created various types of transport, such as cars or airplanes, solely for the purpose of making transport more efficient, enabling for moving faster through uninteresting or currently unimportant places (see Section 3.5). These types of transport act as *Enablers* that allow for Fast Travel to be possible. The Enabler in games is usually teleporting, but it can also take the form of shortening distances between places.

In conclusion, the forces are the same when both of the patterns mentioned above are applied to real life. The resolution involves a certain role—the Restrictor or Enabler. The only difference is that the role in games is either not realistic or merely an unrealistic occurrence explained by a realistic one—such as minecarts in Stardew Valley

when it comes to *Fast Travel*, as the player actually teleports from place to place, despite it being explained as traveling on minecarts.

There are some motives that are present in all of the game patterns. They are potentially highly recurring patterns, which needs to be explored further. One of them is *sufficiently acceptable explanation*: there needs to be an explanation, but the real one is too long or distracting. Sufficiently acceptable explanation is present in case the unrealistic elements or occurrences in games present themselves as realistic ones, such as the aforementioned minecarts in Stardew Valley. In that specific case, traveling on minecarts is the sufficiently acceptable explanation used to explain teleporting.

Everything can be made possible in games, unlike in reality. The explanations given as to why something is possible in games are merely parts of the story. The real explanations are often times too distracting or not interesting. Another example of this are also endermen in Minecraft. From the start, they have been treated as mysterious creatures that possessed some abilities, such as picking up blocks, that other mobs do not. Endermen also possess the unique ability of teleporting certain distances. When killed, they drop ender pearls that allow the players to teleport. Therefore, the only explanation given about this is that players can teleport using ender pearls because those come from endermen which possess the ability of teleporting. This sufficiently acceptable explanation leaves many mysteries, but it also allows for endermen to be unique without the game becoming inconsistent, as the sufficiently acceptable explanation to all the other questions is merely that endermen are mysterious creatures from another dimension.

5. DISCUSSION

In this section, we discuss some further aspects of applying the patterns of recreating reality in games.

5.1 The Cost of a Good Game

It may seem that applying the patterns of recreating reality in games might disproportionally or unenecessarily increase the cost of the games. However, we argue that they are rather simple to implement, while bringing indispensible aspects to games without which the players would probably desert them and move to other games. And it's really easy to move to other games.

Consider *The Presence of an Enjoyable Challenge*. Creating the different difficulties or modes in a game could be as simple as adjusting a few numbers (e.g., the damage enemies deal with when they attack or the time limit given to complete a certain task). Additionally, certain elements could be removed or added to the game based on the difficulty or mode the player had set, which requires adding a boolean variable to the code that would regulate things such as whether or not monsters spawn in the game. This is little work compared to how versatile the game becomes after these changes are implemented.

The more correctly applied patterns there are in a game, the better it becomes. However, applying even just one of these patterns to a game can make a significant difference.

5.2 End-User Development

It is impossible to make a game that would be suitable for everyone. Sometimes it is even hard to tell what the audience really wants to see. Wouldn't it be easier to let the players develop their own game?

Game development requires knowledge, creativity, and time of many professionals. End-user development in games allows players to modify the game to their liking. It gives them the ability to change simple things in simple ways. This way, the developers do not need to worry about what the players may want that much—they merely need to give them the option to decide. This is another way of applying *The Presence of an Enjoyable Challenge*—the players create their own challenge, instead of being limited to a few options they would have to choose from.

An example of this is Minecraft. Minecraft is well-known for offering players many different settings, while also letting them use Java commands to change certain variables in order to change the weather, time of the day, add

specific items to their inventory, etc. As a result, they can change how difficult the game is, as well as adapt the game to fit their own playstyle, allowing them to create the perfect challenge for themselves.

6. RELATED WORK

Some of Barney's game patterns [Barney(2020a), Barney(2020b)] are connected to how reality is recreated in games, even though Barney does not state this explicitly. Thus, *Don't Intellectualize My Pain!* effectively addresses recreating the perception of the health state of the player's character. *The Risk of Knowing You* addresses recreating the sense of empathy based on exposing the player's character to risky situations.

The concept of game patterns is close to that of drama patterns [Vranić and Vranić(2019)]. Drama patterns make drama plays intriguing and appealing the same way game patterns do this for games. There seem to be some similarities on a pattern-to-pattern level. For example, the *Fast Travel* pattern resembles the *Loosely Coupled Situations* [Vranić et al.(2020)] pattern in how it enables to switch quickly between otherwise unrelated or distant places.

Game patterns are generating game worlds in a similar way that Alexander's building architecture patterns [Alexander et al.(1977)] generate the real world. Again, there are similarities on a pattern-to-pattern level, such as the transition border element introduced by the *Fast Travel* pattern that resembles the effects of Alexander's *Entrance Transition* in how it makes the player feel the transition.

Patterns for well-being in life [Iwata et al.(2018), Iwata et al.(2019), Ando et al.(2019)] are about the game of life. However, what is considered enjoyable in gameplay, might be not quite contributing to the well-being in real life. More on how game patterns are related to real life is discussed in the previous section.

7. CONCLUSIONS AND FURTHER WORK

For games to be enjoyable and plausible, the reality within them needs to be recreated. Good practices in achieving this can be captured as patterns in the Alexandrian sense [Alexander(1979)]. This paper brings in the idea of patterns of recreating reality in games, the connections between them, how they are implemented, and how to apply them. Five patterns mined in Minecraft and other successful games have been presented—*The Presence of an Enjoyable Challenge*, *Preservation of Progress*, *Flexibility of the Laws of Nature*, *Expansion of Inventory Limits*, and *Fast Travel*—along with the relationships between them and examples of where they can be found. A story describing gameplay of Minecraft from a player's perspective has been used to show exact examples of these patterns.

The patterns described in this paper represent only a small part of the overall pattern language of recreating reality in games. However, through them, new patterns can be identified. For example, *Flexibility of the Laws of Nature* is a pattern that is present in several other patterns, namely *Fast Travel*, *Expansion of Inventory Limits* and *Preservation of Progress*. It is very likely that there are many more patterns that are yet to be discovered that are connected to this pattern. Some ideas for further patterns may come from drama patterns [Vranić and Vranić(2019), Vranić et al.(2019), Vranić et al.(2020)], from Alexander's building architecture patterns [Alexander et al.(1977)], or even from the fifteen fundamental properties he identified [Alexander(2002)]. Definitely, this is the most prominent direction of further research with respect to this topic.

ACKNOWLEDGMENTS

We would like to thank Mary Lynn Mans for being our shepherd and a member of our writer's workshop group. Our sincere thanks also go to the other members of our writer's workshop group: Rebecca Wirfs-Brock, Jorge L. Ortega-Arjona, and Andrei Brazhuk.

The work reported here was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under grant No. VG 1/0759/19, by the Operational Programme Integrated Infrastructure for the project: Research in the SANET network and possibilities of its further use and development (ITMS code: 313011W988), co-funded by the

European Regional Development Fund (ERDF), and by the Slovak Research and Development Agency under the contract No. APVV-15-0508.

REFERENCES

Christopher Alexander. 1979. *The Timeless Way of Building*. Oxford University Press.

Christopher Alexander. 2002. *The Nature of Order: An Essay on the Art of Building and the Nature of the Universe, Book 1 – The Phenomenon of Life*. The Center for Environmental Structure.

Christopher Alexander, Sara Ishikawa, Murray Silverstein, Joaquim Romaguera i Ramió, Max Jacobson, and Ingrid Fiksdahl-King. 1977. *A Pattern Language*. Gustavo Gili.

Hinako Ando, Karin Iwata, Rei Kono, Kohki Ogawa, Takashi Maeno, and Takashi Iba. 2019. Patterns for Well-Being in Life: 9 Patterns for Being in the World. In *Proceedings of 26th Conference on Pattern Languages of Programs, PLoP 2019*. ACM, USA.

Christopher Barney. 2020a. *Pattern Language for Game Design*. CRC Press.

Christopher Barney. 2020b. Pattern Language for Game Design: Pattern Library. `https://patternlanguageforgamedesign.com/PatternLibraryApp/PatternLibrary/`.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Karin Iwata, Hinako Ando, and Takashi Iba. 2019. Patterns for Well-Being in Life: 9 Patterns for Being with Others. In *Proceedings of 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019*. ACM, Kloster Irsee in Bavaria, Germany.

Karin Iwata, Hinako Ando, Yuki Kawabe, Takashi Maeno, and Takashi Iba. 2018. Patterns for Well-Being in Life: Supporting Life Design Based on 4 Factors of Happiness. In *Proceedings of the 25th Conference on Pattern Languages of Programs, PLoP 2018*. ACM, Portland, Oregon, USA.

Bernd Kreimeier. 2002. The Case For Game Design Patterns. *Game Developer* (Oct 2002).

Robert Nystrom. 2014. *Game Programming Patterns*. Genever Benning. `http://gameprogrammingpatterns.com/contents.html`.

Aleksandra Vranić, Valentino Vranić, and Branislava Vranić. 2019. Drama Patterns: Seeing the Patterns from Within. In *Proceedings of the 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019*. ACM, Irsee, Germany.

Valentino Vranić and Aleksandra Vranić. 2019. Drama Patterns: Extracting and Reusing the Essence of Drama. In *Proceedings of the 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019*. ACM, Irsee, Germany.

Valentino Vranić, Aleksandra Vranić, and Waheedullah Sulaiman Khail. 2020. Growing Organizations with Patterns: Lessons from Drama. In *25th European Conference on Pattern Languages of Programs, EuroPLoP 2020 Online*. ACM, Irsee, Germany. Accepted..