Unlike Roguelike: Inverting traditional progression systems in the context of the roguelike genre.

Number of characters: 74552

Supervisor: Hans-Joachim Backe
Course Title: Making a Devolving Rogue-like Game (Spring 2020)
Course code: 5374990-Spring 2020

Jesper Haderup Eiholt - **jeei@itu.dk**
Anne Christina Elsberg - **aels@itu.dk**
Jeppe Stougaard Faber - **jepf@itu.dk**
Francesco Frassineti - **fraf@itu.dk**
Magnus Wahlers - **mawa@itu.dk**

Abstract

Negative progression is an underexplored concept, within game studies as well as in commercially released games. However, it is a tradition known from old pen and paper games in which characters would age and with that slowly weaken. This paper seeks to explore the inversion of traditional progression systems through prototyping a roguelike with such a system. This is done along with a theoretical outline of traditional progression systems and a study of the roguelike genre itself. Afterwards, we describe the process of implementing such a system and the design challenges that arose as a consequence of inverting a progression system. The prototyping results in the game Relinquish, a roguelike in which the player must atone for their sins by giving up their perks. In the game, the hero starts out powerful and slowly becomes weaker. From this process, we found that negative progression translates well to digital games. Having said that, it can introduce problems of frontloading the learnings in the game. However, the concept of negative progression, in itself, is problematic, as it implies that accumulating power and items is, inherently, the standard way.

Table of Contents

Introduction

It is difficult to say why some types of game design slowly turn into conventions. But sometimes conventions need to be broken in order to create fresh and interesting types of games. Progression in games is an example of such conventions that seem to have reached almost universal acclaim. Is there a need to always progress towards a goal or amass wealth and power to make the player feel accomplished and mighty. Why do games rarely explore the concepts of deterioration, decline or sacrifice?

At the core of the roguelike genre lies progression. Despite being a genre defined by very specific characteristics such as randomness and permadeath, there has been a difference in whether or not roguelikes could have any kind of progression between each session. *Rogue* (Wichman, 1980) itself and popular roguelike *Spelunky* (Mossmouth, 2008) has no progression between sessions, newer roguelikes such as *The Binding of Isaac* (McMillen, 2011) or *Rogue Legacy* (Cellar Door Games, 2013) have molded the genre into having very different progression systems. A specific quality of the roguelike genre is its ability to work as a framework for most genres and gameplay systems while still maintaining its identity. Because of this quality, the roguelike genre seems especially suited for inverting the traditional progression systems.

In this paper, we will explore the concept of negative progression in the context of a roguelike game prototype. We begin with a presentation of our final prototype (see section *An Overview of the Game - Relinquish*). Furthermore, we establish the design parameters and the defining characteristics of the roguelike genre, along with research pertaining to procedurally generated content, and the concept of the implied player and player strategies (see section *Roguelikes - A Literature Study*).

Then, we present our exhaustive design process focusing on developing the prototype, the playtesting, and the changes to the prototype through the process (see section *Developing the Prototype*).

Afterwards, we reflect on our prototype process in relation to our established roguelike characteristics, and the theory pertaining to game design processes often observed in roguelike games (see section *Reflections on the Prototype*). On top of that we have a reflection and discussion on player perception and expectations, as well as the consequences of inverting traditional progressions systems (see section *Reflections on Negative Progression*). This leads to a conclusion where we put the insight from our process into perspective and reflect on the possible change of conventional progressions systems in video games (see section *Conclusion*).

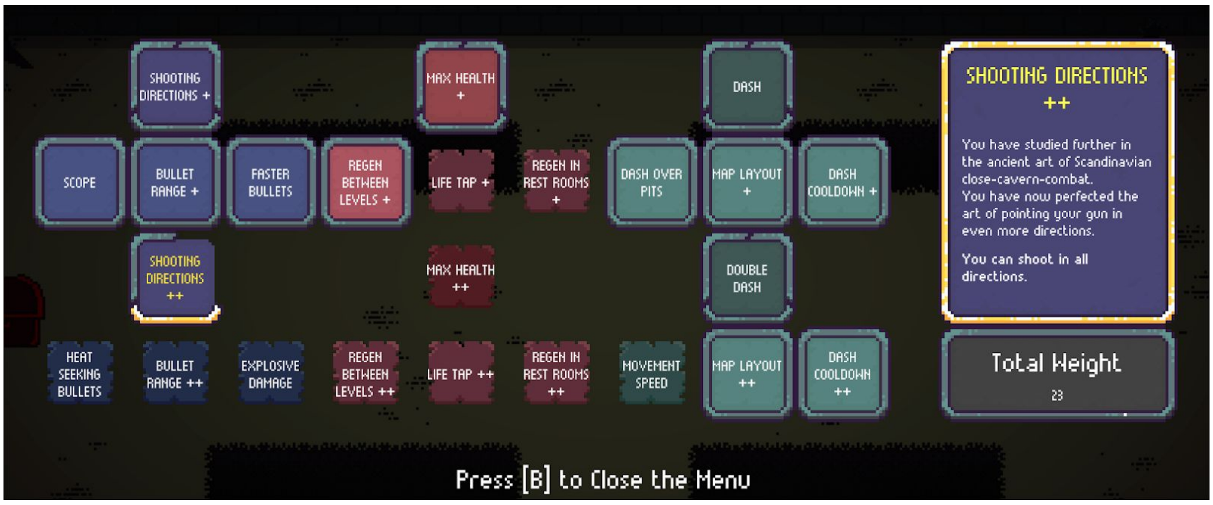
An Overview of the Game - Relinquish

Relinquish is a 2D roguelike with twin-stick shooter gameplay. You play as an experienced dungeon-explorer who has become burdened with the realization that their exploration has wreaked only havoc in their wake. Your primary goal is now to atone for your sins and return to the life you left behind. A mysterious elevator is your only path to the surface, but to ascend you must relinquish your memories of the terrible deeds which you committed and the perks that you have gained as a result.

Link for gameplay video:
https://drive.google.com/open?id=1oludh4avWqUKrxckMo9qqKeyuOT6A3lp

Link for current build:
https://drive.google.com/open?id=1OViXMMjbKB57zB6EnRln7SVR-ibKiiFz

The core gameplay of Relinquish centres around an inverse progression system, where the player is forced to give up various upgrades - represented as weight - to progress through the levels of the game (see Figure 1).



SHOOTING DIRECTIONS ++

You have studied further in the ancient art of Scandinavian close-cavern-combat. You have now perfected the art of pointing your gun in even more directions.

You can shoot in all directions.

Figure 1: A screenshot of the in-game skill-tree system. Initially the player possesses all abilities in the tree and must remove abilities to progress.

To give up upgrades, the player must explore the level to find the various chests scattered throughout (see Figure 2).

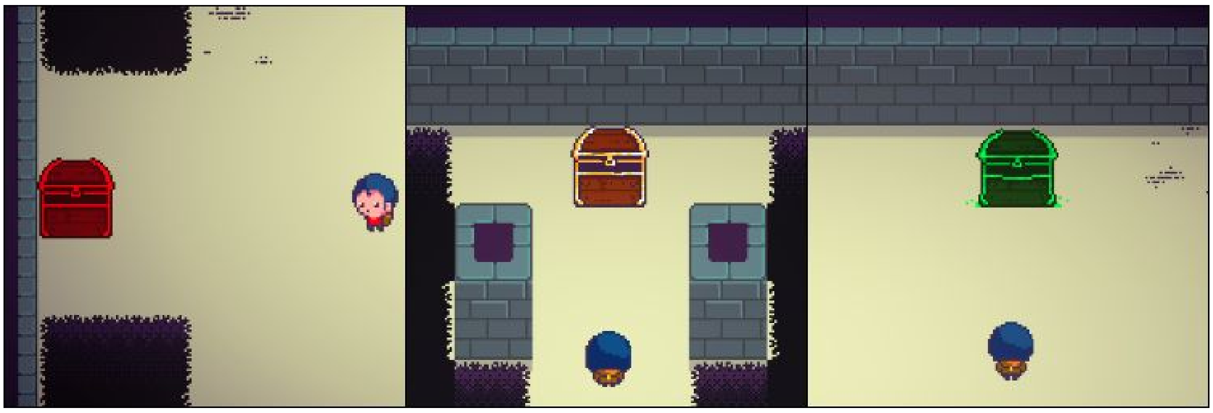


Figure 2: The three types of chests present in the game. The player must find these chests scattered throughout each level to remove abilities.

Once the player has given up enough upgrades, they must return to the elevator which will take them to the next level (see Figure 3).

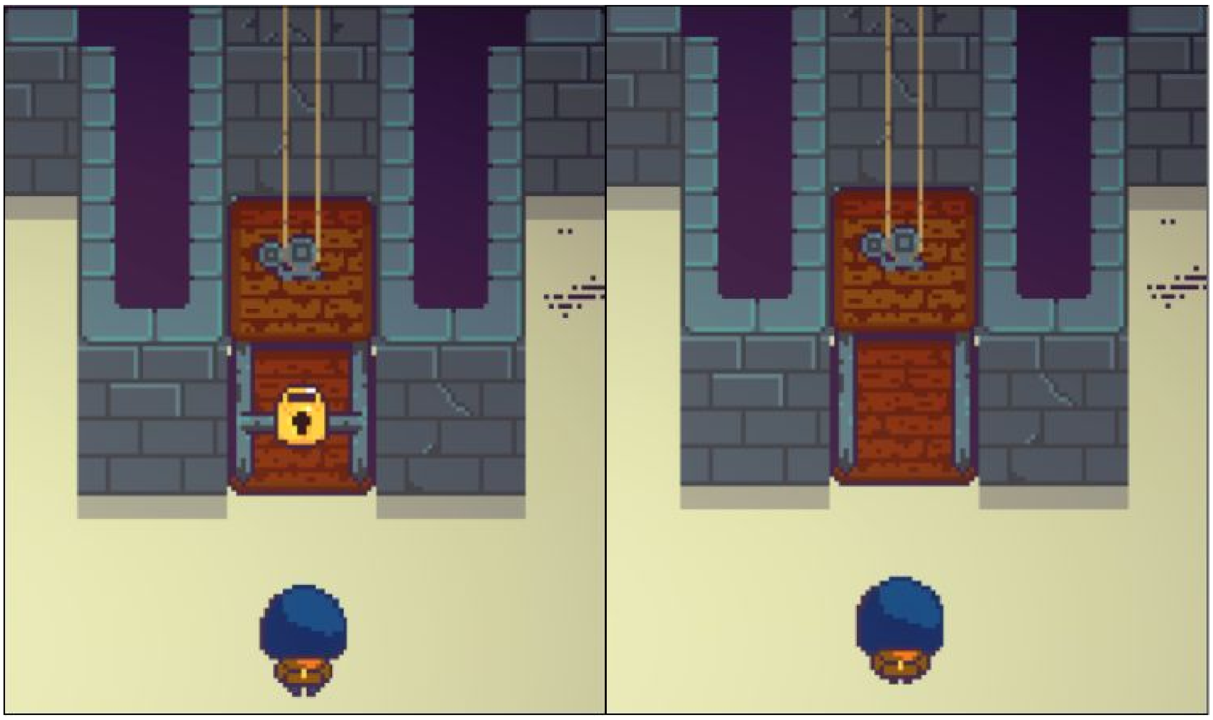


Figure 3: The locked(left) and unlocked (right) elevator. The elevator will only unlock once the player has removed a sufficient amount of abilities.

The primary goal of the game is to complete the final level, which can only be reached once the player has given up all of their abilities. The game implements permanent death.

Various enemies and environmental obstacles block the player's path, some of which are only surmountable if the player has not yet given up certain upgrades (see Figure 4). The primary example of this is the 'pits' - obstacles that the player cannot traverse unless they possess a specific upgrade (see Figure 5).

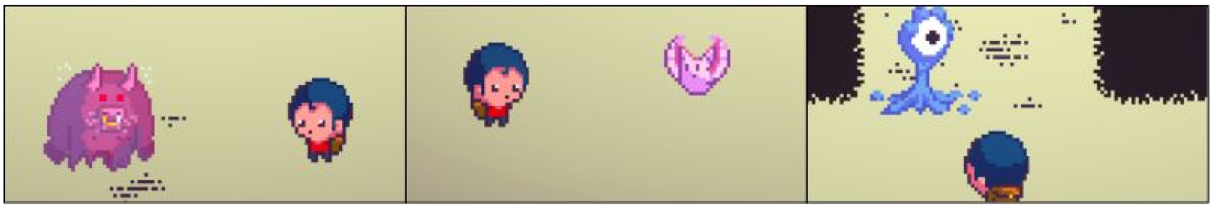


Figure 4: The player is hindered by various enemies appearing in certain rooms.

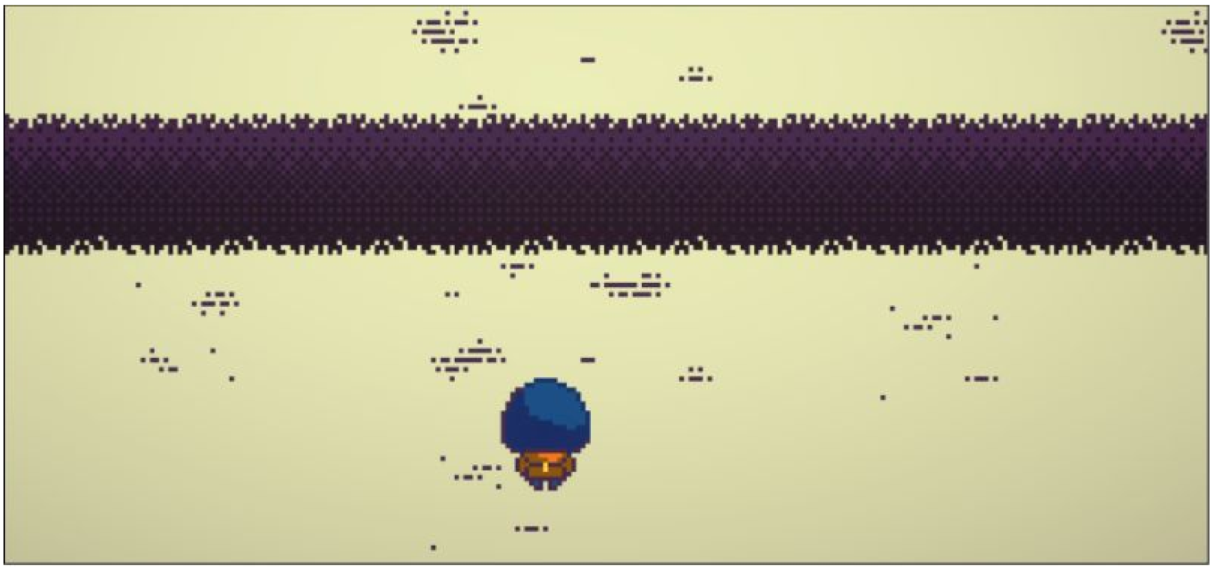


Figure 5: Some rooms contain environmental obstacles such as the pits shown here (the black horizontal hole). To traverse these, the player requires an ability.

Roguelikes and Progression - A Literature Study

In the following section, we will explore the roguelike genre, the evolution of the genre, the attempted definition of the genre as well as our proposed defining characteristics.

The Story of the Roguelike Genre

Rogue (A.I. Design, 1980) was a dungeon crawler video game released in 1980. The genre, dungeon crawling, was based on the concept of navigating through a dungeon looking for treasure, solving different kinds of puzzles, and engaging in prolonged combat with monsters and the like (Ho, Tomitsch & Bednarz, 2016) This was primarily applied as a scenario in tabletop games, specifically the roleplaying platform *Dungeons & Dragons* (Gygax & Arneson, 1974). Game developers had attempted to recreate the genre with games like *pedit5* (Rutherford, 1975) and its spiritual successor *orthanc* (Resch, Kemp & Hagstrom, 1975) with an emphasis on procedural level generation in which the player would have a new dungeon to play when they started the game over (Brewer, 2017). Rogue was the first video game in the dungeon crawler genre to introduce the concept of permadeath in which the player's progression in the game is lost and they have to start over This, along with other traits from the computer versions of dungeon crawlers like procedural generation, would form the inspiration for the sub-genre of dungeon crawlers called *roguelike* which would later transform into a genre of its own (Brewer, 2017).

In the mid-1990s Japanese developer *Chunsoft*, inspired by Rogue, released the first part of the *Mystery Dungeon* series, *Torneko no Daibōken: Fushigi no Dungeon* (Chunsoft, 1993) for the *Super Famicon* home console, which was released in Japan by Nintendo in 1990. This marked the first time a roguelike game had been released for the console market. It was noted that the games, despite their roguelike tendencies, had not implemented permadeath (Brewer, 2017).

No concrete attempts had been made at defining the genre of roguelikes until 2008.

The Berlin Interpretation - A Definition

In 2008 the first International Roguelike Development Conference was conducted in Berlin. At the conference players and developers set out to draft a definition of the genre. This definition would be called The Berlin Interpretation (Berlin Interpretation, 2008) The basis for the definition would rely on canon from what was determined to be the founding games of the genre: *ADOM* (Biskup, 1994), *Angband* (Angband Development Team, 1990), *Crawl* aka *Dungeon Crawl Stone Soup* (DCSS Devteam, 2006), *Nethack* (NetHack DevTeam, 1987), and Rogue. The purpose of the definition was to provide context to the genre without placing any constraints on developers, and as such, the idea was not to exclude games from the genre but to provide parameters for which to indicate how much a game is a roguelike. Games would still be classified as roguelikes when missing a few factors, while on the other hand games with only a few factors would not necessarily be classified as roguelikes. The definition would depend on two general classification factors: high-value factors and low-value factors.

The high-value factors were:

Random environment generation, Permadeath, Turn-based, Grid-based, Non-modal, Complexity, Resource management, Hack'n'slash, and *Exploration and discovery*

The low-value factors were:

Single-player character, Monsters are similar to players, Tactical challenge, ASCII display, Dungeons, and *Numbers*.

Despite the definition allowing for leeway in classifying games as roguelikes, the Berlin Interpretation has drawn its critics mostly based around the idea of the factors being outdated and based on system limitations at the time and not necessarily based on coherent design and narrative choices (Grey, 2013). At the time of the proposed definition, several games had or would soon be released that would revitalise the genre, like *Spelunky* (Mossmouth, 2008), yet these were far removed from several of the factors that were deemed important by the International Roguelike Development Conference (Garda, 2013).

Graphical constraints through the usage of ASCII were only linked to the five founding games of the genre by being the graphical solution at the time (Johnson, 2017). The same can be said about the use of dungeons, as nothing more is stated about the utility of the dungeon in roguelikes, as they are probably a relic from the dungeon crawler genre and have little to no impact on the game experience (Grey, 2013)

Neo-rogue

In the paper, *Neo-Rogue and the essence of roguelikeness* Garda (2013) introduces the term neo-rogue. Neo-rogue is essentially about reinventing the traditions of the original roguelikes. Garda describes how roguelikes have evolved in two waves since Rogue: the first wave was propelled by the release of *Diablo* (Blizzard North, 1997). Diablo replaced the turn-based model of roguelikes with a real-time model in order to introduce more action into the genre. While the game kept the randomly generated dungeons, monsters and loot the game did not include permadeath. Instead, the player was able to save and replay the game using the same character. The second wave introduced what Garda defines as neo-rogue. Garda finds that *Spelunky* (Mossmouth, 2008) started the wave. While Diablo as a concept was about reinventing the genre by introducing action elements, Spelunky and neo-rogues, in general, are rediscovering the essentials of the classical roguelikes. Whereas the traditional roguelikes were modern games, neo-rogue are consciously retro as they are reimaginations of the traditional genres. This makes the possibility of moulding the genre one of the characteristics of neo-rogue games. As Garda puts it:

"Rogue is an almost ideal game regarding the purity of game design (Juul, 2010b) in the sense that an emergent complexity derives from simple rules and mechanics. This tendency is surpassed in neo-rogue by way of postmodern hybridization and remix culture. Neo-rogues are roguelikes which meet the standards of other genres in the shape of: shoot'em-up in Isaac; adventure platformer in Spelunky; or spaceflight simulator in FTL."
(Garda, 2013 p. 8).

The Four Characteristics of A Roguelike

The ever-evolving genre of roguelikes seemed to outgrow the proposed definition of the Berlin Interpretation, which did not offer a proper discussion of the genre in regards to game studies, but more a list of traits and mechanics they thought were prevalent in roguelike games. Based on our expansive playthrough of roguelike games as well at literature pertaining to game design and mechanics which are often utilised in roguelikes, in order to analyse the genre, we have identified the following four characteristics that not only encompass the fundamental design and gameplay mechanics that are inherent to the roguelikes but also allows for experimentation within the genre. These characteristics are not meant to be a definition of roguelikes but will allow us to identify traits that are comparable to play and game design theory, as well as laying the foundation for our own design process. The four characteristics are:

Progression, Rewards, Uncertainty, and *Replayability.*

Progression

The first of the four characteristics is progression. Fullerton (2014) describes the design of progression in games as providing players with smaller milestones while moving towards the goal of a game. These milestones can manifest themselves in different ways. They can be tied to the overall structure of the game such as splitting the game into individual missions or sections. Progression can also be tied to the player character or avatar itself. Zagal & Altizer (2014) study this within games with role-playing elements. They find that character progression in role-playing games traditionally consists of concepts like levelling up through experience and unlocking new abilities. Similar to Fullerton, they describe that character progression systems are gratifying as they provide smaller, but clear, goals for the player. However, not all progression is accumulative.

Zagal & Altizer (2014) describe that some games allow for negative character progression. They find that these systems are most prevalent in pen and paper role-playing games. Negative progression allows characters to decay instead of exclusively becoming stronger. This is usually through character ageing systems in which characters' abilities and attributes slowly deteriorate with age. Negative progression does not need to exist alone and is often used as a tradeoff. Some systems can allow certain aspects of characters to negatively progress while other aspects are improving. For example, a specific weapon can curse the player when wielded.

They made a point of highlighting that this type of progression is underexplored, but find that it can be a useful tool in balancing players that become increasingly powerful in traditional progression systems. They also note that negative progression often originates from the story-world of the game and can bring depth to the characters struggling with weakness.

Skill Trees

While skill trees were not a part of the early roguelike games, they became a part of early computer-role-playing-games such as Diablo (Blizzard North, 1997), and would later appear in games such as Rogue Legacy (Cellar Door Games, 2013) which has a skill tree that is presented as a family manor for the many heroes in the game. Each upgrade either strengthens the player characters or provides some other utility for the player. Skill trees originate from strategy and simulation games, in which they are commonly known as technology trees (or tech trees) (Ghys, 2012).

Heinimäki & Elomaa (2015) study the quality of tech trees in order to provide tools for designers to improve their tech trees as well as a tool for automation of tech tree creation and work as a tool for structuring progression in games. They describe tech trees as being commonly structured like a grid, in which passive or active skills and abilities are unlocked through different progression systems (such as experience points, resources or story progression). This became a popular system letting players specialise their characters in different ways creating unique builds. When introduced in other genres, they are often referred to as talent trees, perks, or skill trees.

In order to assess the quality of a tech tree, Heinimäki & Elomaa (2015) study player-expressed frustration of tech trees online. They find eight main elements that can lead to frustration when interacting with a tech tree. Some relate to having too few or too many skills in the tree, having too few or too many dependencies for advancing the tree. Overall balancing and pacing of requirements to advance are also important considerations, as it can result in technologies/paths/strategies that are dominant strategies. Finally, they describe requirements that are incoherent with the world thematically or historically. It is worth noting that their goal is to automate some of the processes of creating tech trees, which means that their definitions of a good tech tree are affected by this.

Apart from serving as the structure for progression, skill trees are also part of the world-building itself. Ghys (2012) presents the argument that tech trees can convey the technological advancements and ideology of the game world to the point where it can convey some form of technological determinism. He argues that it does so by forcing a set sequence in which advancement can occur, by influencing social changes in history and characterising and linking eras and civilisations. He finds that this is problematic specifically in games that attempt to portray history through technological advancement. A tech tree can, therefore, convey ideologies and be a part of the world-building. World-building through skill trees can be further enhanced through flavour text (Sullivan & Salter, 2017).

The academic discussion of skill trees is still mainly focused on tech trees in strategy and simulation games. While skill trees as character progression seem less explored, the main difference between the two seems to be whether the upgrades apply to one character or to a system.

Rewards

The second characteristic is Rewards. Wang and Sun (2011) present an overview of how reward systems provide positive experiences to players. They state that reward systems' main purpose is to foster intrinsic motivation to keep playing a game.

Wang and Sun propose eight different types of reward systems. It is stated that this list is not exhaustive but it is a starting point for some of the most common systems. These are the systems we deem relevant to roguelikes:

Experience points are systems which reward actions with experience points used for the purpose of levelling up a character.

Item granting systems are systems that reward players with items. What is interesting about items is how their perceived value differs between players and player types. Wang and Sun use Bartle's (1996) player types to explain how some player types will focus on the gameplay changes that the items bring with them, whereas others might use an item to either show off or as a way to facilitate communication with other players. Another characteristic of item granting systems is their ability to encourage exploration because of the ability to scatter items around the game world.

Resources are similar to items, the difference being that they are never used for showing off one’s accomplishment. Rather they are resources that all players should be able to use for more practical use within the game.

Unlocking mechanisms lock content from the player. This is often areas in the game world which are too hard until the player gets a certain upgrade. These systems are also meant to foster curiosity.

Wang and Sun argue that these reward systems' ability to facilitate fun is depending on the player interacting with the system. Different players are motivated by different parts of games. One could find examples of systems which some players would find to be rewards and others punishments.

In a chapter about balancing games, Jesse Schell (2008) lists different types of rewards instead of reward systems. While many of the rewards are directly comparable to Wang and Sun's systems, such as experience given in experience systems, others are added.

Prolonged play is part of most games which includes resources such as life, which ends the game if drained. Rewards which prolong play are giving back the player essential resources which in turn prolongs the play.

E*xpression* refers to rewards which enable players to express themselves. While this is seldom the goal of the game, it can bring players great joy. Furthermore, expression rewards can enable the player to leave a mark on the world.

Completion is often the ultimate reward. Schell argues that the value of this reward lies in how it gives the player closure which is uncommon in players' lives outside of the game world.

Punishment

Rewards are not the only way to provide consequences for player choices. Fullerton (2014) describes that punishments can be used to provoke different player behaviour and challenge in a game but to strive for a balance between rewards and punishments. She finds that punishments are useful for creating tension in games and to make accomplishments feel better in the end. Schell (2008) further elaborates on the types of punishments and different motivations to include them in a game. He finds that they are often reversed rewards.

While Fullerton (2014) and Schell (2008) both suggest prioritising rewards, as they have traditionally been found more useful in controlling player behaviour, they also both encourage implementing punishments in a thoughtful way. Schell observes that implementing consistent, understandable and not random punishments leads to much less sense of loss of control for the player.

Uncertainty

The third characteristic of the roguelike genre is Uncertainty. In *Man, play and games* Roger Caillois (1958/2001) argues that uncertainty is one of six core characteristics of play, as not knowing the outcome of a game is a challenge, and beating that challenge provides enjoyment for the player. In *Uncertainty in Games* Greg Costikyan (2013) proposes 11 sources that offer the player uncertainty in games. Based on games that have been categorised as roguelike the following sources are utilised most often in the genre:

Performative uncertainty is related to the physical performance of controlling a character. Which moves are available to the player, how far does the character jump, how much damage is done when attacked, range of melee attack, etc.

Randomness provides the player uncertainty through outcomes such as level design, enemy appearance, and rewards.

Analytic complexity refers to uncertainty from complex game systems that players must understand in order to make sound decisions.
The sources are not only providing a challenge for the player but also connects uncertainty in general to the other three proposed roguelike characteristics. Uncertainty provides a challenge in regards to rewards by often randomising the items a player can obtain, this in turn also provides replayability as outcomes will often not be the same. As the player progresses through the game they will often rely on reducing uncertainty in order to actually progress, either by unlocking secrets or developing strategies to uncertain elements such as enemy behaviour or level design. As such, the challenge from uncertainty becomes less apparent as the player progresses through the game.

Replayability

The final characteristic of the roguelike is replayability. Replayability or replay value is a common term used in video game vernacular (Frattesi, Griesbach, Leith, Shaffer, & DeWinter, 2011; Roth, Vermeulen, Vorderer, & Klimmt, 2012) used to describe how a game can offer the player several playthroughs. In roguelike, replayability is the sum of the other definable traits.

Progression can either be determined as how much the player progresses in a single play session, or the entire progression until reaching the conclusion of the game. Either way, roguelike offers replayability on both accounts. A player is unlikely to finish a roguelike in their first playthrough, and as such are encouraged to start over and re-play the game. Certain games in the roguelike genre will offer replayability even when having finished the game. This can be manifested in offering branching paths in levels, such as NetHack, different outcomes, alternative solutions to puzzles or combat encounters like in *Void Bastards* (Blue Manchu, 2019), or meta-progression (Bycer, 2013) such as new player characters or harder difficulties as implemented in Rogue Legacy.

Meta-progression is also the centre of rewards and the idea of replayability. Certain rewards will only be unlocked by finishing the game many times. Weapons with more damage, player-character with better skills, items that will ease the player through the levels, etc. As such, the player is encouraged to replay the game in the hopes of unlocking and using said rewards.

If the game offers uncertainty through systems the player is not only afforded replayability through the quality of “extra” content but also through the challenge of not knowing the outcome of the game when utilising said rewards.

The Four Characteristics - In Summary

As mentioned at the start of this section opting for four characteristics in lieu of a proper definition was a way for us to contextualise all the different traits that were linked to roguelikes. None of these characteristics, on their own, are inherent to roguelikes but it is in the interplay between the four that we find the right framework for a design process.

These four are closely connected to design practice, but there are several technical procedures that are connected to the making of roguelikes.

Procedural Content Generation

Procedural Content Generation (PCG) has been a key element of Roguelikes since Rogue where it was used to randomly generate the levels of the dungeon-crawling game. Togelius, Kastbjerg, Schedl & Yannakakis (2011) defined Procedural Generation as the algorithmic creation of game content with limited or indirect user input. PCG can be used for many different reasons such as to automate parts of the workflow for the artists, to create intelligent tools that allow designers to iterate more quickly and more creatively on their prototypes, to generate dynamic content while the game is being played and to adapt such content to players (Shaker, Togelius & Nelson, 2016). Roguelikes commonly use it to generate game levels for every run in order to achieve higher grades of uncertainty and replayability.

Procedurally generated levels need to be playable so the PCG implementation needs to take the design requirements of the game into consideration. Shaker et al (2016) also identified five desirable properties of procedural content generation solutions:

- Speed: how quickly the algorithm generates the content.
- Reliability: how much the content satisfies a given quality criterion.
- Controllability: how easy it is for humans to achieve the desired results by tweaking the parameters of the algorithm.
- Expressivity and Diversity: how diverse is the generated content
- Creativity and Believability: how much the content looks hand-crafted by a human.

In order to generate levels in a short amount of time, roguelikes tend to rely on *Constructive methods* (Shaker et al, 2016). Constructive methods generate the content only once and the output quality is not evaluated, therefore they are very fast. In comparison, *Generate and Test methods* produce new output until a good quality solution is found. This usually provides better results, but also requires much more time to implement.

The most frequently used families of algorithms for generating dungeons in roguelikes that belong to Constructive methods include:

- Space Partitioning,
- Agent-Based approaches, and
- Grammar-based Content Generation.

The following 3 sections are summaries of the descriptions of a subset of algorithms used for PCG in roguelikes taken from Shaker et al (2016). The examples of games listed below are not taken from that source.

Space Partitioning

Space Partitioning algorithms use a top-down approach to subdivide the level into progressively smaller parts (see Figure 6). The result of this family of algorithms tends to be a level with a very structured appearance with no overlapping rooms. Since the room sizes are generated by the algorithm, this approach is not well suited to generate levels with hand-tailored rooms. Also, designers have little control over the final layout of the level since tweaking the parameters is not gonna produce intuitive outcomes.

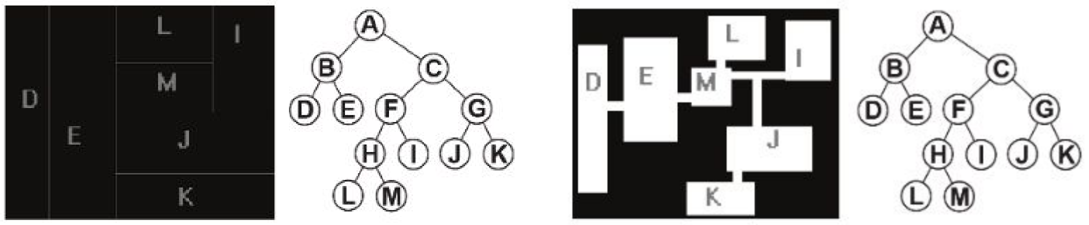


Fig. 6 A visualization of the Space Partitioning approach. Space is split up into regions (left) and rooms are then generated from those regions and subsequently connected (right). Figures from Shaker et al (2016).

Agent-Based Approaches

Agent-Based approaches work in the opposite way compared to the Space Partitioning ones (see Figure 7). Levels are generated by an agent by conceptually digging tunnels and placing rooms in a sequence. This produces a more organic and chaotic dungeon. Tuning the parameters for achieving the desired outcome requires a lot of trial and error. It is possible to place hand-crafted rooms. *The Binding of Isaac* (McMillen, 2011) uses an agent-based approach to generate its levels.

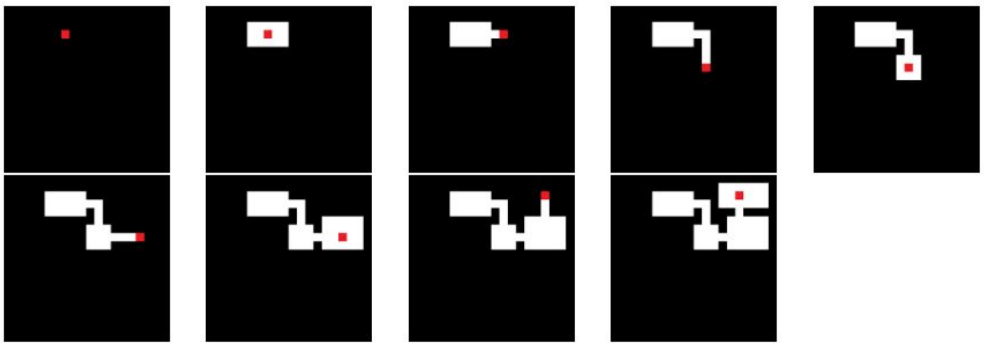


Figure 7: A visualization of the Agent based approach. Abstractly, the red dot represents an agent 'digging' out rooms and connections between them. Figures from Shaker et al (2016).

Grammar-based Content Generation

Grammar-based Content Generation methods use a set of rules (encoded into generative grammars) as a foundation to generate in-game content. Higher levels concepts are built from lower-level ones. One of the possible implementations is the one used in *Dead Cells* (Motion Twin, 2018) to generate its levels (see Figure 8). The designers write a high-level blueprint of a level in the form of a graph where each node represents a random room to be selected from a pool of available rooms of the game according to certain criteria and each edge represents adjacency between two rooms. When the player enters a new level, its layout is generated starting from that blueprint and each node is replaced by a random room according to the criteria specified by the designer (see Figure 9). This method requires much more time to implement, but it's fast, very reliable and controllable, it allows a good level of expressivity and can be used to place hand-crafted rooms, therefore improving believability.

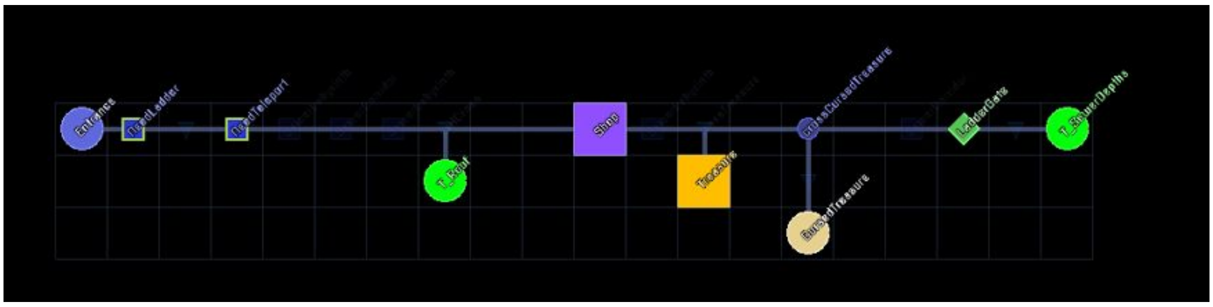


Figure 8: Blueprint of a level in Dead Cells. Here, each shape represents guaranteed elements. The connections between shapes dictate their order in the level. The algorithm then fills in the blanks and creates connections. Figure is a screenshot from worthplayingvideos (2017).

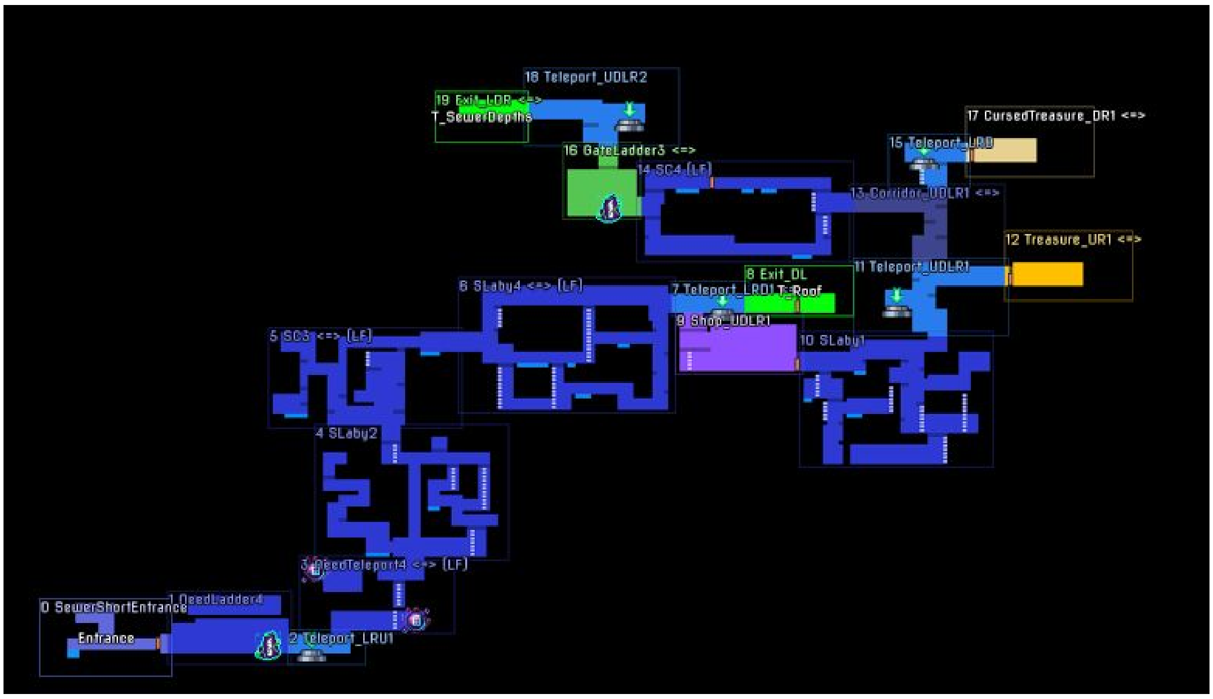


Figure 9. One of the possible generated levels given the previous blueprint. Figure is a screenshot from worthplayingvideos (2017).

Implied Player and Player Strategies

Roguelikes are notorious for their punishing systems and steep difficulty curve. This results in players inventing strategies and tactics for completing the game. While roguelikes can be deliberately vague in how it wants to be played in order to encourage different playstyles, the way a game is designed usually communicates how it wants to be played. This is what Aarseth (2007) refers to as the implied player. He describes the implied player as a role for the player made by the game in which the player is supposed to behave in a specific way. This generally includes playing by the rules and abiding by the restrictions of moral systems and narrative of the game.

Expanding on the concepts of the implied player and play styles, Debus (2018) describes the concept of *implied play styles* in multiplayer games. He describes how games can nudge players towards certain playstyles by the design of mechanics or appearance. However, he argues that play styles can also emerge outside of the game itself in communities surrounding the game.

While games imply how players should play the game, there is often space for experimentation and playstyles while playing abiding by the concept of the implied player. Iacovides, Cox, Avakian & Knoll (2014) have studied strategies players employ when playing single-player and cooperative strategy and puzzle games. They find that, in single-player games, players tend to use five different main strategies. The following are relevant to designing roguelikes:

The first playstyle they find is *Trial & Error*, a playstyle in which deep thought and consideration are discarded in favour of following incidental impulses and trying every possible action in order to progress. This type of strategy often leads to the player not understanding how or why they finally managed to progress.

The second playstyle is *Experimentation* in which players hypothesise cause and effect in order to get a deeper understanding of the systems of the game. This category includes transferring knowledge and logic from outside of the game or from other games and testing it.

The third playstyle is named *Repetition* or *Practice*. This usually includes practising difficult mechanical skills such as aiming in a shooter or jumping in a platform game until they succeed.

They note that players often switch between and employ multiple strategies while playing a game. Often, certain strategies would lead to other strategies.

Research Question

From the literature review, we can observe that negative progression systems have been explored very little and mostly detailing pen and paper games and have not been prevalent within digital games. There is a significant emphasis on progression systems in roguelikes, ranging from having no progression between play sessions and relying entirely on empowering a character through multiple runs. Despite the genre being very moldable, there seems no precedent for making an inverse progression system. This inspired us to study the concept of negative progression through prototyping these systems in a roguelike with the following research question:

How is it possible to invert traditional progression systems and how does it fit into the roguelike genre?

Developing the Prototype

The following section is a run-through of our process of making our roguelike prototype Relinquish. The first part of the section covers the initial ideation phase and decisions in regards to designing for procedurality. This leads to settling on a concept and developing the first prototype. Followed by three testing phases leading to the final prototype and future work.

Ideation Phase

With this project, we wanted to explore aspects of game design that seem under-explored in general. For our initial idea, we wanted to explore how to make a game with a blind protagonist. This led to ideas about a girl with a bat friend slowly turning blind who increasingly would need to rely on echolocation to progress. The game would explore a theme of the character coming to terms with her slowly deteriorating vision. This thought of slowly becoming weaker spawned a new idea based on a campaign from real-time strategy game *Warcraft III: Reign of Chaos* (Blizzard Entertainment, 2002). Warcraft III consists of different campaigns that revolve around certain heroes. Over the course of a campaign, a hero would increase in level and gain new abilities. However, in the expansion *Warcraft III: The Frozen Throne* (Blizzard Entertainment, 2003), one campaign had reversed this progression. The hero in this campaign starts out being the highest level but slowly succumbs to a curse that weakens him. As we believed that this negative type of progression could be explored further and failed to come up with other examples of games using it, we thought it would prove an interesting design challenge.

One thing stood out in this campaign. When levelling up characters, the player is able to choose the order of which skills to unlock, but when the character delevels, this order is simply picked for them. Taking away this agency from the player seemingly disallowed players to create strategies and play with different playstyles, as the game imposed this order onto the player. This observation spawned the idea that players themselves being able to delevel a powerful character could provide for interesting choices for a player.

When inserting this delevelling into a specific genre, a roguelike seemed like a good choice, because it is a moldable genre that allows for many different ways of doing character progression. The genre is great for creating short but punishing play sessions with a lot of replayability.

Designing for Procedurality

As described earlier, a roguelike can tap into many genres and concepts and can vary greatly on the most basic elements such as perspective, mechanics and setting. While it was certain that we wanted to create something best suited for inverting the progression systems of roguelike, we needed to find out what our baseline reference could be. We identified two types of perspectives that we believed could be fitting for a delevelling roguelike. We considered making an angled top-down perspective like the original Rogue or The Binding of Isaac (see Figure 10), or a side-scroller like Rogue Legacy or Spelunky (See Figure 10).

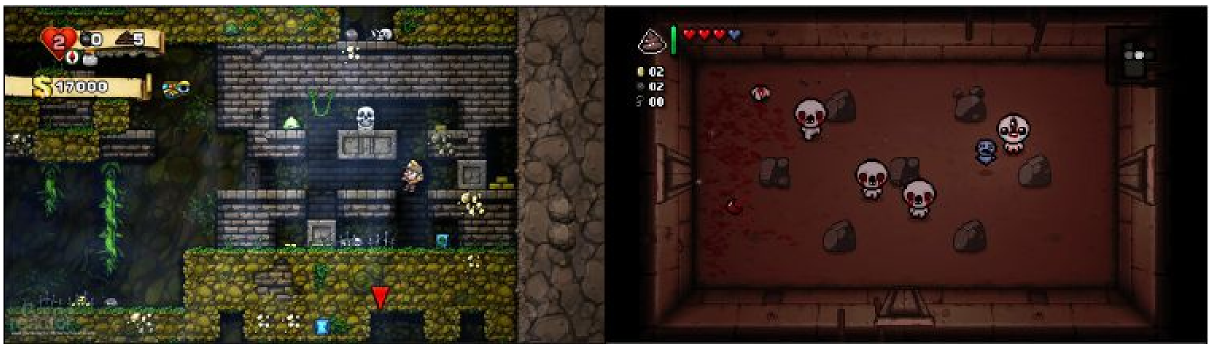


Figure 10: We looked at two possible perspectives inspired by other roguelike games. One approach was a side-scroller perspective like Spelunky (left) or an angled top-down perspective like *The Binding of Isaac: Rebirth*(Nicalis, 2014) (right)

The challenge with side-scrollers is that they likely need to include some form of platforming, which does not seem to fit the concept of delevelling well. In a platform game, players should slowly internalise the movement of the character, which requires time and practice. Adding new abilities and skills to a jump again requires learning to internalise. At the time, we believed it would require too much thought into how level design and the procedural generation could create interesting gameplay. But making a platformer is, in itself a difficult task, because the physics-based nature of platforming needs a lot more fine-tuning to feel just right. Rather, we wanted a setting to allow us to explore the concept of delevelling further without focusing too much on other systems.

Games in the bullet hell genre like The Binding of Isaac and *Enter the Gungeon* (Dodge Roll, 2016), on the other hand, seemed like the perfect setting for delevelling. These games are often harder in the first half of the game, and often rely on an end state in which is easier due to the player has become very powerful from the combination of skills or objects they have picked up. But this, of course, is also connected to individual player skills. Reversing this structure would make for a game which would be easy at the start and difficult at the end. With all of this in mind, we decided to use The Binding of Isaac and Enter the Gungeon as our main sources of inspiration, as we thought these could facilitate the delevelling the best.

Using the floor-structure from these games also allowed us to hand-craft each room on a floor, but use random generation to determine the general layout of a floor. In our prototypes, we even tried to randomly generate the floors, using dice-rolls on a grid to place rooms, and randomly assign paths and detours between the points. We did this to simulate a feeling of randomness in the levels while not yet having the procedurality in place (see Figure 11).

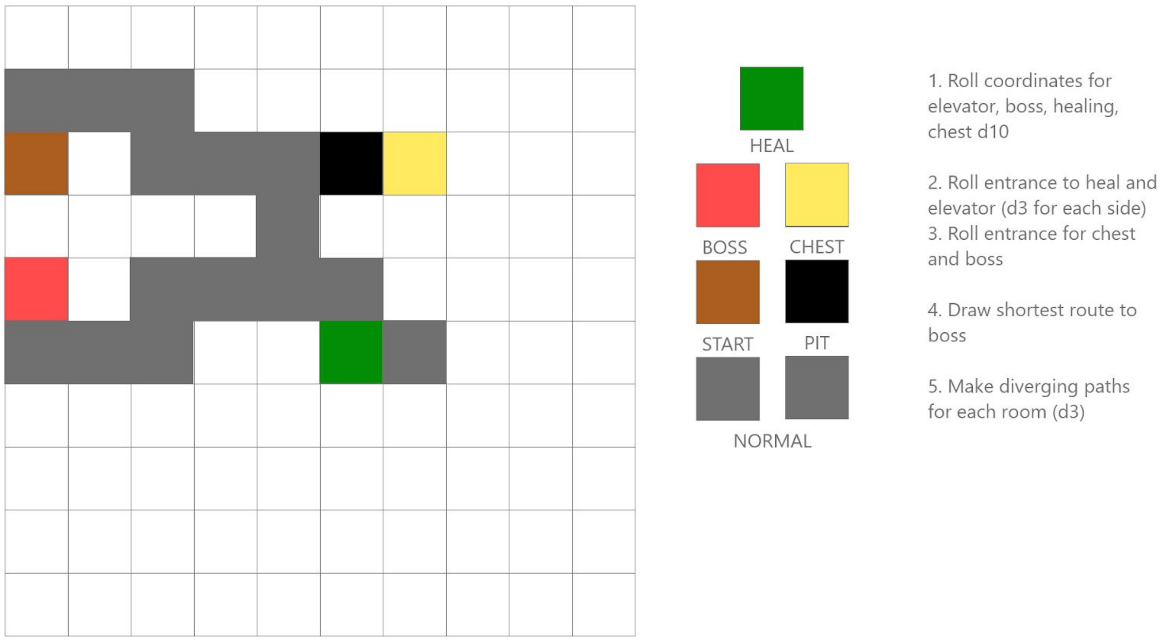

HEAL
BOSS
CHEST
START
PIT
NORMAL
1. Roll coordinates for elevator, boss, healing, chest d10
2. Roll entrance to heal and elevator (d3 for each side)
3. Roll entrance for chest and boss
4. Draw shortest route to boss
5. Make diverging paths for each room (d3)

Figure 11: Example of one level generated by dice-rolling. On the left, the level is presented and on the right, the rules of how it was generated are described.

Planning the Procedural Generation

At this point in the process, we decided on the type of procedural generation that we would use in the game since this would have an effect on large parts of any future implementation.

The general structure of our levels was planned to be a collection of adjacent rooms that connected to each other. The procedural generation algorithm would be responsible for generating randomized levels by randomly placing these pre-built room-templates next to each other (see Figure 12).

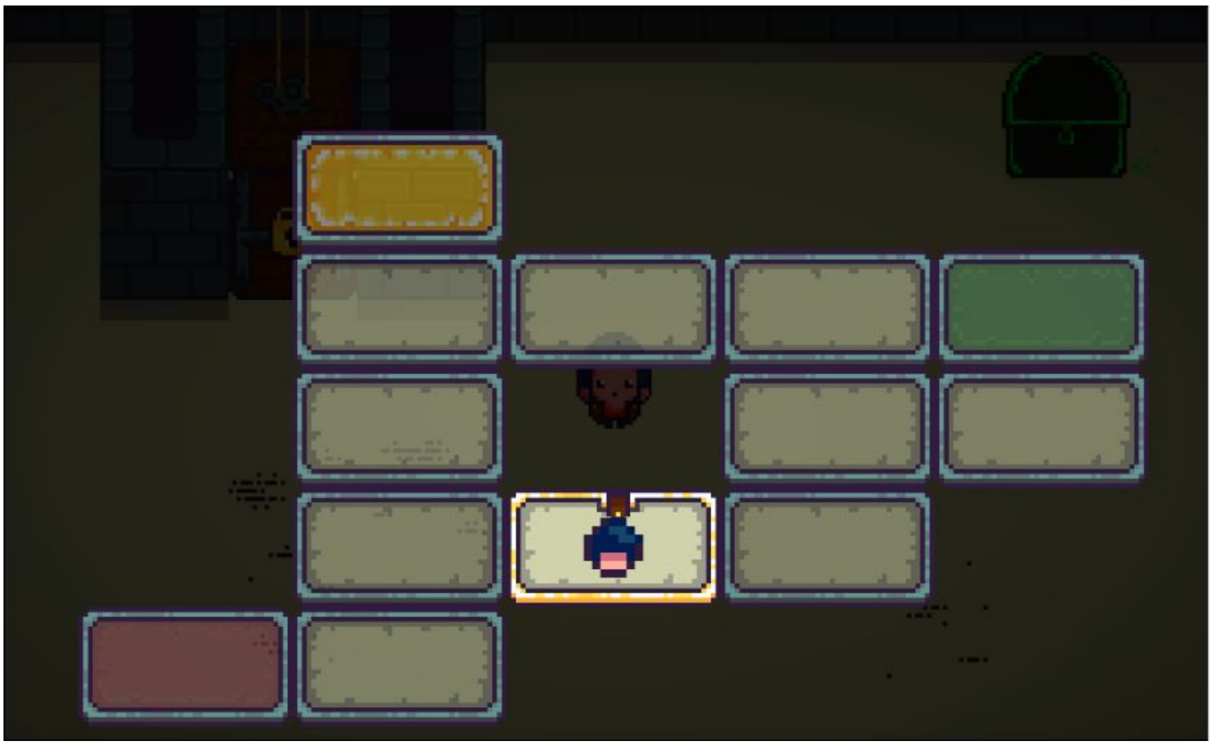


Figure 12: A screenshot of the in-game map in Relinquish. This showcases how levels are constructed by placing rooms adjacent to each other.

Following the research presented previously(see section: *Procedural Content Generation*) we eventually decided to use a grammar-based approach to procedural generation. This choice was mainly grounded in the fact that this approach would allow us to put constraints on the randomization, such as making certain rooms required and enforcing specific layouts (certain rooms can only appear next to other rooms, etc).

These constraints would, in turn, allow us a greater measure of control over the design of the levels themselves. Some of the core considerations in this regard was:

- Basing the floor layout on pre-built rooms allowed us a simple way to prevent impossible or overly difficult challenges to appear in the levels.
- The difficulty of each floor could be varied by constraining the types of room-templates that would appear on a floor or adjusting high-level layout elements, such as the number of rooms that would be placed between two healing/resting rooms.
- Each level could be generated with the player's current abilities in mind, preventing incompletable- or overly difficult level layouts. A prime example of this comes in the form of our considerations on the 'pits': if the player does not have the ability to traverse pits, the algorithm should not place important objectives behind rooms that require this ability.

Settling on the Concept

Having settled the core idea and the genre we wanted to insert it into, we found the first great design challenge. As the player would only delevel, she would necessarily always be at its highest point at the beginning of a game. When having witnessed the chaos that is the odd synergies and chaos of not knowing the functionality of everything in roguelikes such as The Binding of Isaac or *Risk of Rain 2* (Hopoo Games, 2019), even when players have picked up each individual item or skill during the game (see Figure 13). At this point, we were worried that front-loading the abilities would increase the need for elaborate tutorials, but the roguelike genre can allow for much confusion at the beginning of the game, as it is meant to be replayed anyway.

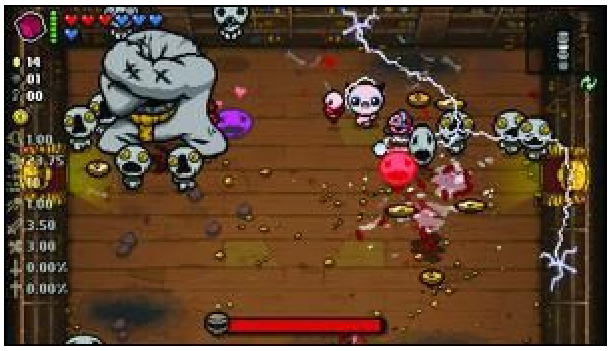


Figure 13: Certain entries in the genre, such as The Binding of Isaac: Rebirth are reliant on very chaotic interactions between their upgrades. Image taken from (Eurogamer, 2017)

Our purpose of making a game with purely negative progression spawned many design challenges concerning traditional progression systems. Killing enemies normally results in enemies dropping some sort of reward, but we needed to decide whether or not this would go against our negative progression. Having monsters drop things incites killing them, so would players want to kill for no gain? Similar to many of the games we were inspired by, we wanted players to explore each level. But having absolutely nothing to gain by exploring our world posed the challenge of making exploration worthwhile in a different way. These challenges spawned the idea of the elevator and the scale system.

Placing chests around each floor can at the very least ensure that players will need to explore more, and adding the healing rooms further invites this (see Figure 14).

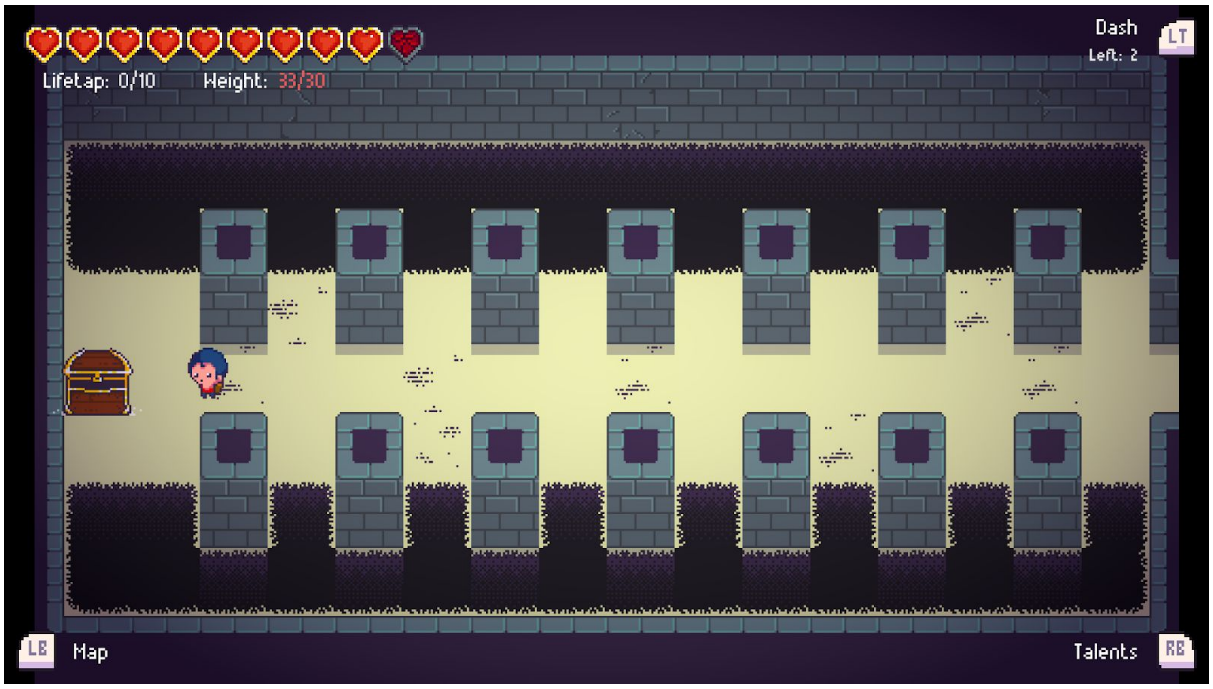


Figure 14: Placing chests around each level was one way of giving incentive to exploration.

Making the length of the game be directly dependent on the delevelling allows players to strategise and prioritise whether or not they want to play more floors but be more powerful for a while, or whether they want to give up as much as possible but play fewer floors.

At the end of the concept phase, we had two main player experience goals. We wanted players to experience a situation in which they come to the realisation that the skill they just gave up would have been useful at this moment - and that this would be a positive feeling. The second feeling would be seeing an enemy which they easily defeated earlier (when they were more powerful) and now fear because they are weaker rather than the enemy being stronger (see Figure 15).

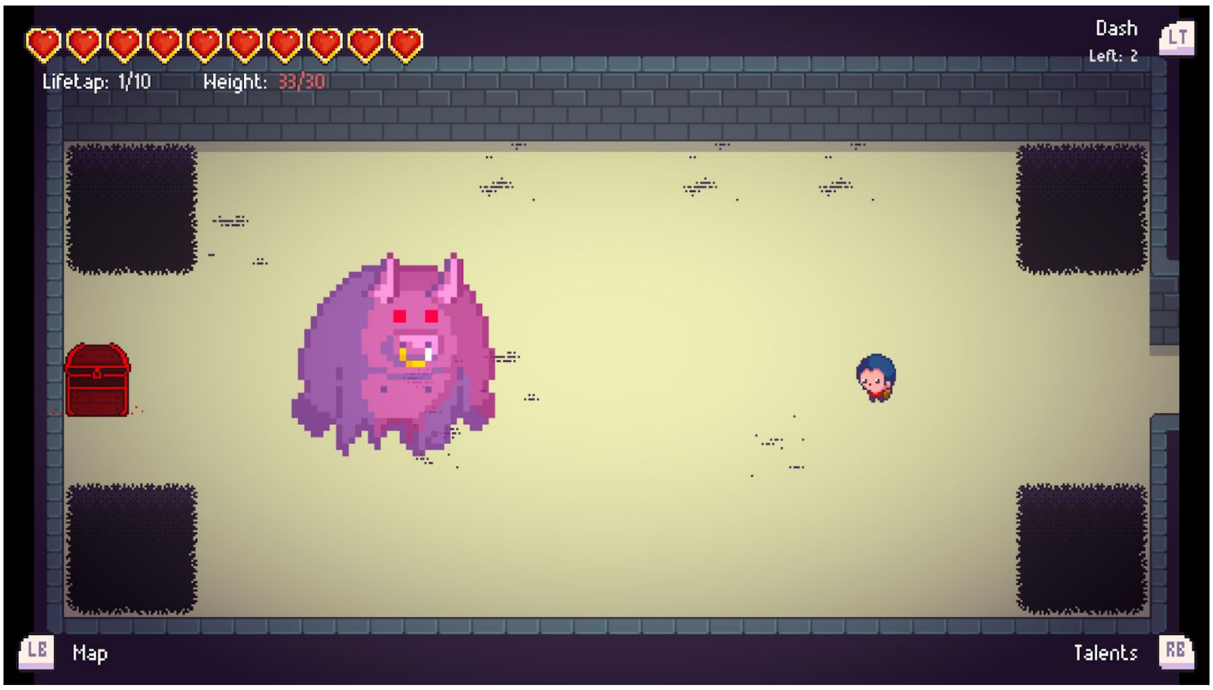


Figure 15: Initially easily-defeatable enemies become progressively harder to overcome as the player gives up more skills.

First Prototype

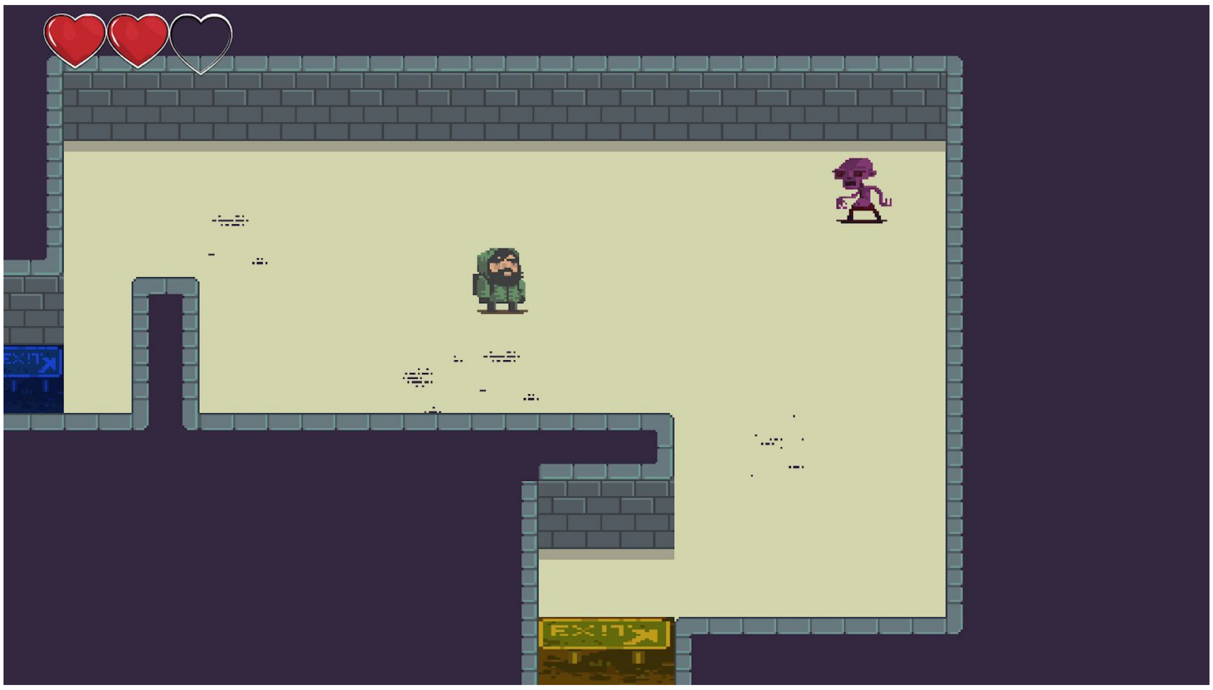


Figure 16: A screenshot from the first prototype build of Relinquish.

With the core concept settled, we made our first prototype (see figure 16). The goal of the prototype was mostly to confirm the potential of the concept. The first prototype included the most essential parts of the concept. The character was able to move, shoot and dash. At this point, the negative progression system was presented as a choice between three random perks, similar to how it is presented in *Nuclear Throne* (Vlambeer, 2015) (see Figure 17), but while it at first glance seemed similar it was in fact tied to a fixed progression. At this point, we had three distinct perks. These included losing health, losing the dash and limiting the shooting directions to eight. The prototype also included the most important parts of each floor. Including the pits, a boss room and the treasure room.

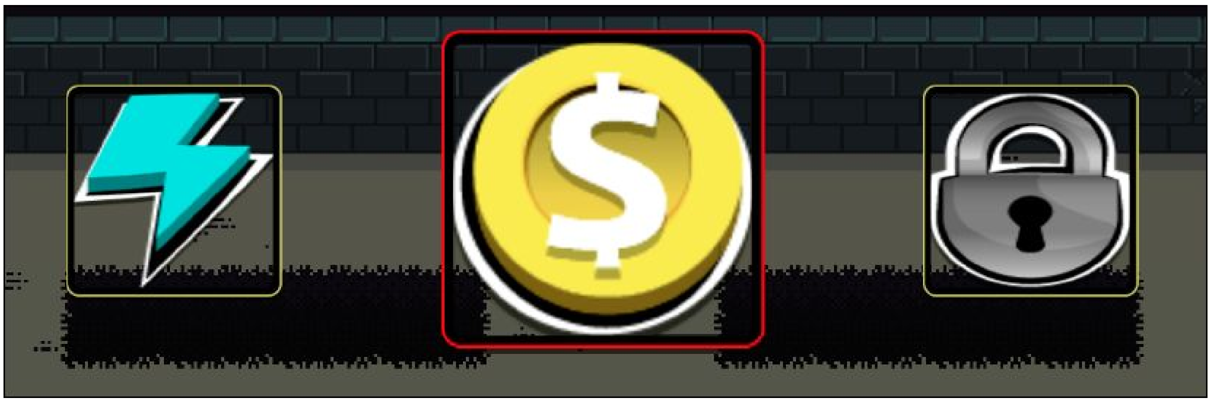


Figure 17: Image of the system in the first prototype that would eventually become the skill-tree system. In this version, the player must remove one of three randomly selected abilities.

Testing the Prototype

While tests with external users can be good in this part of the process (Fullerton, 2014), the prototype was mostly meant for ourselves to evaluate. Although the focus was to enable us to confirm the potential ourselves, we tested this prototype with ten users in a physical and unstructured setting. We tested the concept, the basic game feel and the general usability of the game, which enabled us to find discrepancies in our understanding of the concept and the actual product.

The test confirmed that our concept was enjoyable and testers were generally excited about playing a game with negative progression. As with most early prototypes, many elements needed to be balanced and improved, such as balancing the usefulness of the dash.

It was particularly noticeable how players would talk about our delevelling structure. Nobody would articulate the delevelling as a punishment or that it felt bad. It was as if the premise of the game removed the feeling of being punished whenever the player delevels. And even in this early stage, players were able to articulate and describe different strategies that they would like to employ in our game. Some testers expressed the want to play as a “glass cannon”, which is an aggressive playstyle focusing on the attack while not prioritising abilities that give the player more health. After hearing how well our game was able to facilitate these traditional playstyles, it became a goal to encourage this.

Choosing a Skill Tree

While standard progression introduces character skills slowly, purely negative progression necessarily ends pushing all learning to the beginning of the game. For this reason, we found it necessary to structure the order in which skills are being removed from the player. We added this needed structure through the addition of a skill tree (Figure 18). Skill trees were useful in this situation as they lock the path of progression and give the developers more control of the experience. The problem could, of course, have been solved in various ways, for example by reducing the deleveling to only include decreases in attributes. We chose the skill tree over this approach as we wanted each delevel to feel impactful on the gameplay. Furthermore, this allowed us to design for specific playstyles.

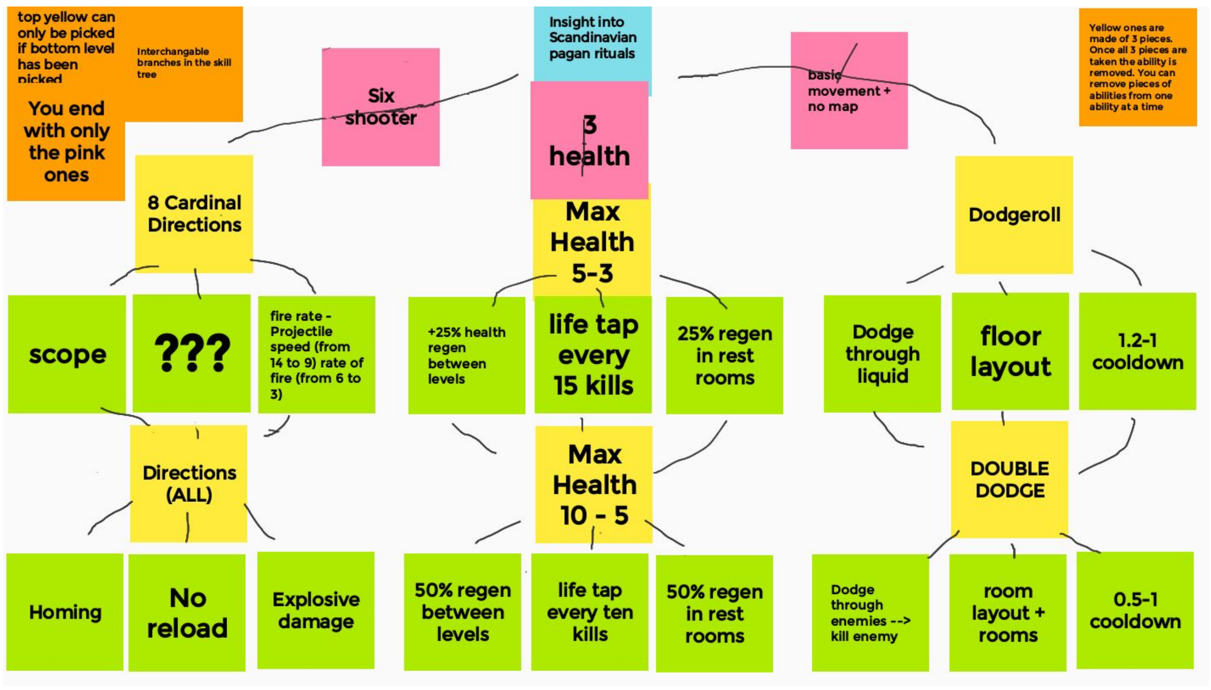

top yellow can only be picked if bottom level has been picked
Interchangable branches in the skill tree
You end with only the pink ones
Six shooter
Insight into Scandinavian pagan rituals
3 health
basic movement + no map
Yellow ones are made of 3 pieces. Once all 3 pieces are taken the ability is removed. You can remove pieces of abilities from one ability at a time
8 Cardinal Directions
Max Health 5-3
Dodgeroll
scope
???
fire rate - Projectile speed (from 14 to 9) rate of fire (from 6 to 3)
+25% health regen between levels
life tap every 15 kills
25% regen in rest rooms
Dodge through liquid
floor layout
1.2-1 cooldown
Directions (ALL)
Max Health 10 - 5
DOUBLE DODGE
Homing
No reload
Explosive damage
50% regen between levels
life tap every ten kills
50% regen in rest rooms
Dodge through enemies --> kill enemy
room layout + rooms
0.5-1 cooldown

Figure 18: One of the design documents showing how we planned the skill-tree system

Second Prototype

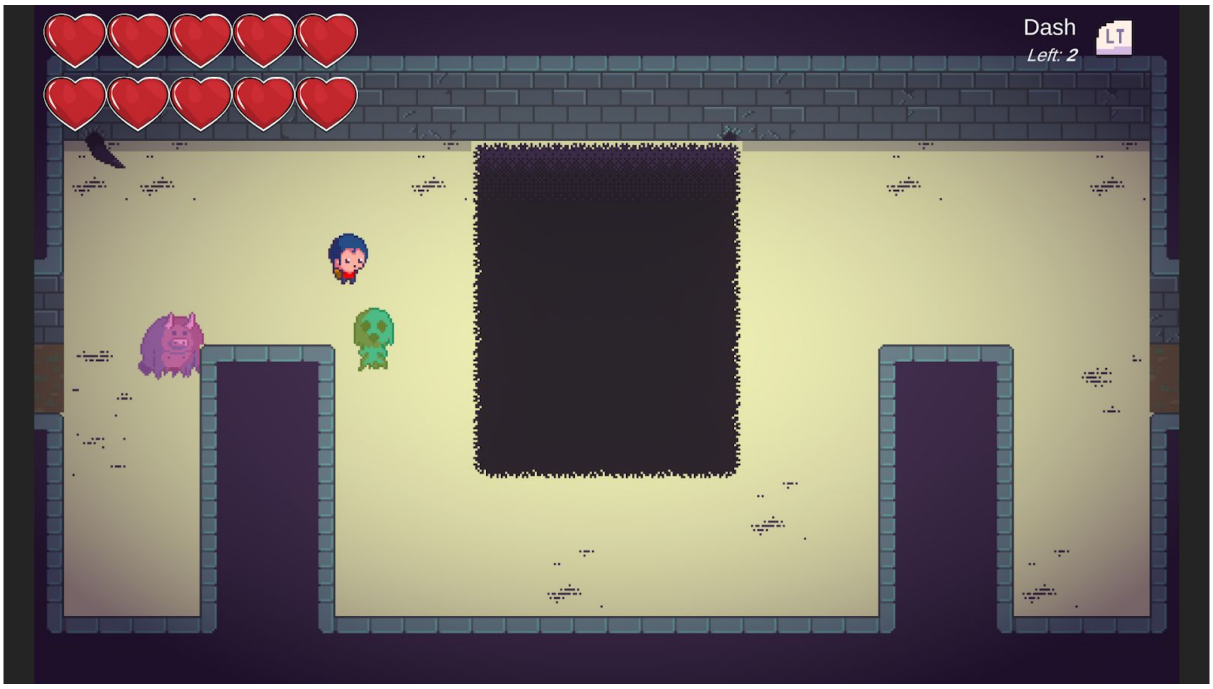


Figure 19: A screenshot from the second prototype build of Relinquish.

In the second prototype, the concept had evolved to include the scale system (see Figure 19). This meant that players were now able to drop up to four perks per floor. Because of this addition, we also needed more perks, which were then introduced with the skill tree. These categories were offensive skills, defensive/healing skills and utility skills (see Figure 20).

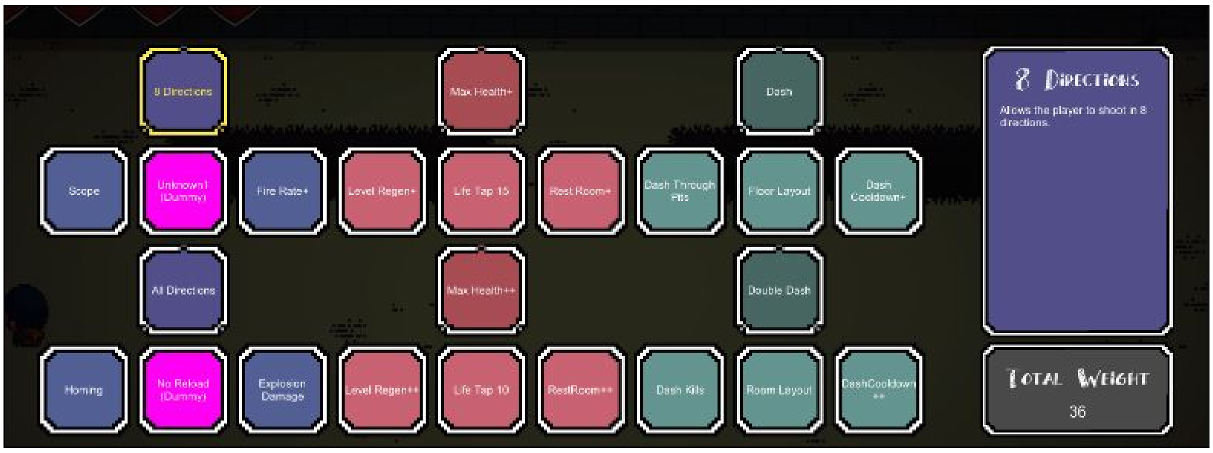


Figure 20: The first draft of the skill-tree system as present in prototype 2.

The perks were designed to fulfil the player experience goal of making the player feel overpowered at first and progressively turn into a more vulnerable state. We recognised that while being overpowered might be pleasing but the lack of challenge might make the game less exciting after a while. Because of this, we tried to intentionally place the strongest perks at the beginning of the game in order to let the player get to more challenging parts of the game more quickly.

Testing the Second Prototype

We wanted to test the new skill tree in order to see how people responded to an expansive set of perks. Our fear was that it would be overwhelming, as well as having imbalances that would affect the gameplay experience. With the added perks we also wanted to find out how the length of the game would impact the overall feel, and lastly, we had yet to implement the narrative setting but decided to tell the players beforehand in order to see if it would enhance their experience.

For this test, we decided to do a more structured test setup, applying many of Fullerton's (2014) suggestions on how to conduct a playtest. She describes a range of different test methods and how to ensure obtaining the best feedback. Something she emphasises is to avoid leading players and encourage testers to think out loud. She provides a list of common interview questions which can be useful at different stages of a playtest.

The testing was not done in a physical setting due to circumstances beyond our control. Our three testers would download a build of the prototype and stream their session on Discord, an online communication tool used primarily for online gaming. They were told the narrative setting and asked to think out loud while playing. After the session they would answer a prepared list of questions, some of the more general was suggested by Fullerton (2014) and others were more specific to suit our game (Appendix I).

The tests went fairly well despite the remote setting. Testers responded well to the added perks. We did observe some hesitation at later levels but this was ascribed to fear from the testers of discarding something useful as they had realised the perks had a bigger impact than they expected at the beginning of the session.

Some incoherence was noted in the defensive/healing tree in which testers had a hard time seeing how the perks interplayed. A perk called Dash Kills, where dashing through enemies killed them were never used by any of the testers.

We observed testers having strong and differing opinions when choosing different strategies throughout the sessions. Some would rely heavily on damage alone, some would discard all the map perks early on only to regret it later, and some would try to do a balanced build with perks from all branches.

One major observation was the lack of difficulty. Even at the later levels, people would kill the bosses and enemies easily.

Finally, we observed something which challenged the concept. The explosive damage perk was able to damage the player herself. At safe distances, this would be no problem for the player, but in close-combat, this proved to be quite the disadvantage. This left us with conflicted feelings. On one hand, it would add to the narrative concept of being so powerful that the character would not only be a danger to everything around it but actually be a danger to themselves as well. On the other hand, it introduced an odd feeling of controlling a character that was powerful in every way but had this one very severe weakness. Ultimately, we decided to not include it, but with some balancing, it could be reintroduced, should we want to pursue the feeling of being too powerful.

The Decision to Leave Out Procedural Generation

After the tests in the process, we reached an impasse. While procedural generation is arguably one of the most defining factors in the roguelike genre, the implementation is also a time-consuming process. Though we had intended the procedural generation to be in the final build of the game we reallocated our resources into making the skills and progression feel good. While this decision had multiple implications for the replayability of the game it also allowed us to test the game and make better decisions about the core gameplay. Though we had not implemented the procedural part of the game, our approach to the level design was indeed procedural because of the aforementioned random approach using a die. This led to a random layout of each floor, simulating procedural generation thus giving us the opportunity to evaluate the game as if the layouts had been generated by the game.

The Final Prototype and Future Work

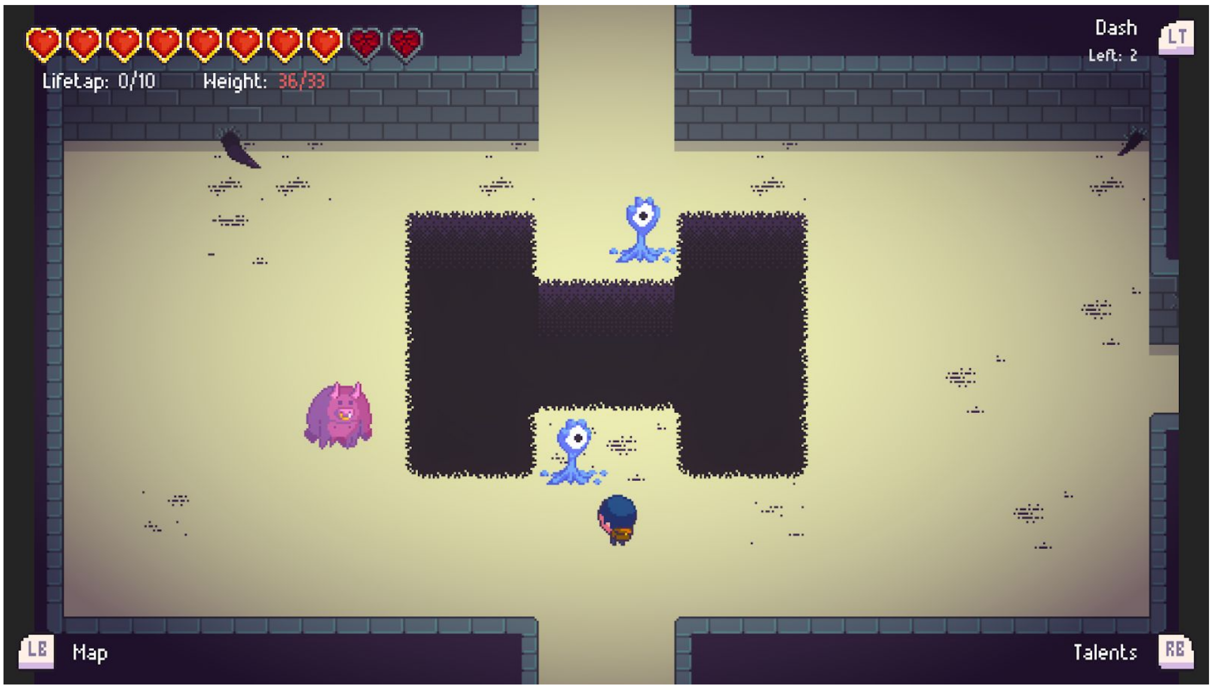


Figure 21: A screenshot of the final prototype build of Relinquish.

Following the playtest, we primarily worked on providing feedback, balancing and polishing the game for hand-in. The playtest had shown that the difficulty of our game was generally too low, which also made it difficult to test the effects of the perks. We reiterated on the skill tree, removing the dash kills perk which had not seen use in any of our playtests and replacing it with the movement speed perk (see Figure 21).

In terms of balancing, we mostly focused on tuning the enemies to be stronger in different ways. Especially the bosses that had previously been big versions of the smaller creatures with the same health and attack. Adjusting these levels alone, had a huge impact on the difficulty of the second half of the game. Being more powerful, enemies would still die easily in the beginning, but later on, the difficulty would become incredibly hard. The final two floors seem to be the most difficult, as the player has very few perks left but still need to go through a lot of rooms.

Final Playtest

With the final changes in place, we decided to do another round of playtesting before the hand-in. We wanted to test the new difficulty, the changes made to the skill tree (including the explosive damage immunity), and how players respond to our updated UI (including sprites from an external artist, see appendix I) and feedback such as sounds (See appendix I). We reused the test setup from the second test, but with reduced interview questions, and managed to recruit two additional playtesters who had not tried the game beforehand.

With explosions out of the way, it became more apparent that the player is very strong at the start. However, playtesters did not immediately notice this. Some would, upon replaying the game, be very surprised how much stronger the character felt, now they had tried the weaker versions. The increased difficulty at later stages of the game was a lot more apparent now. The inability to heal yourself and the very short range of the basic weapon proved to be too much of a challenge, allowing only one of the testers to actually finish the game.

Despite our efforts to improve the UI on the skill tree and on the floor progression to have better explanations, playtesters were still unsure when to progress sometimes. This uncertainty of the players has led to several discussions on how to teach the game to new players. Roguelikes are not known for not providing much information about their elusive nature, but having all of the perks at the beginning and these slightly more complex progression systems might need a better explanation than the one we provide right now.

Future Work

While the prototype we made was sufficient for examining negative progression in roguelikes our playtests and intuition tells us there is more to be explored with the concept. As the project has been set up to be expanded it was natural to ask ourselves what additional findings the concept might be able to provide. The addition of procedural generation might grant insight into the interplay between negative progression and procedural content. While our simulated generation was capable of giving insight into the roguelike genre and negative progression it has not been adequate for exploring how we can design procedurally generated negative progression. As this part has been vital for the replayability of roguelikes by extension their success it is compelling to explore how procedurality and negative progression can successfully be mixed.

In the same way, procedural generation is set up to be implemented, other parts of the game have been created in such a fashion that it should be relatively simple to implement. One such example is in the addition of more complex boss fights. While the bosses in the final prototype do serve to provide players with more interesting choices for strategies, these might not be sufficient for creating the generated by defeating a boss knowing your character power has decreased but your skills as a player have increased. Implementing more complex boss fights would enable us to better study the player experiences provided by negative progression.

Summary of Process

The process was directly affected by feedback from our testers, knowledge gained throughout our developing cycle, and the literary framework. The design process was iterative and several choices were made along the way that changed the final prototype from what we had envisioned at the beginning of the process.

Reflections on the Prototype

Relinquish has two prominent gameplay systems; the general floor requirements and the skill tree. In the following section, we will reflect on the design of these two systems in relation to the literature and the proposed characteristics: progression, rewards, uncertainty, and replayability.

Floor Requirements

The level design can be divided into chest dispersion, room functions, boss, and enemies. The overall play experience hinged on the dynamic between these interconnected elements.

How we gated the access to chests would dictate the pace of the game, how long players would grow accustomed to their new perk related circumstances, and the feeling of progression throughout the experience. Fullerton (2014) and Zagal & Altizer (2014) describe progression either as smaller milestones providing players with a sense of progression or as levelling up a character. In our game, the chests act as milestones for the game's progression. Not only in regards to the skill tree but in the milestones of deleveling, the chests are the points of progression. Initially, we only provided players with the one chest in the boss rooms. Players would fight the boss, delevel three perks, and move on. This would reduce the player to only having one goal on each level. In adding a chest directly related to the task of killing all the enemies on a level, we not only provide a task that spawns a possible new player strategy, but we also provide the player with a secondary milestone on the level. Making the chest entirely optional, as the player can progress without it, and by linking it to a task, this plays into Wang and Sun's (2011) item granting system, as this directly encourages the player to explore, as enemies will be scattered around the levels. The last chest leads us to room functions.

In order to provide players with a possibility of healing during a level, we designed the restroom. Linked to the perk Regen in Rest Rooms, players would be able to interact with a fairy who would heal the players depending on lost health and the perk. In a later prototype, we added access to a third chest through the fairy but at the cost of healing. The room itself and the resulting healing can be categorised under the same category as the optional enemy chest, as the knowledge of possible healing somewhere on the level encourages exploration.

Acting as a challenge the player would add value to the chest in the boss room based on the difficulty of the boss battle. While intended as a narrative and combat bookmark to each level, the openness of the level design made it possible for players to head directly for the boss room. Either way, regardless of difficulty, it provided a milestone for the level and a direct connection to further progression.

Regular enemies, while providing a general challenge for the player and directly connected to one of the chests, in-directly also acted a punishment, or at least as a direct consequence of punishment. As the player would delevel, enemies that were easy to dispatch at the start of the game would become increasingly difficult. This is somewhat intuned with Schell's (2008) idea of shaming. While the game does not directly tell the player that they messed up, the increasingly difficult enemies are a direct result of the player's choices, and as such the player is confronted with the mechanical consequences.

Skill Tree

The second prominent gameplay system of the game is the skill tree. It is the main source of the negative progression, and while it has ties to floor requirements it mostly concerns the character.

In order to evaluate the success of the skill tree in itself, we can apply the quality measure provided by Heinimäki & Elomaa (2015). Their first measure is the number of skills present within the tree. This was specifically important for Relinquish, as it directly dictates the overall length of the game as well. Having too many skills would also increase the amount of time the player is powerful, which could drag out the first part of each session. When asked, testers thought the length of the game was fitting.

Another important measure of their model was balancing and pacing of the tree. Specifically having strong paths be too obvious. We could observe this to some degree. Players had strong preferences toward the damage tree, while often emphasising the importance of the utility tree. However, their preferences varied, and players often described that they would have built their character differently if they were to play again.

Skill trees are commonly built with the concept that the skills should increase in power the more points the player has, leaving weaker skills to be picked at the beginning and powerful skills at the end. While we strived to replicate this (in reverse) some of the last skills the player removes are also some of the most useful skills. When on the last two levels, there is a great chance that players can no longer heal at all and the dash through pits trait halts progression on each floor because it permanently removes the ability to remove one weight.

An interesting dynamic of balancing is how the big perks which are thrown in three pieces seem to have different purposes whether they are in the top row or in the third row. When players start removing the perks on the third row, it allows them to keep the perk for a little longer while progressing. But when the player has to drop the final perk of a tree, the game has become more difficult, and removing three weight for a perk seems to be a hindrance to finishing the game rather than a help for having the skill for a longer time.

Heinimäki & Elomaa's (2015) also outline a measure of whether or not the tree is too rigid in structure and does not allow players to take different paths in different instances of the game. The skill tree in Relinquish is interesting in this regard because it has a fixed start and end. The player will always have all of the perks to begin with and end with none. While the middle can differ in what order the perks are removed, this removes a lot of the uncertainty that is more prevalent in roguelikes with randomness. With time, the player would have tried most combinations of removing the skills and the replayability would decrease.

When the player opens the skill tree for the first time, it is difficult to assess the exact consequences of these choices. This is what Costikyan (2013) describes as analytical complexity. Our skill tree can be reduced to three main strategies and players will familiarise themselves with the strongest strategies and different paths to take. Over time, the analytical complexity will decrease making the game feel less replayable. Already on the second playthrough players will have a basic understanding of the structure and abilities in the skill tree. At the current state, we have no meta-progression and nothing changes between each run. It would be possible to refuel the analytical complexity by adding many systems to switch up the trees by introducing new branches or by allowing players to save some weight and end the game without losing everything.

Having a structured skill tree without randomness provides for a more stable, but also less replayable experience. It is, however, not the structure of placing perks in a tree with dependencies that inherently make it so. Adding some randomness could also create uncertainty when the analytical complexity runs dry. Creating odd synergies through random items and abilities might make it more exciting to do multiple times. Something that often works well within roguelikes with random progression systems is having to adapt your strategy when the desired items do not drop.

When it comes to player strategies, we deliberately wanted to design for different strategies and playstyle. Going back to the strategies outlined by Iacovides, Cox, Avakian & Knoll (2014), we wanted players to experiment rather than use trial and error strategies when choosing perks in the tree. However, the large number of perks available at start seemingly prompted players to choose which perks to give away somewhat randomly until further in the game when they were weaker, which further confirms the analytical complexity of the skill tree.

Something which we found very useful when outlining how to categorise and segment the perks into different branches to provide the opportunity to build the character for a specific playstyle. Having three main sections of the tree with unique identities in itself implies that the player can pursue different builds of their perks and allows players to more consciously attempt a certain playstyle.

Through the design process, we learned that it is possible to invert progression systems in roguelikes, although the introduction of the skill tree was needed to account for the inversion and overall player experience - an unintended consequence was the emergence of several interesting areas related to the general idea of negative progression in games. We will discuss this in the next section.

Reflections on Negative Progression

While our concept includes two negative progression systems through the scale system and the skill system, it can be discussed if we have been able to create a game without any elements of standard progression systems. The current version of the skill system seems to be inherently negative in its progression, but this was not always the case during development.

Throughout the process, it has been the goal to make every perk negative to the player's powers. Having said that, we have encountered at least one perk which was not strictly negative to the progression. As already mentioned, the explosion damage perk would originally damage the player as well. Combined with the level design, this perk was the deadliest part of the game until we decided to make the player immune to her own damage.

Zagal & Altizer (2014) explains how it is common for games to include negative progression systems in the form of small drawbacks, e.g. in the case where a weapon makes the player stronger but gives her a debuff. In a game with a standard progression system, acquiring the player-damaging explosion perk would be defined as negative progression. In our case, removing said perk is an example of standard progression since it gave the player an advantage. This highlights how benefits and drawbacks in perks that affect the player's power, are reversed by situating them in a negative progression system. It also highlights the relation between individual powers and the progression system that they are situated in, as changing the progression system will reverse the perception of the powers.

Player Perception and Progression

The perceived value of different skills will always depend on the player and her strategy. While some players might be motivated to pick a skill because of its ability to kill the enemies in the game others might use the skills for roleplaying purposes.

For example, one could imagine a player who wanted to make the game as hard as possible for herself. This person would perceive the offensive abilities in a much different way than the person trying to find the optimal strategy for completing the game. Wang and Sun (2011) describe this relation between the player and perceived rewards. In Relinquish, the skill tree is directly tied to winning the game. Although Zagal & Altizer (2014) only explain negative character progression it can be argued that the character is progressing by ascending. In this way, the progression towards the end goal is tied directly to the character. While the skill tree provides a strictly negative character progression, the same system includes a positive progression towards finishing the game. This ties progression directly to goals. It seems that all winnable games include positive progression since it is possible to get closer to the goal of winning.

While goals can be explicitly stated in games, such as in the form of written objectives, most goals are only implied. Furthermore, individual players might create their own goals, making it impossible to fully analyse how a system affects the progression towards *every* goal.

This invalidates the question of asking if it is possible to have a game with exclusively negative progression since we can never know the exact goals which are facilitated by the game.

Player Expectations and Conventions

While a game can facilitate almost any type of player goal, some goals are inherently implied by the game. For example, if a game never states that the goal is to win it, it is implied because it is a state which the player has not yet obtained. Furthermore, several conventions have developed throughout the history of games. In this way players generally know it is a good thing to win. Having said that, games do not only imply winning. In Relinquish the player as giving up her powers. As it is generally the convention that one should strive to become more powerful these two opposing factors seem to clash.

This explains an important trait about negative progression. In Relinquish the negative character progression is not defined by the loss of abilities but instead the conventional implicit (and unobtainable in this case) goal of becoming stronger. Likewise, all negative progression systems are defined by the inability to progress towards their conventional goal, because their point is to move the player further away from that goal. This means that negative progression is defined as a system which makes the player move away from what is conventionally a goal.

While it is the case that most games are about accumulating power, currency, points or something else, the term negative progression is seen as an opposite of standard progression, which follows established conventions that the player should always amass fortune, items or power. While this is not meant to be a discussion of semantics, using the word negative as a description of progress enforces the tropes that progress should be focused on this accumulation of power and treasure. Giving up power and treasure does not need to be negative and could even be positive. Instead, it could be named based on whether the player is subtracting or accumulating something in order to progress towards a goal.

Conclusion

Our initial fascination with underexplored aspects of game design led to the idea of inverting traditional progression systems in the roguelike genre. Our initial literature study showed a somewhat underrepresented genre in games studies, lacking a clear definition. This resulted in an expansion of our literature study to encompass design practice related to observed game systems and mechanics in roguelike games. This inspired our four roguelike characteristics: progressions, rewards, uncertainty, and replayability. These four are often used in game design, and as such are not inherently connected to the roguelike genre but traits related to roguelike appear in the intersection between the four. These would become the basis for our design framework.

Our design process was exhaustive and included three playable prototypes and three phases of playtesting. This became a very iterative process where the prototype shifted from a traditional roguelike progression system based on randomness to a structured skill tree offering increased usability. This was the result of the problems that arose in the process of inverting the traditional progressions system, as the player would have to learn all the skills at the start of the game. By introducing the skill tree we ensured that the player would not be overwhelmed with information when starting out.

Reflecting on our prototypes in relation to our literature study we found that while negative progression challenges certain aspects of the design process, the results will often be the same and rely on the same four characteristics proposed for traditional roguelikes. We also reflected on the uncertainty that is connected to the skill tree, and how that offers replayability, if only for a limited time, as more playthroughs eventually result in the player knowing all the possible combinations of perks. Adding randomness, which is more traditional in regards to roguelike progression, would add to the replayability factor as this could create interesting synergy between perks, while also ensuring that the player would not always get the combination they wanted.

Furthermore, we reflected on whether or not we had been able to design a game without using any standard progression systems. We conclude that while our character progression can be considered negative, we still have an overarching traditional progression which aims at enabling the player to win the game.

Negative character progression ties into the idea that it should be the opposite of traditional progressions goals like becoming stronger and amassing wealth, which implies that it is inherently negative to lose or give up something.

We set out to explore if it was possible to invert traditional progression systems in roguelikes. While it is possible to design negative progression systems, certain problems will arise in terms of usability. These problems can be mitigated by adding structure to the progression standardising the sequence of choices presented to the player. The roguelike proved easy to navigate from a designer's perspective, and while not all of our implemented systems were traditional roguelike systems the genre accommodated well for these new additions. While negative progression systems seem unpopular and underutilised in game design, we believe that it is possible to implement and create appealing gameplay using these systems without compromising user experience.

Ludography

A.I. Design. (1980). *Rogue*. [Unix], California, USA: Epyx Computer Software.

Angband Development Team. (1990). *Angband*. [Unix, Microsoft Windows, Mac OS], Coventry, UK.

Biskup, T. (1994). *Ancient Domains of Mystery*. [Linux, Amiga, MS-DOS], Germany: Thomas Biskup.

Blizzard Entertainment. (2002). *Warcraft III: Reign of Chaos.* [Microsoft Windows, Mac OS], California, USA: Blizzard Entertainment.

Blizzard Entertainment. (2003). *Warcraft III: The Frozen Throne.* [Microsoft Windows, Mac OS], California, USA: Blizzard Entertainment.

Blizzard North. (1997). *Diablo.* [Microsoft Windows, PlayStation, Mac OS], California, USA: Blizzard Entertainment.

Blue Manchu. (2019). *Void Bastards.* [Microsoft Windows, Xbox One, Nintendo Switch, PlayStation 4], California, USA: Humble Bundle, Inc.

Cellar Door Games. (2013). *Rogue Legacy.* [Microsoft Windows, Linux, OS X], Ontario, Canada, Cellar Door Games.

Chunsoft. (1993). *Torneko no Daibōken: Fushigi no Dungeon.* [Super Famicon], Tokyo, Japan: Enix Corporation.

DCSS Devteam. (2006). *Dungeon Crawl Stone Soup*. [Web browser].

Dodge Roll. (2016). *Enter the Gungeon.* [Microsoft Windows, OS X, Linux, PlayStation 4], Texas, USA: Devolver Digital.

Gygax, G., Arneson, D. (1974). *Dungeons and Dragons.* [Analog game], Washington, USA: Wizards of the Coast.

Hopoo Games. (2019). *Risk of Rain 2.* [Microsoft Windows, Nintendo Switch, PlayStation 4, Xbox One], Texas, USA: Gearbox Publishing.

McMillen, E. (2011). *The Binding of Isaac.* [Microsoft Windows, OS X, Linux], California, USA.

Nicalis. (2014). *The Binding of Isaac: Rebirth.* [Multiple Platforms], California, USA.

Mossmouth, LLC. *Spelunky.* [Microsoft Windows], California, USA.

Motion Twin. (2018). *Dead Cells.* [Linux, macOS, PlayStation 4, Switch, Microsoft Windows, Xbox One], Bordeaux, France.

NetHack DevTeam. (1987). *NetHack.* [Cross-platform].

Resch, P., Kemp, L. & Hagstrom, E. (1975). *orthanc1.* [PLATO Network], USA.

Rutherford, R. (1975). *pedit5.* [PLATO Network], USA.

Vlambeer. (2015). *Nuclear Throne.* [Microsoft Windows, OS X, Linux, PlayStation 4, PlayStation Vita], Utrecht, Netherlands.

References

Aarseth, E. (2007, September). *I Fought the Law: Transgressive Play and The Implied Player*. Paper presented at the DiGRA 2007 – "Situated Play", Tokyo, Japan. Retrieved May 14, 2020 from http://www.digra.org/wp-content/uploads/digital-library/07313.03489.pdf

Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD research*, *1*(1), 19.

Berlin Interpretation. (2008). Retrieved May 10, 2020, from http://roguebasin.roguelikedevelopment.org/index.php?title=Berlin_Interpretation

Brewer, N. (2017). Computerized Dungeons and Randomly Generated Worlds: From Rogue to Minecraft [Scanning Our Past]. *Proceedings of the IEEE*, *105*(5), 970–977.

Bycer, J. (2013). The Procession of Progression in Game Design. Retrieved May 8, 2020, from https://www.gamasutra.com/blogs/JoshBycer/20130523/192906/The_Procession_of_Progression_in_Game_Design.php

Caillois, R. (2001). *Man, play and games*. Urbana, IL: University of Illinois Press.(Original work published 1958)

Costikyan, G. (2013). *Uncertainty in Games*. MIT Press. Cambridge, MA.

Debus, M. S. (2018, July). *Implied Play Styles: An Analysis Tool to Understand Player Behavior and Meta-Game Emergence*. Paper presented at DiGRA 2018 – "The Game is the Message", Turin, Italy. Retrieved from digra.org/wp-content/uploads/digital-library/DIGRA_2018_paper_254.pdf

Frattesi, T., Griesbach, D., Leith, J., Shaffer, T., & DeWinter, J. (2011). *Replayability of video games. Worcester Polytechnic Institute, Worcester, England*. From https://digitalcommons.wpi.edu/iqp-all/923/

Fullerton, T. (2014). *Game Design Workshop: a Playcentric Approach to Creating Innovative Game*s. CRC press. Boca Raton, FL.

Garda, M. B. (2013). Neo-rogue and the essence of roguelikeness. *Homo Ludens, 1(5),* pp. 59-72. Retrieved from https://www.academia.edu/6422882/Neo-rogue_and_the_essence_of_roguelikeness

Ghys, T. (2012). Technology trees: Freedom and determinism in historical strategy games. *Game Studies*, *12*(1), pp. 143-52.

Grey, D. (2013). Screw the Berlin Interpretation! Retrieved May 13, 2020, from http://www.gamesofgrey.com/blog/?p=403

Heinimäki, T. J., & Elomaa, T. (2015). Quality measures for improving technology trees. *International Journal of Computer Games Technology*, *2015*.

Ho, X., Tomitsch, M., & Bednarz, T. (2016). Finding design influence within roguelike games. In *International Academic Conference on Meaningful Play* (pp. 1-27). East Lansing, MI.

Iacovides, I., Cox, A. L., Avakian, A., & Knoll, T. (2014, October). Player strategies: achieving breakthroughs and progressing in single-player and cooperative games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play* (pp. 131-140). Toronto, Canada.

Johnson, M. R. (2017). The Use of ASCII Graphics in Roguelikes. *Games and Culture*, *12*(2), pp. 115–135.

Roth, C., Vermeulen, I., Vorderer, P., & Klimmt, C. (2012). Exploring replay value: shifts and continuities in user experiences between first and second exposure to an interactive story. *Cyberpsychology, Behavior, and Social Networking*, *15*(7), pp. 378-381.

Schell, J. (2008). *The Art of Game Design: A book of lenses*. CRC press. Boca Raton, FL.

Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural content generation in games*. Switzerland: Springer International Publishing.

Sullivan, A., & Salter, A. (2017, August). A taxonomy of narrative-centric board and card games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (pp. 1-10). Cape Cod, MA.

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. N. (2011, June). What is procedural content generation? Mario on the borderline. In *Proceedings of the 2nd international* *workshop on procedural content generation in games* (pp. 1-6). France.

Wang, H., & Sun, C. T. (2011, September). *Game reward systems: Gaming experiences and social meanings*. Paper presented at the DiGRA 2011 – "Think, Design, Play" (Vol. 114). Hilversum, The Netherlands. Retrieved from http://www.digra.org/wp-content/uploads/digital-library/11310.20247.pdf

worthplayingvideos (2017, March 19). *Dead Cells [PC] Procedural Generation Dev Diary Trailer* [Video]. Youtube. https://www.youtube.com/watch?v=dvSbKXM5Vsc

Zagal, J. P., & Altizer, R. (2014, April). Examining 'RPG elements': Systems of character progression. Paper presented at the 9th International Conference on the Foundations of Digital Games, Ft. Lauderdale, FL. Retrieved from http://fdg2014.org/papers/fdg2014_paper_38.pdf

Appendix

Appendix I: List of Interview Questions for the 2nd and 3rd Playtest

What was your impression of the game after playing a couple of minutes?
Has it changed now?

Was there anything you found frustrating?

Were there particular aspects that you found satisfying

What was the most exciting moment in the game?

Did the game feel too long, too short, or just about right?

What was the most important decision you made?

What was your strategy for winning?

(Did you find any loopholes in the system?)

What elements do you think could be improved?

Was the game's premise appealing to you?

How would you make the story and game work better as a whole?

How did the controls feel? Did they make sense?

Did anything feel clunky, awkward, or confusing?

Are there any controls or interface features you would like to see added?

If you could change just one thing, what would it be?

What did you think of the skills/talents?

Appendix II: Assets

External Sounds:

Sounds were taken from:

ZapSplat (n.d.) *ZapSplat*. Retrieved April 22, 2020, from https://www.zapsplat.com/
and
FreeSound (n.d.) *FreeSound*: Retrieved April 22, 2020, from https://freesound.org/

Chest:

ZapSplat (n.d.) *Child-stair-gate-hinge-squeak-and-creak-1* [MP3]. From: https://www.zapsplat.com/music/child-stair-gate-hinge-squeak-and-creak-1/

Charge Enemy Windup:

ZapSplat (n.d.) *Person-breath-exhale-through-diving-snorkel* [MP3]. From: https://www.zapsplat.com/music/person-breath-exhale-through-diving-snorkel/

Little Robot Sound Factory (n.d.) *Evil-demon-voice-vocalization-001* [MP3]. From: https://www.zapsplat.com/music/evil-demon-voice-vocalization-001/

Charge Sound:

Little Robot Sound Factory (n.d.) *Evil-demon-voice-vocalization-008* [MP3]. From: https://www.zapsplat.com/music/evil-demon-voice-vocalization-008/

Felix Blume (n.d.) *Cow-moo-close-up* [MP3]. From: https://www.zapsplat.com/music/cow-moo-close-up/

Audio Hero (n.d.) *Cows-cattle-moo-mooing* [MP3]. From: https://www.zapsplat.com/music/cows-cattle-moo-mooing/

Charger Stomp:

000600 (March 6, 2013) *foot stomps ok* [Wav]. From https://freesound.org/people/000600/sounds/180015/

Dash sound:

ZapSplat (n.d.) *Bendy-stick-whoosh-through-air-fast-4* [MP3]. From: https://www.zapsplat.com/music/bendy-stick-whoosh-through-air-fast-4/

ZapSplat (n.d.) *Cane-stick-whoosh-whip-in-air-2*[MP3]. From: https://www.zapsplat.com/music/cane-stick-whoosh-whip-in-air-2/

Elevator Sounds:

Felix Blume (n.d.) *Elevator-ascending* [MP3]. From: https://www.zapsplat.com/music/elevator-ascending/

ZapSplat (n.d.) *Elevator-bell* [MP3]. From: https://www.zapsplat.com/music/elevator-bell/

ZapSplat (n.d.) *4-disposable-nappies-drop-on-carpeted-floor* [MP3]. From: https://www.zapsplat.com/music/4-disposable-nappies-drop-on-carpeted-floor/

Explosion:

ZapSplat (n.d.) *Gunshot-exterior* [MP3]. From: https://www.zapsplat.com/music/gunshot-exterior/

SmartSound (n.d.) *Shot-with-a-revolver-from-100m-away* [MP3]. From: https://www.zapsplat.com/music/shot-with-a-revolver-from-100m-away/

Heal Room Sound:

DJczyszy (February 23, 2019) *chimes-overheads_005* [flack]. From: https://freesound.org/people/DJczyszy/sounds/460482/

Peashooter:

ZapSplat (n.d.) *Game-sound-designed-bubble-pop-17* [MP3]. From: https://www.zapsplat.com/music/game-sound-designed-bubble-pop-17/

Shooter shot:

ZapSplat (n.d.) *Cartoon-bubble-popping-or-other-pop-3* [MP3]. From: https://www.zapsplat.com/music/cartoon-bubble-popping-or-other-pop-3/

Shooter impact:

ZapSplat (n.d.) *Cartoon-slime-drip-fall-and-hit-surface* [MP3]. From: https://www.zapsplat.com/music/cartoon-slime-drip-fall-and-hit-surface/

Taking damage:

ZapSplat (n.d.) *Cartoon-voice-high-pitched-grunt-in-pain-8* [MP3]. From: https://www.zapsplat.com/music/cartoon-voice-high-pitched-grunt-in-pain-8/

ZapSplat (n.d.) *body-hit-ground-hard-with-splat-squelch-of-blood-and-guts* [MP3]. From:

https://www.zapsplat.com/music/body-hit-ground-hard-with-splat-squelch-of-blood-and-guts/

Teleport:

Leszek_Szary (December 21, 2012) *teleport* [Wav]. From: https://freesound.org/people/Leszek_Szary/sounds/172207/

UI Select:

Breviceps (November 16, 2018) *448086* [Wav]. From: https://freesound.org/people/Breviceps/sounds/448086/

UI Wrong:

Wobesound (October 14, 2019) *488402* [Wav]. From: https://freesound.org/people/wobesound/sounds/488402/

External UI Sprites:

Hyohnoo. (2016) Free pixel Xbox buttons you can use (^▽^*) [PNG]. twitter.com/Hyohnoo. From: https://twitter.com/Hyohnoo/status/747866064466415616/photo/1