Toni Aaltonen

# 3D platformer level design

Bachelor of business administration

Data processing

Spring 2018

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

## Abstract

**Author**: Aaltonen Toni

**Title of the Publication:** 3D platformer level design

**Degree Title:** Bachelor of business administration, game industry

**Keywords:** game design, level design, 3D platformer, psychology

## 3D platformer level design

The goal of this Bachelor's thesis was to design a level for Biting Mascot's 3D platformer project Captain's Tail and explore the intricacies of 3D platformer level design.

The first chapter elaborates on the meaning and goals of level design, so that the reader will better understand the subject of the thesis. It is then explained what was discovered while researching the common practices and different methods of level design in all game genres. Finally, these theories were attempted to put into practice in our company's 3D platformer project.

The level design was based around the theoretical info found during research, then tested thoroughly and all observations were documented. To end the thesis, all observations are analyzed into a list of conclusions about 3D platformer level design.

Table of contents

1 Introduction ....3

1.1 What is level design? ....4

1.2 The process of level design....6

2 The premise....7

2.1 Assignment backgrounds....7

2.2 My personal level design backgrounds....9

3 Level design research ....11

3.1 How to get started ....11

3.1.1 High-level constraints....11

3.1.2 Common constraints ....12

3.2 Designing level structure ....24

3.3 Navigation and player guidance ....29

3.4 Visual language....36

3.5 Composition and framing ....37

3.6 Pacing....41

3.7 Conclusion ....45

4 Captain’s Tail level design process ....47

4.1 Level 1 planning and sketching ....48

4.1.1 Area 1....53

4.1.2 Area 2....55

4.2 Level 1 white-boxing, playtesting and iteration ....56

4.2.1 Area 3....59

4.2.2 Area 4....63

4.3 Level 1 division....68

4.4 Level 1 conclusion....70

4.5 Level 2 conclusion....71

5 Retrospect ....75

5.1 Constraints and checklists....75

5.2 Visual timeline and bubble diagram....75

5.3 Theoretical information....76

5.4 The process ....77

5.5 Final words 78

6 References 79

6.1 Information sources 79

6.2 Image sources 81

## List of symbols

Game mechanic = A construct of rules or methods designed for interaction with the game state, thus providing gameplay. [25] For example, pressing a button to make a game character jump counts as a game mechanic. *Core* game mechanics refer to the most important game mechanics of a specific game.

Gameplay = The specific way in which players interact with a game. [26] For example, if a player jumps in a game, it is considered gameplay, because the player is interacting with the game's mechanics.

Platforming = Refers to the action of traversing from platform to platform in a game, usually by jumping. The core game mechanic of many platformers.

Platformer = A genre of video games where jumping is usually the most important core gameplay mechanic. For example, games like Super Mario Bros. and Crash Bandicoot.

White-box = In this thesis, white-box refers to the phase in level design when a level is constructed out of simple shapes for playtesting purposes. White-boxing refers to the process of building a level with said simple shapes.

A game's vision = An imagination of a game before the final product even exists. Usually the project lead is responsible for communicating a game's vision to the rest of the team during game development, to ensure that everyone knows what kind of a game they're making.

Target audience = A group of people that a specific game is designed and marketed for.

Design pillar = A key element in a game's design, determined early on in game development. They usually dictate what qualities are important for the game or what emotions the game should invoke in players. Design pillars should always be considered when making design decisions during development. A game's vision is usually made up of design pillars that define it.

Critical path = The main path of a level that the player should take if they wish to travel to the end of the level.

Branching path = Other paths beside the critical path, usually leading to optional content or in-game rewards.

FPS = First-person shooter. A genre of video games where the player controls their character from a first-person perspective and usually includes shooting as a core game mechanic. For example, games like Doom and Counter-Strike.

NPC = Non-playable character. Side characters in games, which the player is usually able to interact with somehow.

# 1 Introduction

The goal of this thesis was to explore all the important things that must be taken into consideration when designing levels for 3D platformers and to some extent, games of other genres. There isn't much information specifically on 3D platformer level design on the internet and I felt like I could use this thesis to help the reduce the deficiency, by letting the reader observe my successes and failures and learn from them. I also did this thesis as an assignment for our company; Biting Mascot. Biting Mascot is working on a 3D platformer project and I am responsible for the game's level design.

I start by explaining what level design actually is and why it's an important part of some games' development process in the first place. This is important in understanding the actual topic much better, especially if one is not already familiar with game development.

After this, I explain the premises of this thesis. This includes the details of my assignment, what were my restrictions, what previous experience I have had with level design and how all that affect the contents of this thesis.

Next, I talk about the common practices of level design that I discovered during my research. These practices are mostly theoretical info found from reliable sources on the internet or literature. Meaning that said practices may have worked well in their cases and might theoretically work in other cases as well, but we cannot ultimately say for sure unless we try them out ourselves. Also, these practices are from all sorts of genres, not just 3D platformers. This is because some methods and principles carry over from genre to genre.

After that comes the documentation of the actual level design process of our project, where I attempted to utilize all the applicable practices and principles I found during my research. I elaborate on what I did and why did I end up doing so. The documentation includes some test results from the levels, such as what parts of the level worked right from the start, what had to be changed, what had to be scrapped completely and so on. I also talk about any additional observations I made during the process.

Lastly, I move on to the analysis section. I analyze the observations I made during the design process of what worked and why. I also analyze the success of the project and how it could have gone better in retrospect. I end the thesis by talking about how I plan to approach level design differently in the future.

## 1.1 What is level design?

Level design is one of the subgenres of game design and it is a crucial part of most games' development. In most cases, a game's level design dictates the player's experiences in the game. Any scene of player interaction can be qualified as a level within a game, whether it's a mission with numerous objectives, a puzzle that must be solved or an open world where the player can roam freely and play with the game's mechanics as they see fit. A level designer's job is to design these levels from start to finish, but what is the goal of level design in the first place? [1]

The utmost goal of level design is to satisfy customers, as is the case with other areas of design as well. In order to achieve this, one must understand why the target audience would want to play our game. The most common reasons are escapism, challenge and uniqueness which all fall into the category of entertainment. There are even more different reasons why certain people play certain games, but more than often it is about entertainment. This is why the more precise goal of level design is to utilize the game's mechanics to create interesting and entertaining scenarios as well as offer the player objectives to complete. [2] [3] Mega Man will serve as an example of what this actually means.

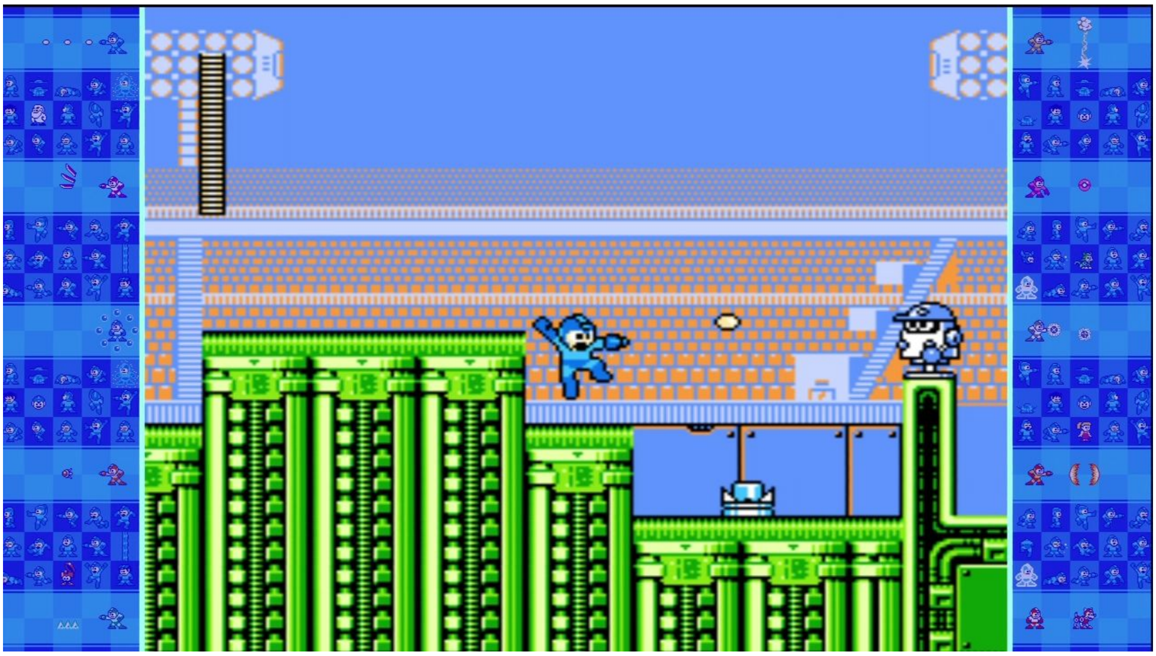


Figure 1. Screenshot of Mega Man 10, a side-scrolling 2D platformer game.

In Mega Man games, the core gameplay mechanics include running, shooting and jumping, which is an essential part of the platformer genre. The level designer of these games has had to create objectives, challenges and rewards for the player for utilizing these mechanics in the game. This is why the player must run to advance in the levels, shoot to defeat enemies and jump to cross gaps and other obstacles. The entertainment comes from combining all these mechanics in all kinds of different situations in the levels and overcoming the challenges that the game places in the player's path. The game mechanics alone might be fun enough in some cases and don't necessarily need level design to be entertaining, but level design cannot even exist without the game mechanics. Level design is there to give the game mechanics a purpose, a goal to work towards using those game mechanics. If Mega Man games would not have levels designed specifically for the mechanics of the game, it would most likely not be a very entertaining experience.

This is why levels usually don't translate to other games very well, unless the games are similar enough, such as games from the same genre. Therefore, there are no universally correct answers or methods to level design. Each game has its own unique set of game mechanics and own unique levels designed specifically for those mechanics.

## 1.2 The process of level design

What does the level designer do exactly? There are many different methods to go about level design, but the basic principle is the same; the level designer must understand the game's mechanics and their purpose as well as the game's vision in order to properly design levels for the game. It is also important to grasp how the game's mechanics work together and what kinds of possibilities their synergy offers in terms of level design. The rest of the process I have described below is just the most common way of approaching level design, according to numerous sources.

Once the level designer has a firm grasp on the game's vision and core mechanics, the actual creative process can begin. The designer must think of an objective for the level and what the player must do to achieve it. As I mentioned before, to make the level entertaining, the player must utilize the game's mechanics to achieve the level's goals. The designer may also have to take the level's theme into consideration, as it may provide the designer with important constraints or inspiration. A rough idea of the level is usually sketched on paper or documented in some other clear but cost-effective way.

After a sketch of the level has been made, it is time to move into the "white-box" phase. In this phase, a prototype of the level will be built to playtest its functionality. At this point, no art assets will be added to the level and the level will usually be constructed of nothing but white boxes, hence the name white-box. Once the level has been built, the playtesting can begin. During playtesting, the white-box level goes through an iterative process and the level must be adjusted according to the test results. Once the results are satisfactory, the iteration may end and art assets will be added to the level. After the art has been added and the level looks "finished", it may still need some playtesting to ensure that the level's readability or other important qualities have not been disturbed by the added art assets. Once that has been done, the level may be considered finally finished. [3] [4]

# 2 The premise

## 2.1 Assignment backgrounds

This thesis is an assignment for our small game development company, Biting Mascot. Researching and documenting the level design guidelines of a 3D platformer will help the company in the future, both with this project and possibly with other future projects.

The project I am designing levels for originally started as a student project in September 2016. We made a demo of a mascot-based 3D platformer with Unreal Engine 4 called The Land of the Scurvy Dog, which included only one level. It revolved around a pirate-themed world where the main character is a pirate corgi, who’s on a quest for treasure. We wanted to learn about 3D platformer development and working with Unreal Engine 4, since it was our first time using said game engine.

We finished the demo in December 2016 and published it on itch.io, a website where anyone can publish their games. After that, we left the demo be and started working on other projects, but despite that we eagerly showed the demo around at a few events, such as Assembly and Northern Game Summit. This way we got some really good feedback on the game and it even got some pretty good reception, resulting in some people constantly asking us when we’re going to continue the project. We founded our startup company at the end of 2017 and eventually in March 2018 we decided to continue the 3D platformer project, or rather, start again from scratch with Unity, a different game engine. The picture below is from the Unreal Engine demo.

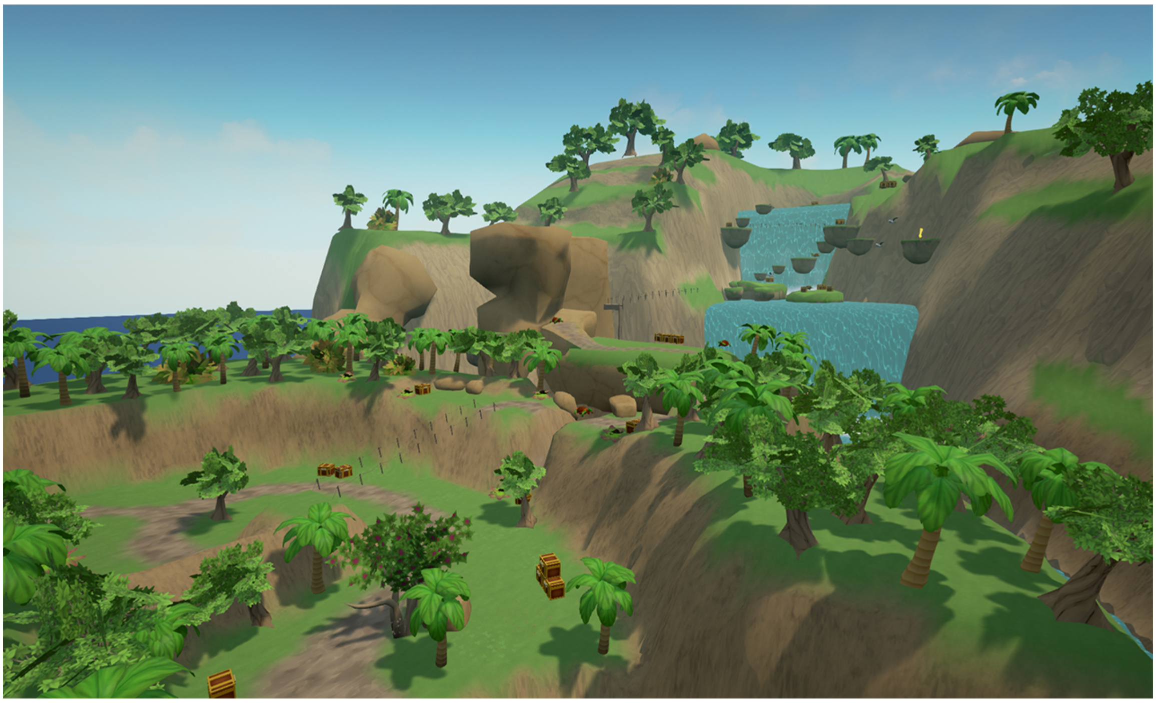


Figure 2. A screenshot of The Land of the Scurvy Dog.

In the Unreal Engine demo, we designed the game's mechanics to make the movement feel fluid, fast and responsive. Originally, I designed the levels to be strictly linear, much like in games like Crash Bandicoot.  This was because I wanted to design specific challenges and experiences for the player, which is much easier in linear level design, since the designer can always rely on the player taking a specific path and crafting everything around that path. Due to the pirate theme of the game, together we decided to settle on a small tropical Caribbean island as the theme of the demo level. This also affected the level's design; in order for the level to be an island, it would have to be surrounded by water. This led us to the decision that the player should not be able to swim, so they could not break the linearity of the level. This also allowed us to use water as an impassable obstacle whenever we wanted to restrict the player's access to certain areas. And also, since the island is tropical and nigh uninhabited, the geometry of the level has to emulate nature's asymmetrical and uneven terrain.  However, after finishing the demo and getting feedback from multiple events, we realized that the linearity of the levels did not serve the mechanics as well as they could have. The testers liked the mechanics and the movement though, so we decided that the level design would have to change to better accommodate the mechanics.

As we decided to start the project again from scratch in March 2018, we took a long, hard look at it to figure out what made the game fun and what we would have to focus on. We decided to keep the fluid and responsive movement as one of our design pillars and started adjusting some mechanics to further emphasize this. We also added **momentum** as one of our new design pillars, meaning that we wanted to make sure that none of the mechanics would forcefully break the player characters momentum to a screeching halt. To better befit the mechanics, I decided to try designing more open levels.

We made a deal with another student team at the end of 2017 to test some more open level design in the project for us. They made an open island level with multiple critical paths to the level's goal. The open nature of the level did prove to be more fun to navigate than the linear level in the demo, however the multiple critical paths did not work so well. After finishing one of the critical paths to the level's goal, the player was left with a feeling of uncertainty, a feeling that they've missed something important, since they've only seen a small part of the level after reaching the goal. This led me to the decision that I should try to design open levels with only one critical path by placing open areas *around* the critical path in the form of open fields, optional branching paths and a critical path that loops in on itself.

I had 3 weeks to design, sketch, white-box, playtest and finish our first level in the project that started from scratch and I have documented my work during these 3 weeks in this thesis.

## 2.2 My personal level design backgrounds

I started tinkering with in-game level editors at a young age and I've always been a big fan of Legos. My first big level design project was a rom hack of Super Mario World that I made during junior high school using a level editor called Lunar Magic. What that basically means is that I used an unofficial level editor to make my own levels in an existing game with ready-made game mechanics. The project included over 60 2D platformer levels. The levels were not that good from a professional point of view, but they served as a huge learning experience for me.

Soon after that I started a rom hack of another 2D platformer; New Super Mario Bros. for the Nintendo DS. I never finished this project, but I did manage to create over 40 levels before I quit. This served as another great learning experience.

In addition to those two projects, I have designed and created levels for Portal 2, Time Splitters, LittleBigPlanet, Modnation Racers, Super Mario Maker and some other smaller games with in-game level editors. During my studies I have designed levels for multiple projects, including top-down shooters, puzzle games and of course platformers.

Due to my track record, I’ve been familiar with platformer level design for quite a while now, but this project is my first experience with 3D platformer level design, which is a whole new can of worms. I did learn some valuable things when we made the demo, which helped me prepare for this level design task, but I still went through a whole new learning process while making this thesis.

# 3 Level design research

Before I started my actual level design process for this project, I looked up some information on efficient ways to start designing levels and what are the most common things to take into consideration during level design. In this chapter, I share what I found and what I planned to use in my process.

I started searching for information about level design in general. The most common things that came up in multiple articles and videos regarding level design were ease of navigation, player guidance, pacing, difficulty curves, views, vistas and composition. I will elaborate more on these later. First things first; how to get started.

## 3.1 How to get started

Approaching a blank slate may be difficult for many, but luckily level design rarely comes first in game development. There is always something to draw **goals and constraints** from for a level, usually from the core game mechanics, since those are what the designer will be designing levels for. Planning is a key element in level design and should usually be done thoroughly before jumping into the game's level editor. There is usually a checklist of things the designer must first determine before they can start designing a level, namely the constraints.

### 3.1.1 High-level constraints

The most solid constraints are the ones that come from a higher level in a large company, since there may be very little a single developer can do to change them. These constraints include deadlines, franchise constraints, audience constraints, client requirements and engine constraints.

**Deadlines** refer simply the project's schedule and one should design levels fit for that schedule. This is extremely company-specific, but usually once the schedules have been made, all the designer can do is obey them and aim to design levels fit for those schedules.

**Franchise constraints** usually include what things are associated with the franchise and how to avoid making the franchise lose its identity. For example, no one would probably design a level set in a dark, rotten dungeon for a game about Care Bears. Figure out what makes the franchise and follow those constraints.

The game's target audience determines its **audience constraints**, for example what kind of gameplay appeals to the target audience, how challenging should the game be, should the designer take players with specific disabilities into account and so on. Levels should always be designed for the game's target audience, otherwise there's nobody to buy the game.

**Client requirements** only apply if the game or level has been commissioned by an external organization and the development team is being paid to do it. It is really up to the team how strongly they are willing to follow the client's demands, but it should always be kept in mind that the client is the one paying for the development.

And finally, **engine constraints** determine the engine that the game is running on and what limits the engine sets for development. Usually this only applies to the optimization or complexity of the levels to ensure good performance when playing, but it is still an important thing to be considered. It is important to look up if there are any things that should be looked out for when using public engines. When using an in-house engine, one should discuss about this with the team's in-house engineers. [8]

### 3.1.2 Common constraints

Once the high-level constraints have been determined, the designer should determine the more specific constraints and objectives of the level they are about to design. These are the most common constraints in general level design, not all of them apply to every sort of game. Still, one should consider each when planning a level, even if the answer is just “no”.

Whether the level is intended for **single-player** or **multi-player** gameplay should usually be determined early, since single-player and multi-player level design are two very different matters. Some levels can even be both; single-player levels repurposed as multi-player levels or vice-versa, but whatever the case, it should be determined early on, as it will affect many things. [5]

Figure 3. A screenshot of Uncharted 3: Drake's Deception.

For example, in games like Uncharted, the single-player levels are usually extremely linear, contain plenty of cinematic set-pieces and focus a lot on the narrative content. But the multiplayer levels of Uncharted games are completely different, since they're usually much more open, offer plenty of routes to travel by and are designed gameplay-first to provide a fun and balanced gameplay experience for all players. The content focus is completely different when it comes to single-player and multi-player level design, but a level can still be designed for both. It just requires a lot more time and thought put into it.

The **level's purpose** can vary quite greatly. The designer has to ask themselves why is that level in the game in the first place. Is its purpose to teach the player a game mechanic? Or is it to advance the game's story? Or maybe it is to show off the technology in the game engine? It can even be all of the above. One can think of the big picture and assign "roles" to each level. [5] [8]

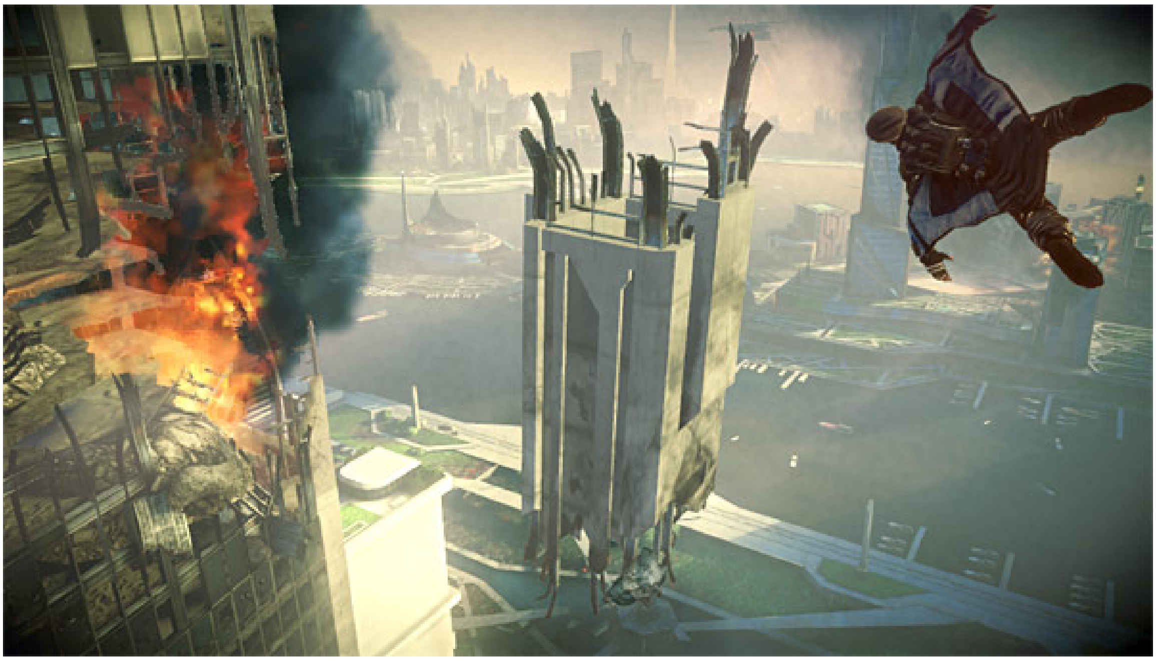


Figure 4. A screenshot of KillZone: Mercenary.

For example, the first level of KillZone: Mercenary had clear purposes to fill. The narrative purpose of the level was to rescue an admiral from the enemy army. Another narrative purpose was to include a tank boss fight, which was also gameplay related. And finally, the level had to include a wingsuit flight sequence to show off the power of PS Vita. These were the instructions given to the level designers from higher above and they had to build it all into a complete level. In smaller studios, level designers might have to set the purposes of their levels themselves. [9]

The **theme** of the level can and will direct quite a lot of things, from the intensity of the level to its environmental structure. Whether the level's theme is the streets of Victorian London or the mountaintops in the Himalayas, it is a considerable constraint that affects numerous aspects of the level. It is possible to design a level gameplay first and then decide what coat of paint to apply afterwards, but in most cases knowing the theme of the level can help the designer in the design process and offer inspiration in case of mental blocks. It should be ensured that the theme and the level's design are consistent. [7] [8]

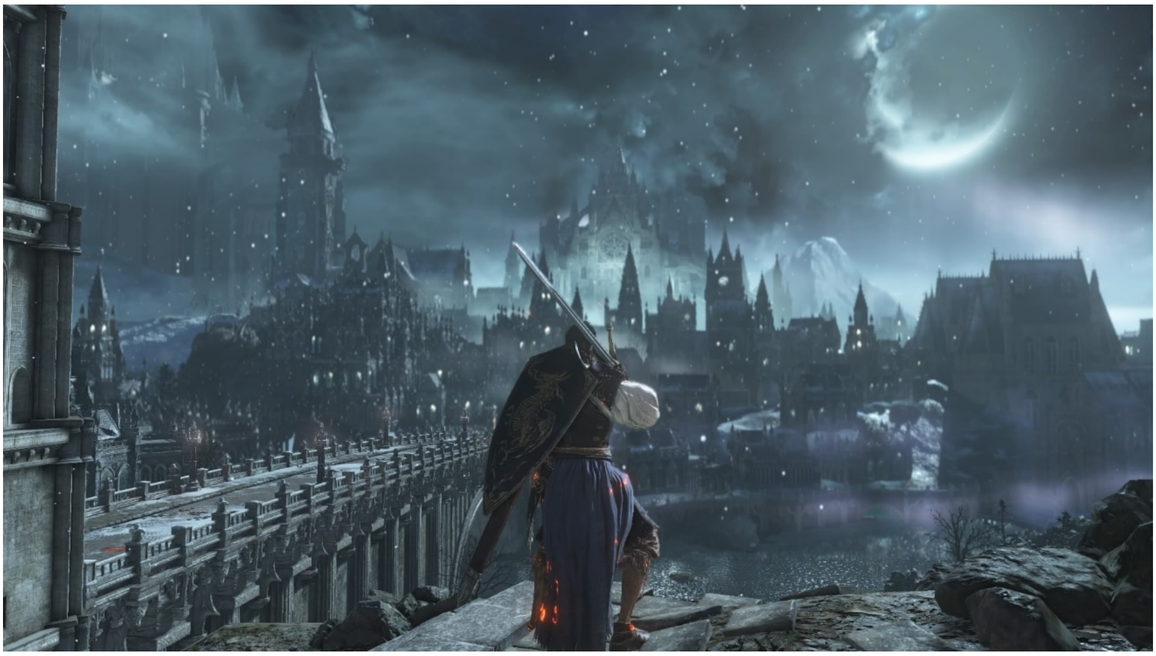


Figure 5. A screenshot of Dark Souls 3.

For example, in each Dark Souls game, there are a variety of different areas, each with their own unique theme. Whether it’s a gloomy town, an immense cathedral, a mucky swamp or the depths of a volcanic cave, all areas have distinct layouts and structures based on their theme. The designer has to think of what kinds of qualities make or break that theme. The level can’t have a town without plenty of buildings, it can’t have a forest without plenty of trees, it can’t have a volcanic cave without lava pits and so on. And likewise, the designer has to think about what things would feel out of place in a certain theme. There probably would not be a pit of lava in the middle of a snowy town, unless it is a unique theme that the level is precisely going for.

The **goal of the level** is what can vary the most between games and genres. If it’s a racing game, the goal is most likely to reach the finish line, usually at the end of a track. If it’s a puzzle game, the goal of the level is to solve the puzzle with the information and tools given to the player. The level goals within genres vary greatly also, so there really is not a one-size-fits-all solution to the goal of a level. The designer should consider the experience they want the player to have in their level and think about what type of goal would make the player have this experience with the mechanics in the game. [5]

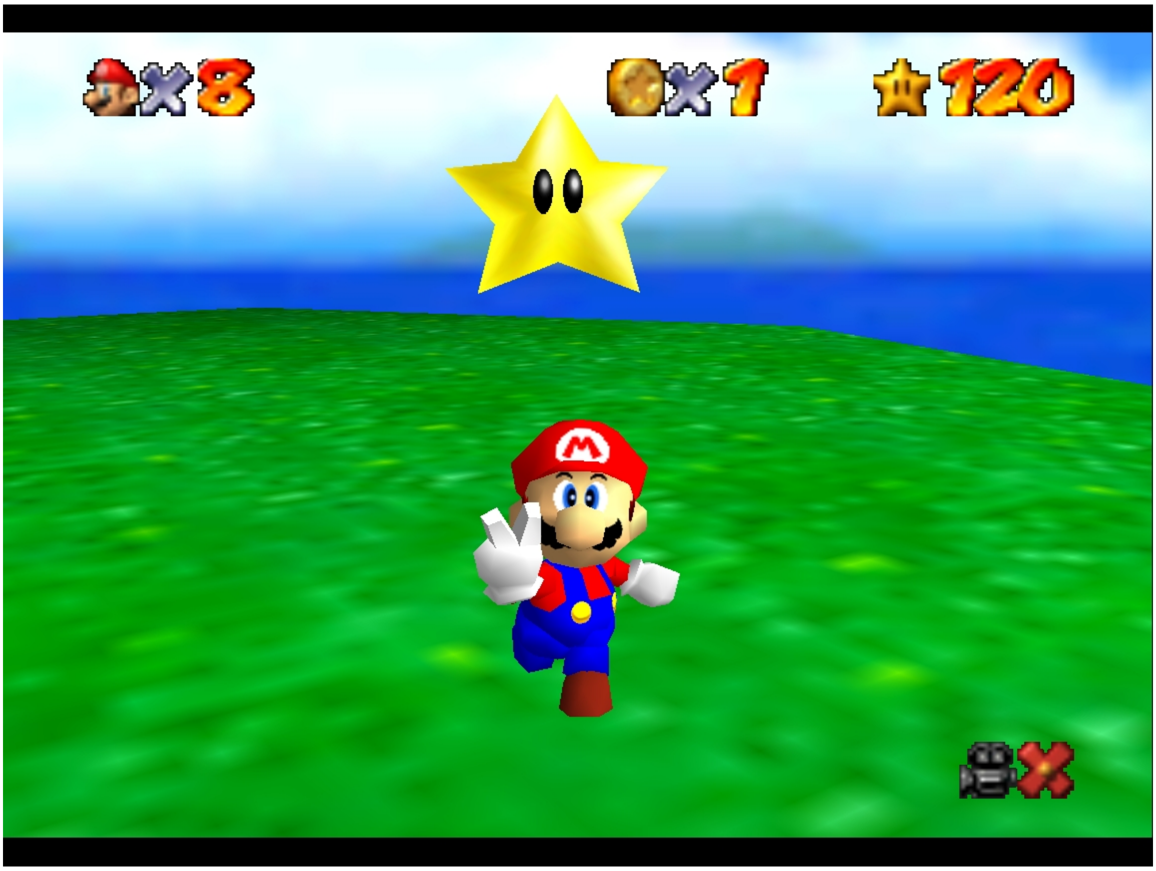


Figure 6. A screenshot of Super Mario 64.

For example, the goal of each level in Super Mario 64 is to find and collect a power star. By making the goal of each level consistent, the player is instantly able to recognize how they're supposed to complete a level. If they see a star, all they have to figure out is how to get there. Stars in Super Mario 64 are usually placed at the end of platforming challenges or hidden away in secret paths, so that the player will have to explore and utilize their platforming mechanics in order to reach the stars and complete levels.

Speaking of **player experience**, it's important to nail that down as well. What does the designer want the player to do in the level? Will it be an exciting action-packed level full of thrills? Or a serene level where the player can just relax? Or will it be a spooky level to show the player the true definition of fear? These things are usually dictated by the themes of the game, but the designer also needs to think about the moment-to-moment gameplay within the level. How will they invoke these feelings of serenity or terror in the player? How do they want the player to utilize the game's mechanics in different segments of the level? What is the experience they want the player to walk out of their level with? The level designer is responsible for the player's experience, they should make it worth their while. [5]

Figure 7. A screenshot of Dead Space 3.

For example, in horror games like Dead Space, the desired player experience is tension. These kinds of games resort to all sorts of tricks in order to keep the player's tension up as long as possible and then release the tension with a frightening moment at just the right time. The way they do this in their levels is that they make sure that the player knows there is a constant threat of something horrible that could attack the player at any time. This keeps the player on their toes even though nothing exceptionally exciting is happening all the time. Then they make the players face the horrific monstrosities not only to provide release for the tension, but also as a sort of reminder that the threat is still present, keeping up the tension again until the next encounter. It's all about pacing and predicting player behavior.

The **scale of the level** is usually determined by the deadlines and the overall game's scale. Still, it's important to make estimates of different time values of the level and try to aim for those. How long does it approximately take to beat the whole level for a first-timer? How long does it take between certain segments of the level? The designer should make the estimates based on the experience they want the player to have, not just throw random numbers in there. One should think about the purpose of the level and how to pace its contents properly and base the estimates around that. But one shouldn't stress too much about the accuracy of the estimates, things might change during playtesting and iteration and some segments may have to be cut or added in the level based on fresh information. The initial estimates are there to give the designer a goal to work towards when making initial designs. [5] [7] [8]

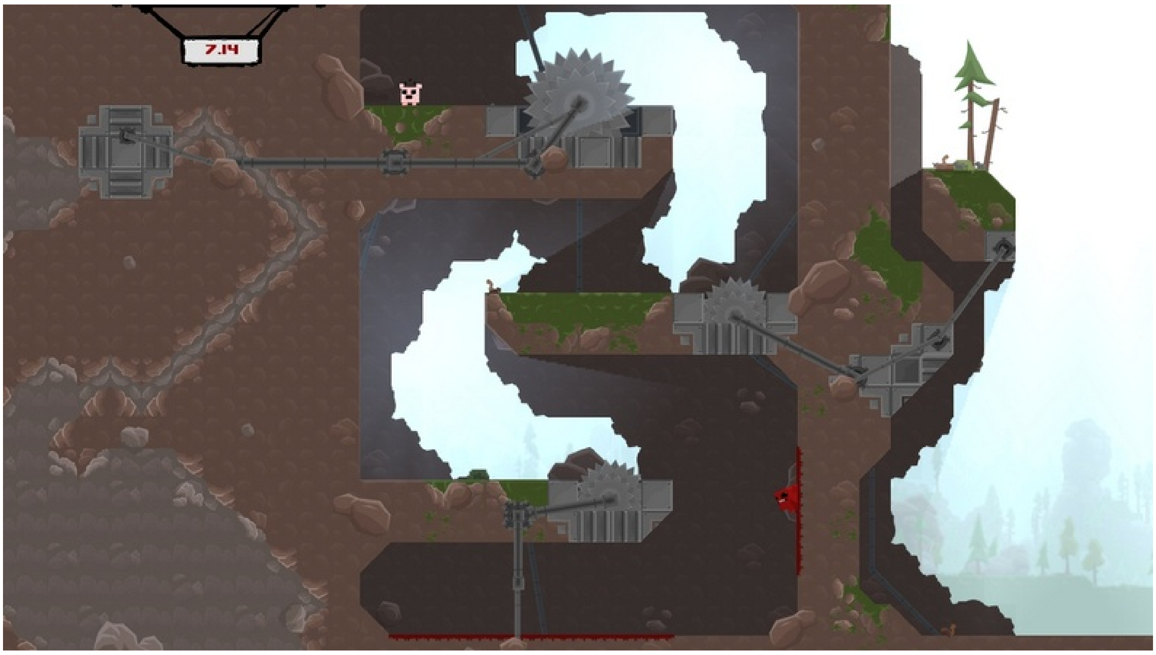


Figure 8. A screenshot of Super Meat Boy.

For example, the levels in Super Meat Boy are extremely short when compared to a lot of other games, usually lasting from 5 to 30 seconds. This is a conscious design decision, made because of the game's level of difficulty. All levels in Super Meat Boy are incredibly hard and players are expected to fail in them a lot. But by making the levels really short, the player never has to redo much of their progress after failing a level, making failures much less frustrating and encouraging experimental trial-and-error gameplay. Even the respawn time after dying lasts only about a second, so after a failure, the player is instantly set for another attempt at the short, but tough-as-nails level. This is why it's important to think about what scale would best fit the game's levels and its gameplay.

The level's place in the **level progression** usually dictates what can be expected from the player at this point. What mechanics have been taught to the player? How difficult challenges has the player experienced thus far? The challenge of the game should rise in relation with the player's proficiency with the game's mechanics. This is why it's important to keep in mind the level's position in the level progression to create a sensible difficulty curve for the game. [8]

Figure 9. A screenshot of Portal.

For example, in Portal, a first-person puzzle game, all levels teach the player something new within an otherwise familiar context. Either they teach the player new things about familiar mechanics or they introduce entirely new mechanics with familiar ones. The game gradually gets more and more complex with each level and in order to do this, each level has to consider what the player already knows. The designer cannot expect the players to pull off complex tasks with mechanics they're not yet familiar with.

**Metric constraints** refer to the player's abilities and their limits. What is the player's movement speed? How high can they jump? How large is their field of view? How far can the player shoot? The designer must be aware of these in order to both allow and deny the players of performing certain actions within the level. For example, if the designer doesn't want the player to be able to skip an entire segment of a level by jumping to an unintended area, they have to know the limits of the player's jump capabilities to create a big enough gap so that the player cannot jump to skip the segment. [7] [8]

The **history of the environment** relates to the theme of the level. What happened in the level's environment before the player arrives there? This mostly affects the narrative side of the level, which is not that important in some games. But if the game relies on its narrative, it's important to keep this in mind. It can help with environmental storytelling and keeping themes consistent. [5]

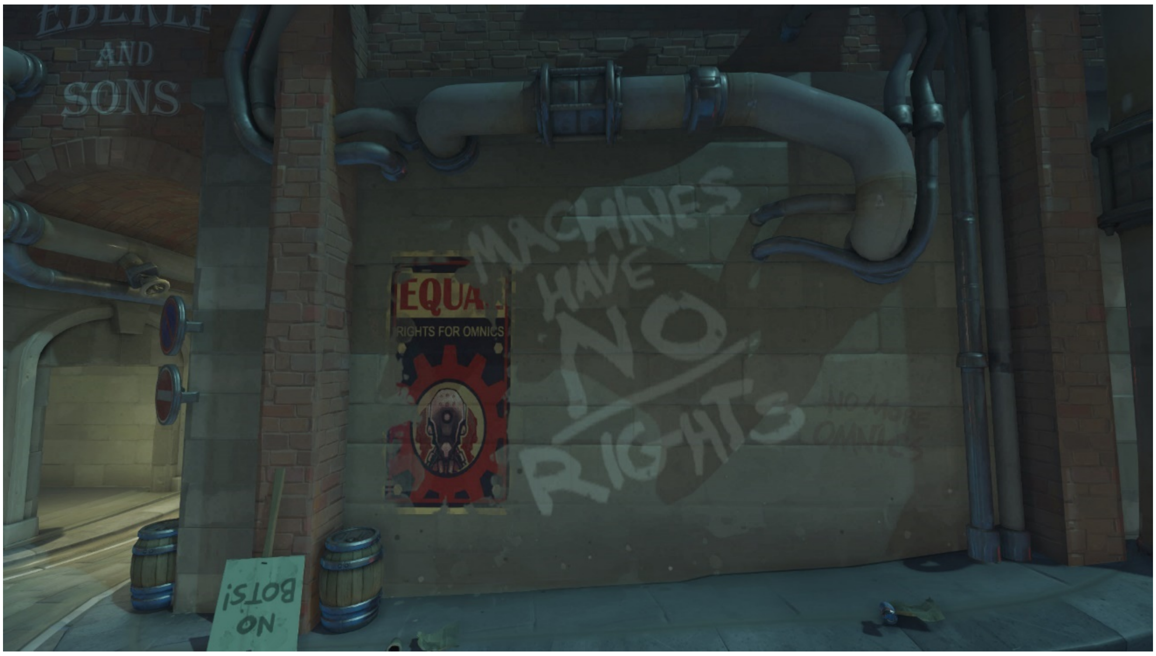
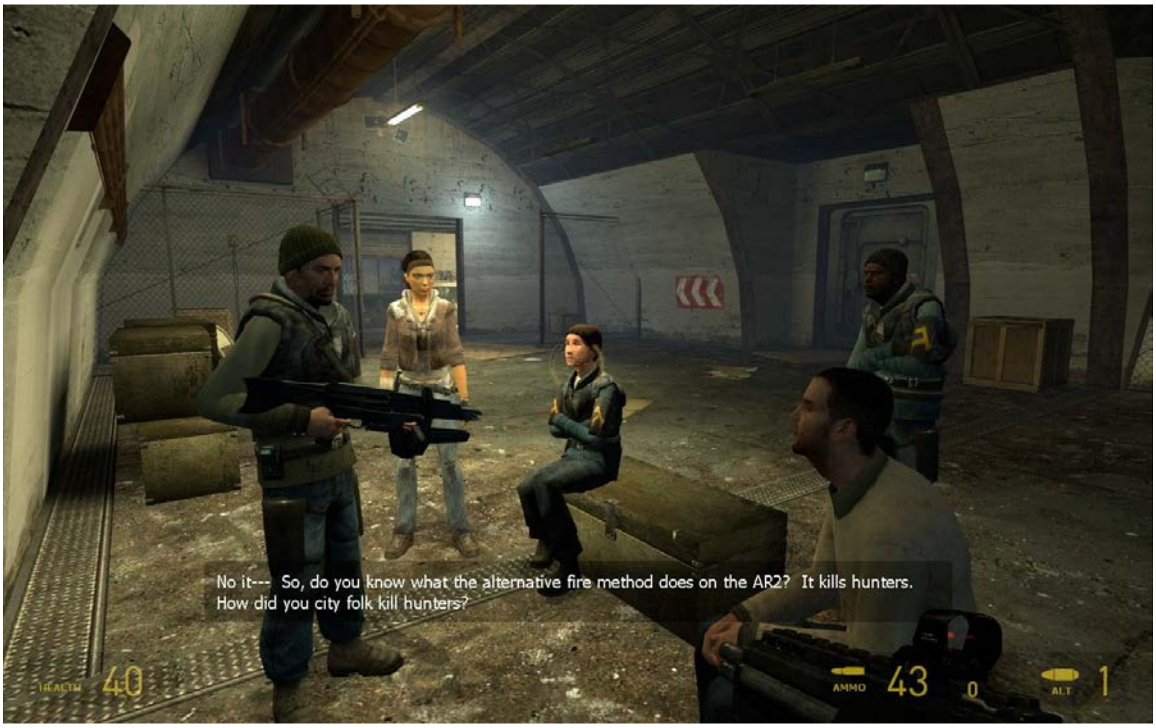


Figure 10. A screenshot of the King's Row level in Overwatch.

For example, Overwatch is a multi-player focused first-person shooter, which has very little explicit storytelling. However, each level in Overwatch contains a lot of implicit information about the game's background story. For example, in the picture above there are anti-robot graffiti and protest signs next to a pro-robot poster. This tells the player that there is clearly controversy about the rights of the robotic citizens of this city and there have been protests or even other conflicts on the matter. The level tells about its backgrounds not through explicit telling, but through the actual environment.

Besides its history, what other **narrative elements** does the level have? Is the designer trying to tell a story through the environment? Will there be NPCs to interact with and support the narrative? Will there be linear scripted sequences to tell a story with? What is the overall message the designer is trying to convey with the level and how does it fit with the rest of the game's narrative? Again, this is not that important in games with less narrative focus, but it is still important to determine the methods one wants to convey their narrative to the player with, even if the narrative focus of the game is not that big. [8]



No it--- So, do you know what the alternative fire method does on the AR2? It kills hunters.
How did you city folk kill hunters?

Figure 11. A screenshot of Half Life 2: Episode Two.

For example, Half Life games convey most of their story through scripted scenes and NPC dialogue. However, the game never freezes or takes control away from the player entirely, keeping the gameplay interactive even while NPCs are talking.

The **visual trademark** of the level is what people will remember the level for. It is usually the level’s theme or a huge landmark of some sort, but it can also be something unique that the player does in the level. A visual trademark gives the level an identity and such levels will feel less similar to each other. [5]

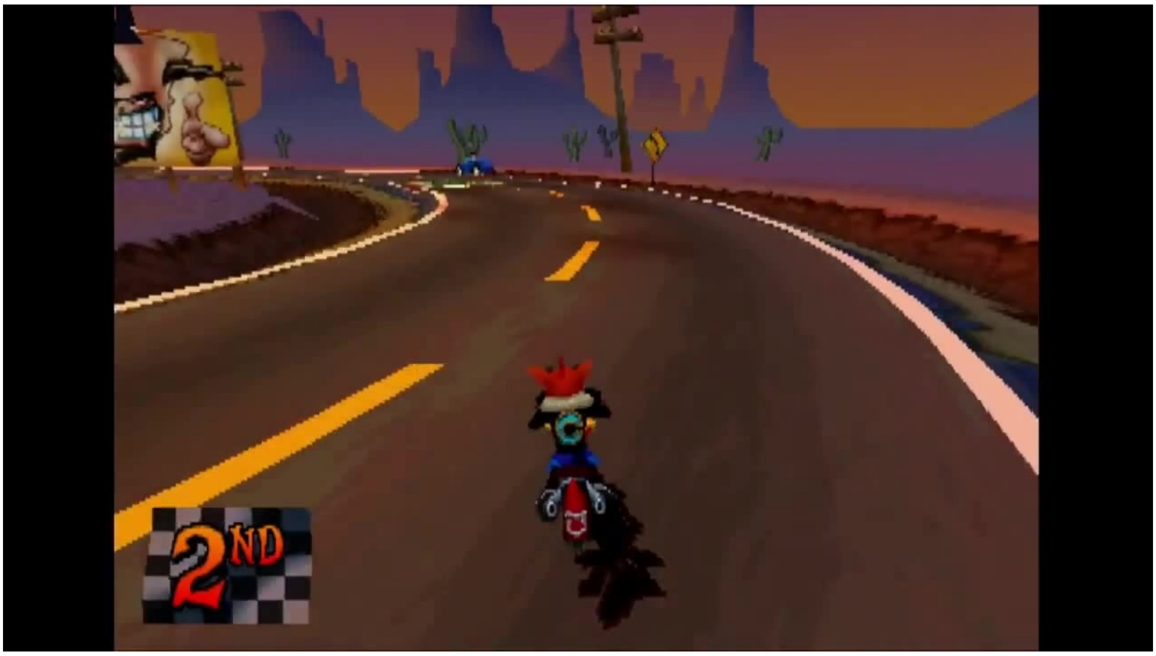


Figure 12. A screenshot of Crash Bandicoot 3: Warped.

For example, the motorcycle levels in Crash Bandicoot 3 have a clear visual trademark. It’s why they are called motorcycle levels in the first place! Nowhere else in the game does the player ride a motorcycle to complete levels, giving these levels a distinct visual trademark. And even though there are four motorcycle levels in the game, all four of them also have something unique to remember them by, most notably the time of day. The first level is a race during daytime, the second one at night, the third one at dusk and so on. It’s much easier to recall the levels with these kinds of visual trademarks.

**Set-pieces** refer to the smaller memorable parts of the level, giving identity to the different segments within the level. These can also be smaller landmarks or thematically something different than the rest of the level. Like a tiny village in a level that is otherwise nothing but a thick jungle. Like before, it can also be a segment where the player does something completely different, utilizing different game mechanics. Or it can be a narrative element or anything to make that segment feel a bit different from the rest of the level. This helps the player remember the level as a whole and create mental maps of the area, easing navigation. [7] [8]

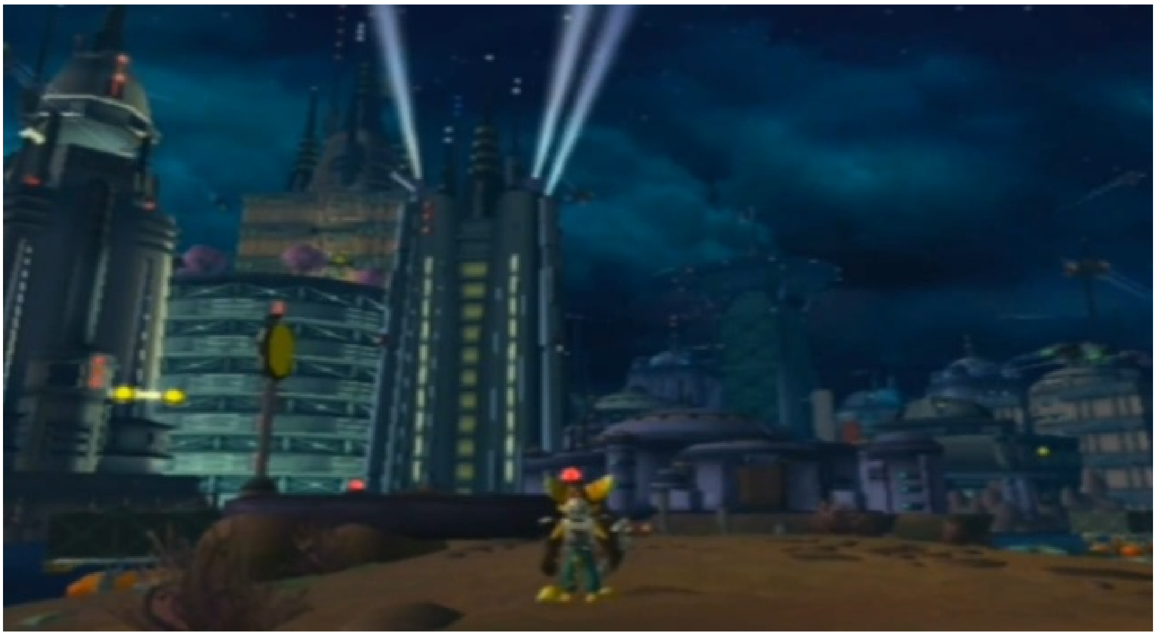


Figure 13. A screenshot of Ratchet & Clank.

For example, the Blackwater City level in the original Ratchet & Clank has three major set-pieces; the city area, the hoverboard race and the sewers. All of these areas offer different kinds of gameplay and also completely different layouts in terms of level design, making them all memorable. The city area is a combat-heavy area with a lot of run-and-gun gameplay. Once the player clears the city, they reach the hoverboard races, which offers a whole different type of gameplay where the player has to race to win with a hoverboard. And finally, the sewer area, which is heavy on puzzles at first, but then escalates into a high-action platforming sequence once the sewers start to fill with water and the player has to get out before they drown.

And finally, **how the level will look like** will most likely be determined by the level's theme. But just knowing the theme is necessarily not enough. It really helps to draw some inspiration out of reference imagery from websites like Flickr or Pinterest. It can help the designer envision the different views in the level or get around mental blocks. [5] [7]

Figure 14. Concept art from Jak & Daxter.

Bigger studios have concept artists who bring the team's visions to life with vivid imagery. Level designers can then use this imagery as a guide to how the level will look like and work from there. Smaller studios rarely have this luxury, so level designers must either decide the level's looks with the art lead or by themselves. Whenever professional concept artists are not available, gathering mood boards from Flickr or Pinterest can work just as well.

Once all this is figured out, the designer should have a good bunch of clear dos and don'ts. The slate is no longer blank, it is filled with guidelines to help design a level that best serves the game. [5] [6] [7] [8]

## 3.2 Designing level structure

Once the most common constraints have been covered, the designer should have at least some kind of an idea of the level they are going to make. How to proceed from here is up to them, since there are as many methods as there are people. Some methods work better for others and it's up to the designer to find the one that suits them and their needs. But coming up with a method from scratch may be difficult, so I did some research on the most common ways to begin designing a level and its structure.

Unless the designer already has some ideas for the level's contents, **brainstorming** is usually a good place to start. Brainstorming alone or with a team while keeping the known constraints in mind is a good way of getting some initial ideas for the level. There are multiple different ways of brainstorming, but I won't go into the details of brainstorming techniques in this thesis.

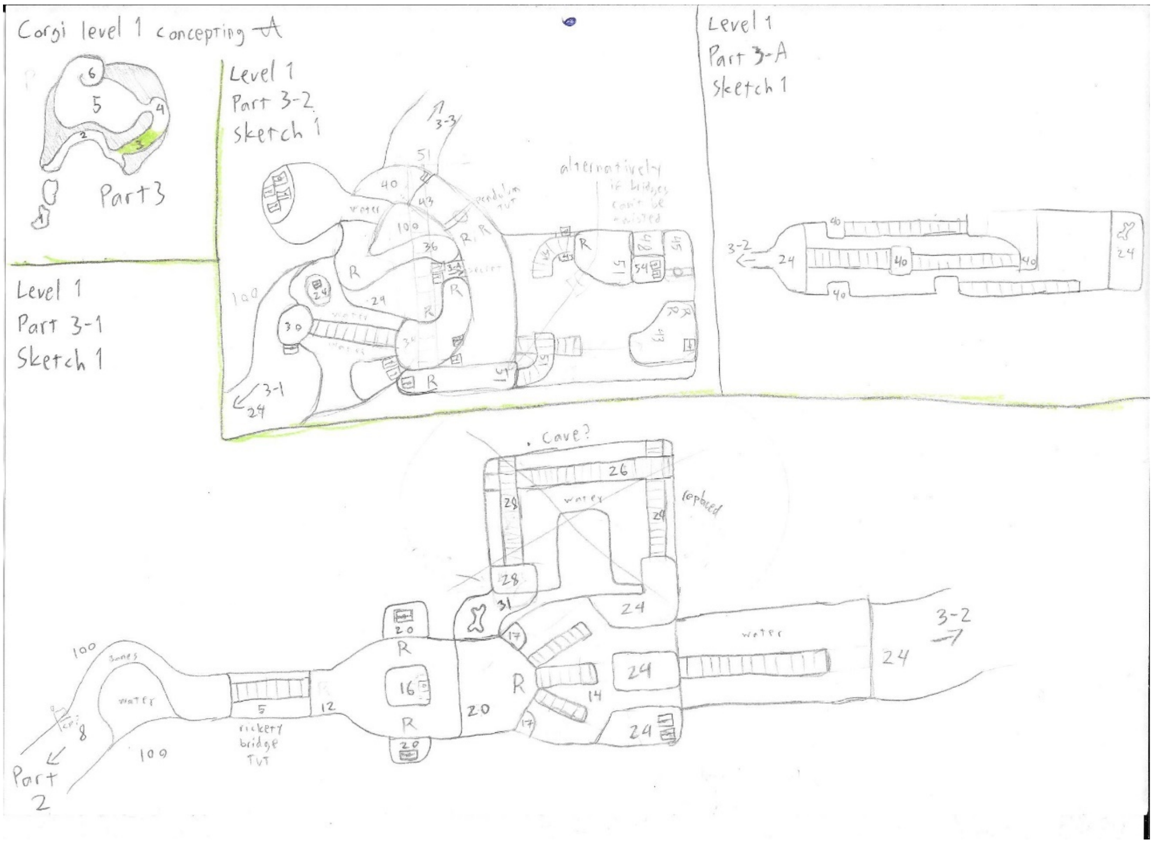


Figure 15. Rough sketches from Captain's Tail.

Once there are some ideas for the level's contents and set-pieces, **rough sketching** is the most common way to go from there. Some people prefer to skip this phase entirely, but most people do not recommend doing so, unless the designer is absolutely sure about what they are doing. Rough sketches refer to a top-down or a side-view map of the level, depending on the game. They are usually made either with a pen and paper or on computer software like Adobe Illustrator. The point of this phase is to help the designer perceive the level better and give them a clearer goal to work towards when working in the level editor. Rough sketches can also be used for communicating ideas to the rest of the team, so it is important to keep their readability in mind. One should not get too caught up on the tiny details of the level at this point. The designer should sketch only the most essential things that they will need when building the level. Determining what’s actually essential and what is not can be difficult at first, but it’s something designers get better at through experience. [6]

In addition to rough sketches, the designer can also try nailing down the level’s structure using **bubble diagrams**, a technique that an Insomniac Games employee, Mike Stout, uses. Once the designer has some basic ideas for the level, they should narrow them into different “areas” of the level, each with their own role within the level. For example, area 1 introduces the player to a new type of enemy, area 2 forces the player to deal with the new enemy in addition to some familiar enemies, area 3 serves as a pace breaker with some different gameplay and so on. Stout recommends dividing levels into seven areas, due to the rule of seven, which refers to a pacing technique that I will elaborate more about later. Once the designer has some areas, they should then connect those areas into a bubble diagram, to help them get a grasp on the level’s structure and the relations between different areas.

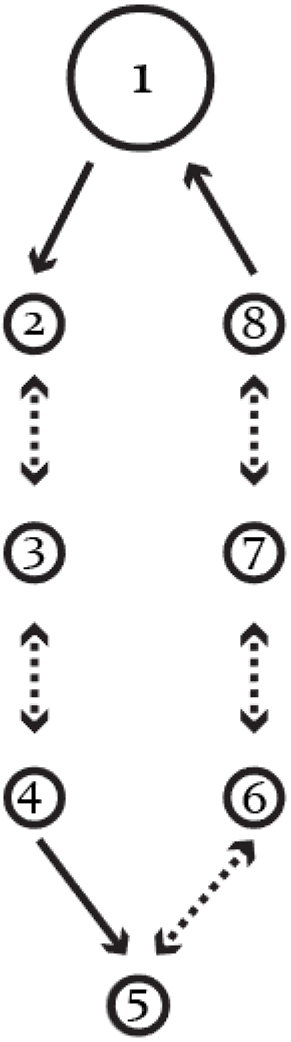


Figure 16. An example bubble diagram by Mike Stout.

Once the designer has finished their bubble diagram, they can start the previously mentioned rough sketching by sketching each area individually. Once the sketches of each area are done, the designer should link the sketches together according to their positions in the bubble diagram. The designer should sketch the links between the areas and remember to include well-placed rest spots here and there to ensure good pacing. I will go into more detail about pacing a level later. Once that's done, the designer should have a clear map of the level that they can now start building in a level editor. Stout also emphasizes that this is just one method and designers should only use it if it works out for them. [8]

And last but not least, there is the **visual timeline**, a technique used by Pete Ellis, an employee at Guerrilla Cambridge. A visual timeline is first and foremost used to plan the level's pacing, but it can also be used to design the level's overall structure using gameplay beats. According to Ellis, gameplay beats are the main events and activities within the level, such as set-pieces or introducing new mechanics. The visual timeline also includes the level's narrative beats, which is the storytelling within the level. In his visual timeline example, Ellis utilizes the tried and true narrative model from movies called the three-act structure. While it generally has been used in movies to develop drama, it can also be used in games to pace a level's contents, whether they're gameplay-centric or narrative-centric.

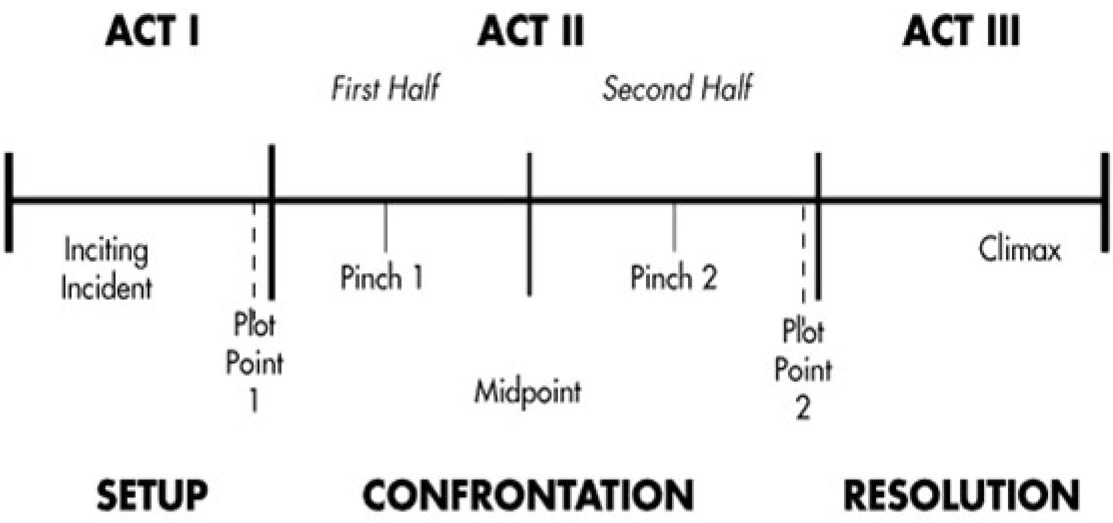


Figure 17. The “three-act structure” narrative model.

Using this three-act structure, the designer can place the level's contents on the timeline to build the level's structure and pacing. They can also see where the level would need less intense gameplay and where the level would need more intense gameplay.

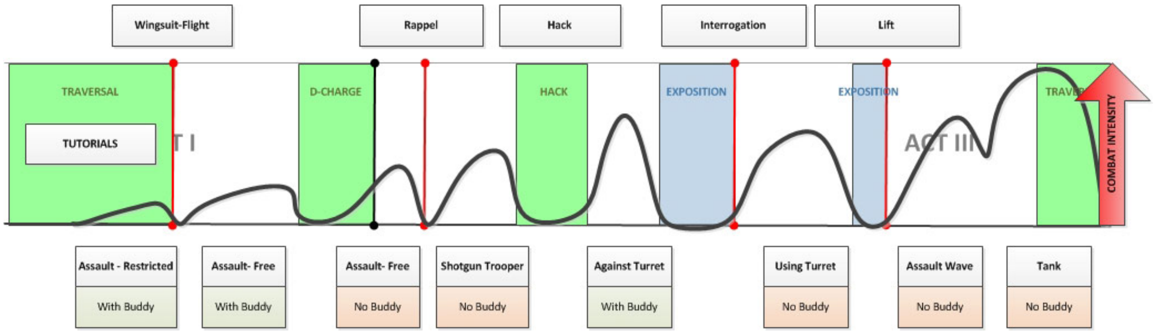


Figure 18. Pete Ellis' visual timeline of KillZone: Mercenary's first level.

The visual timeline works best on extremely linear games, because there's less room for freeform player interaction within the level that could affect the pacing. It is also recommended to utilize rough sketching or other planning methods in addition to the visual timeline, since the timeline only helps one plan the pacing between the level's contents. [9]

The designer can utilize any of the aforementioned methods or even all of them while planning, but they should avoid using them if they have noticed through experience that they are not working out for them. It is also important for designers to experiment and find their own methods, but it's always easier to find them once they have tried some existing ones. Next, it is time to elaborate on the nitty-gritty details of level design that can help designers in the initial design phase as well as during iteration.

### 3.3 Navigation and player guidance

One of the most important things about a level is that it should be fun to navigate. [23] In order to achieve this, the level should make navigation as intuitive as possible. If players feel lost and have no idea where they should be going to proceed, they will most likely get frustrated and quit the game. To avoid this, there are plenty of tricks designers can utilize to guide the players through environments, regardless of game genre.

The key to this is utilizing **contrasts**. Human eye is naturally drawn to contrasts in our field of vision, because our brain cannot process everything in our field of vision at once. Creating a contrast in the environment draws the player's attention and subconsciously guides them towards it. Here are a few examples on how different kinds of contrasts can be created in game environments to guide the player. [10]

A lot of games use **color contrasts** to attract the player's attention to mechanically important objects in the environment and also to guide the player around. It's important for designers to think about the levels' color palettes, so that they can find effective contrast colors that they can use to guide the player's attention whenever needed. [10] [15]

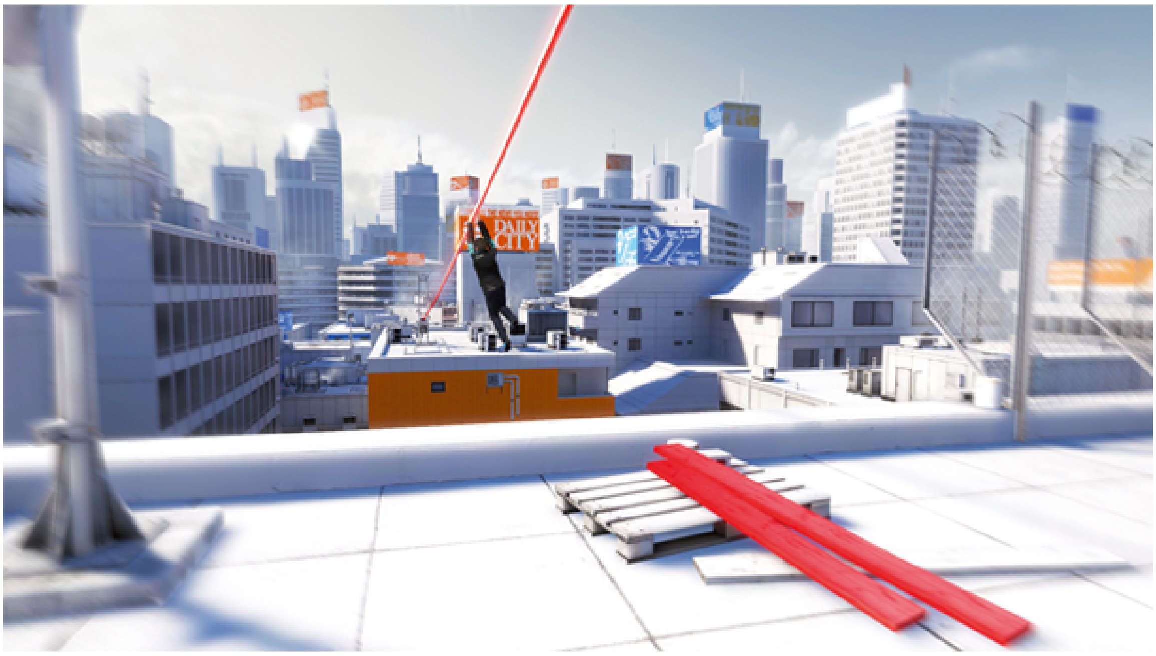


Figure 19. A screenshot of Mirror’s Edge.

For example, Mirror’s Edge uses the color red to highlight objects that the player can perform parkour actions with. Eventually the player learns to instinctively follow any red objects, which are used to guide the player into the right direction on the rooftops of a huge city. [15]

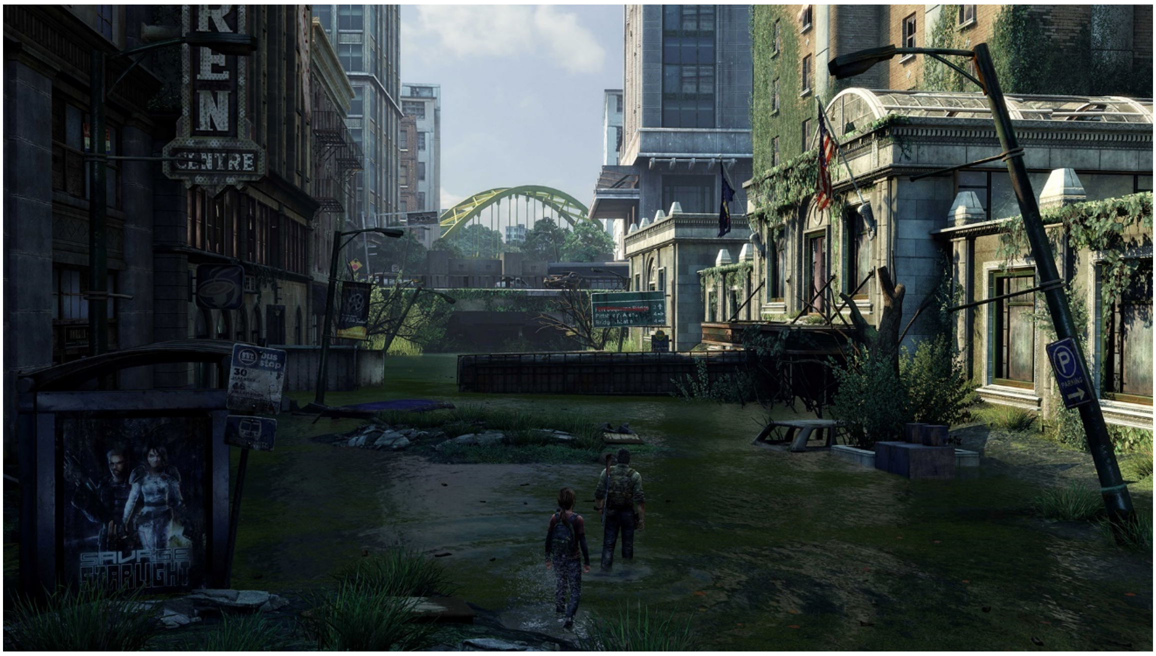


Figure 20. A screenshot of The Last of Us.

As another example, the survival action game The Last of Us uses the color yellow to highlight important landmarks as well as smaller waypoints in the environment. Yellow is rarely used elsewhere in the game world, so it works well as a bright, contrasting color. [10] [15]

Game environments usually consist of geometrical shapes and forms, regardless of whether they're in 2D or 3D. This is why **geometry** is a very universal player guidance tool in level design. A common way to do this is to create a conflicting vertical shape in a mostly horizontal environment and vice-versa. [7] [15]

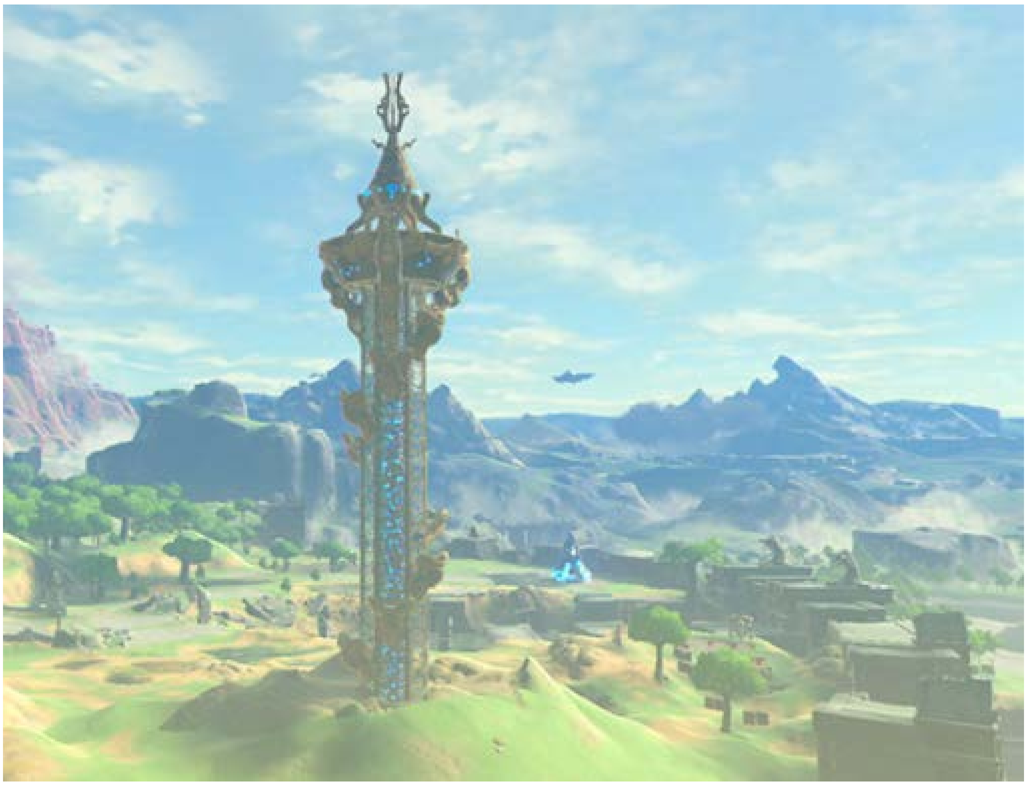


Figure 21. A screenshot of The Legend of Zelda: Breath of the Wild.

For example, in The Legend of Zelda: Breath of the Wild's vast open world there are these tall Sheikah- towers which the player has to climb in order to receive a map of the surrounding area. The reason these towers pop out so well is because there is nothing else so steeply vertical in the environment. They are easy to spot even from a distance and they are usually placed so that once the player has climbed to the top of one of these towers, they can already see two or three other towers in the distance. This way the player always has a clear goal to work for, without needing to explicitly tell them a thing.

Designers can also use geometry to create guidelines that can point in the direction they want the player to go. These can be literal arrows or guidelines, but in most cases, they are embedded into the natural objects or shapes within the environment.

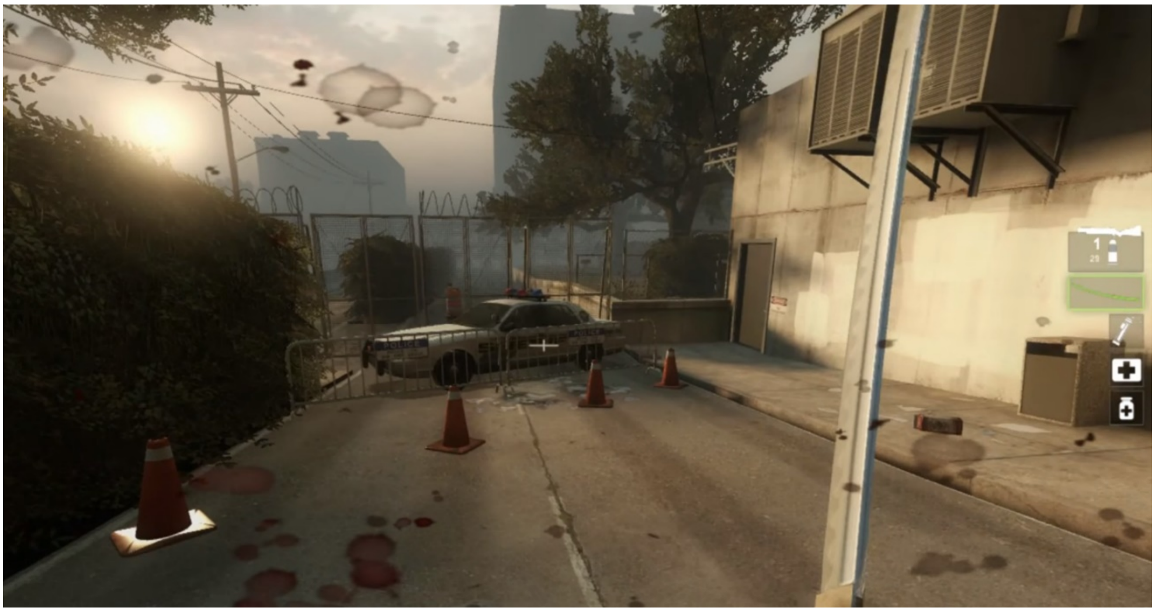


Figure 22. A screenshot of Left 4 Dead 2.

For example, in this particular scenario from Left 4 Dead 2, the player is being hinted towards the doorway on the right. The slanted police car and the fence in front of it are placed so that they create a guideline towards the door. In addition to that, they’ve used repeating shapes or patterns to further strengthen the guideline effect: the four traffic cones form a guideline of their own towards the door. The designer can create similar patterns by using identical shapes in a row to create an invisible guideline towards something. It should be made sure that the environment’s geometry isn’t too cluttered, otherwise such clever guidelines can get totally lost within the player’s view. [15]

It is possible to create **lighting contrasts** by using the relationship between light and shadow. A light at the end of a tunnel naturally attracts us humans, because our eyesight was not designed for darkness. We seek light, because we do not feel safe covered in darkness, not knowing what may be lurking. It is a base survival instinct and therefore easily exploitable. [12] [16] [17]

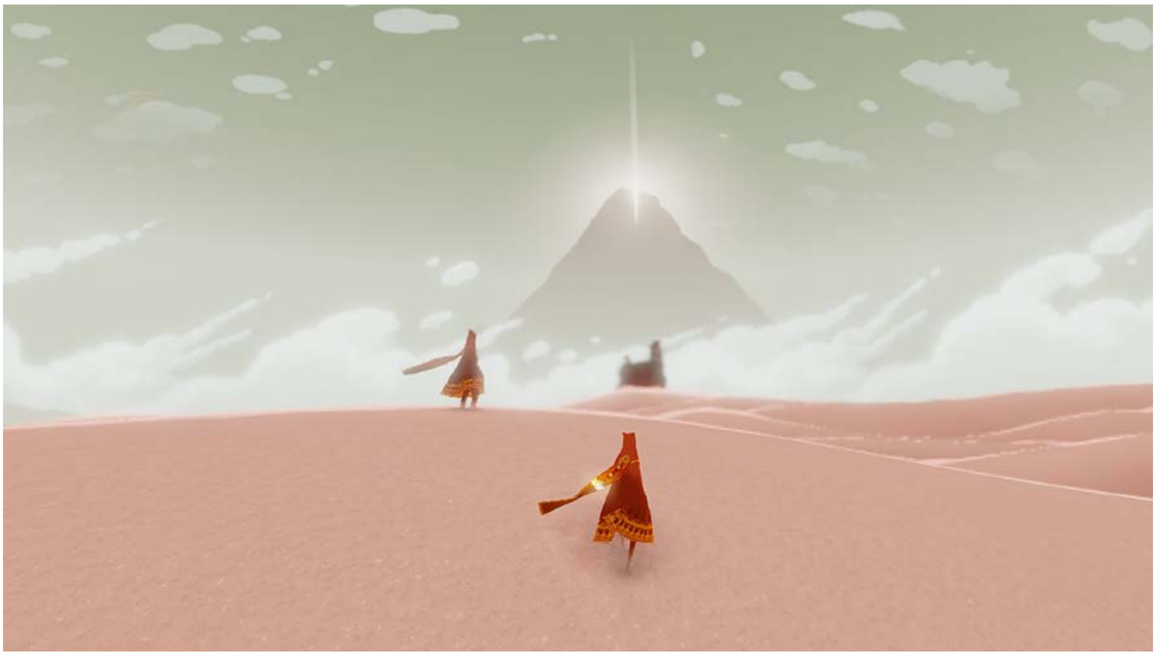


Figure 23. A screenshot of Journey.

For example, the end goal of Journey is to reach the shining bright light at the top of a distant mountain. It is the first thing the player sees in the game and it is never explained what the bright light is or why the player would even want to go there. Despite that, all players instinctively follow the bright light to find out what it is. In addition to that, the light sits on top of a huge vertical contrast in a vast horizontal desert and the sides of the mountain point towards the light at the top, creating a highly effective multi-layered contrast.

Because most environments are usually static, some **motion** can easily create a contrast for the player. Whether it's birds flying towards the player's objective or the door of a house suddenly blasting open, movement is sure to catch the player's eye as long as the background is static enough. [15] [19]

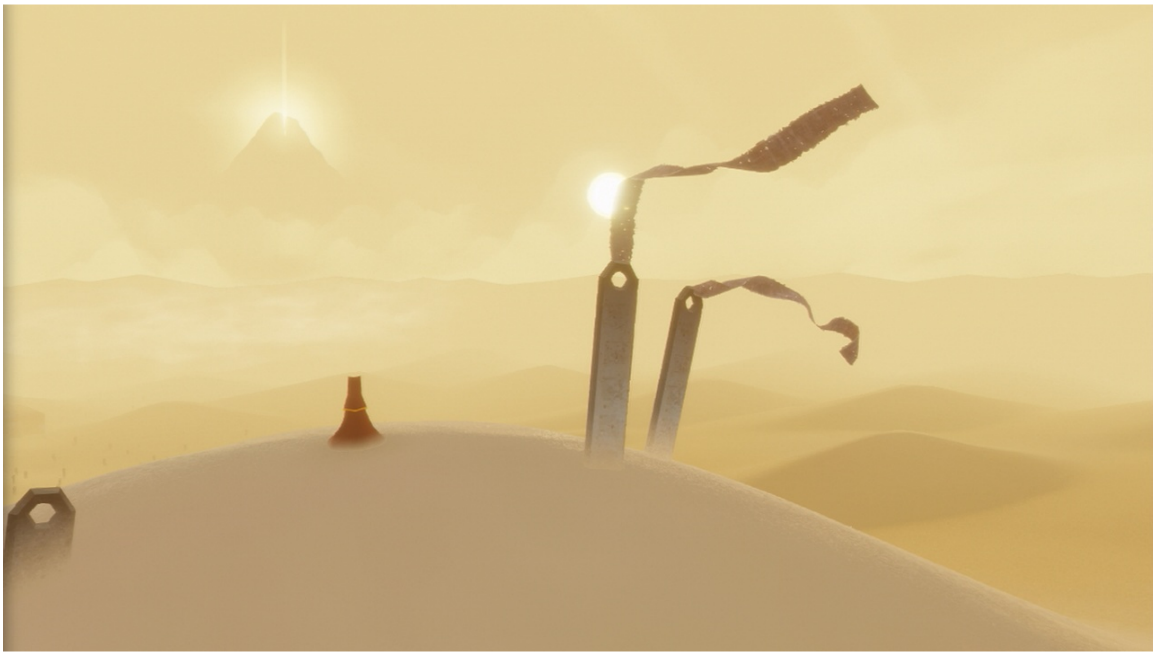


Figure 24. A screenshot of Journey.

For example, Journey uses these stone pillars with clothes attached to them to guide the player around the desert. The clothes are in constant motion due to the wind, which allows them to pop out of the desert environment really well.

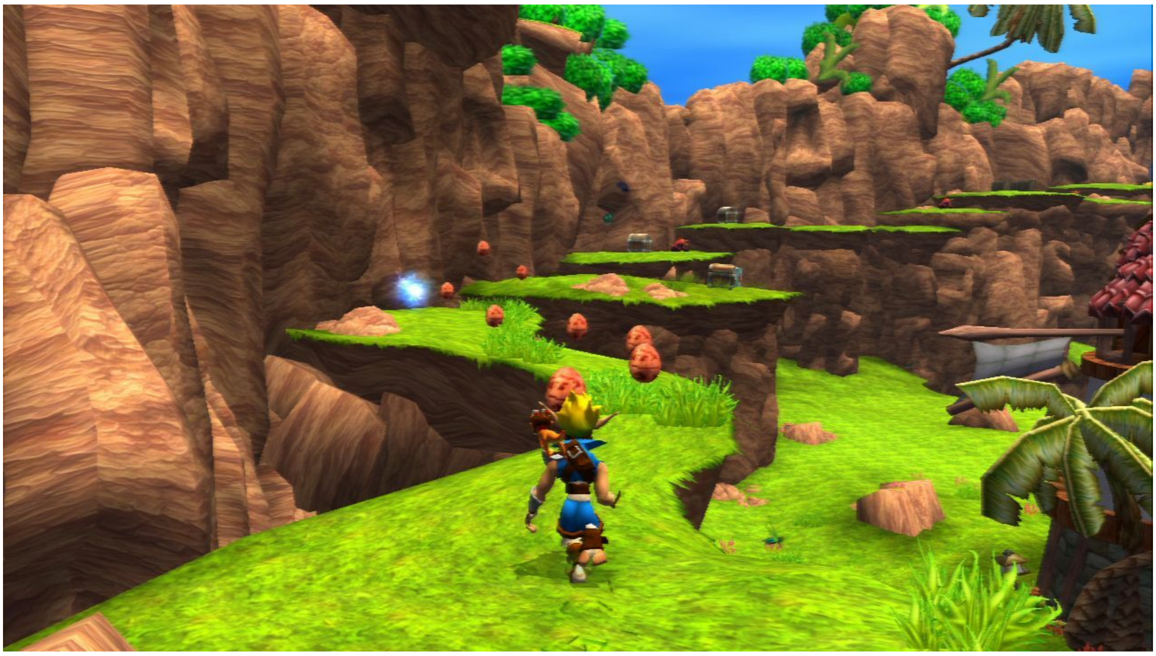


Figure 25. A screenshot of Jak & Daxter.

Most collectibles or other important objects in games also have constant motion attached to them. For example, in Jak & Daxter games the player has to collect these red precursor orbs to advance in the game. The orbs float slightly up and down while rotating constantly, ensuring that the player always notices them within their view.

Using the previously described techniques, designers can create **landmarks** in their levels, such as the Sheikah towers in The Legend of Zelda: Breath of the Wild. The point of landmarks is to help the player perceive their current location in relation to the rest of the environment. The player should be able to find and recognize these landmarks whenever they get lost so that they can continue navigating through the level. Landmarks can also create a sense of progression, which helps the player enjoy their trip to their eventual goal. Players find relief in knowing that they are advancing in the right direction instead of having to wander aimlessly without a clear goal. [7]

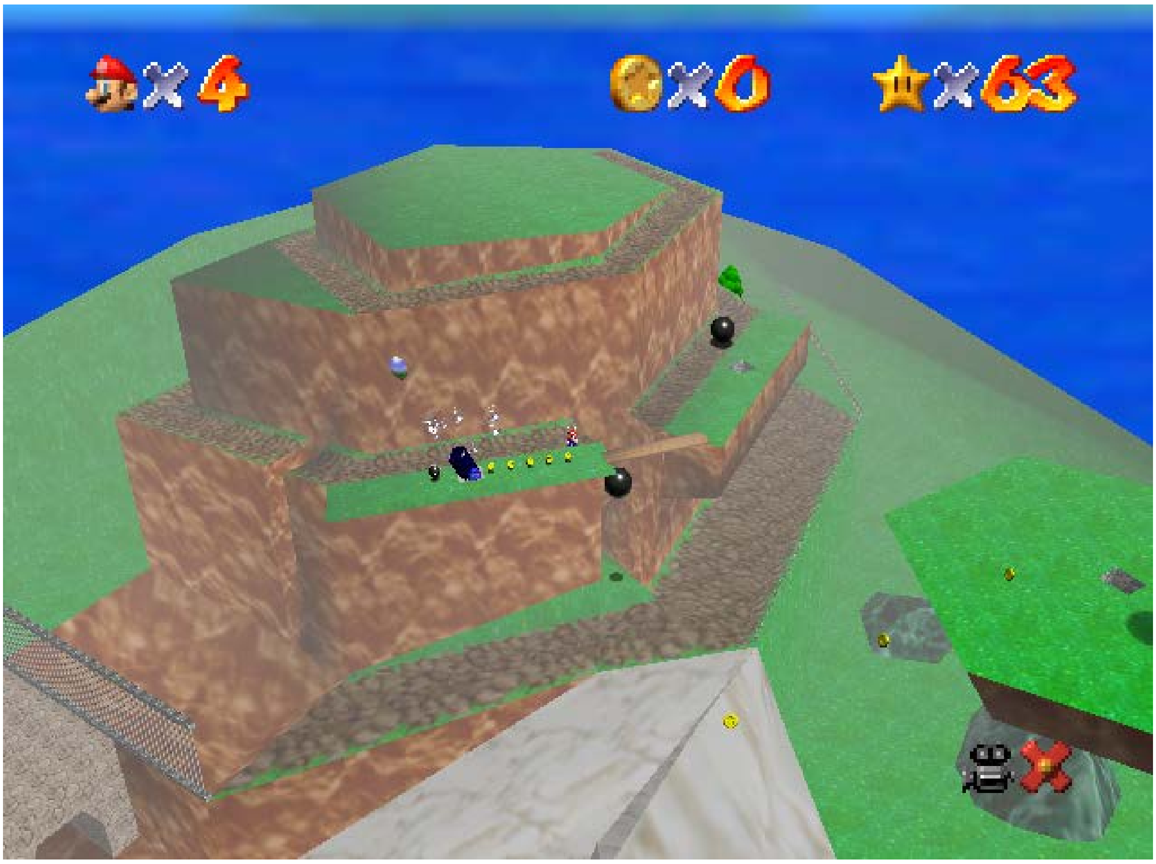


Figure 26. A screenshot of Super Mario 64.

For example, in Super Mario 64’s Bob-Omb Battlefield level, the player’s goal is to reach the top of the battlefield’s mountain. The mountain is so huge that it can be located from anywhere in the level, easing player navigation.

Also, the designer should keep in mind that if there are too many contrasts in the player’s view, none of them will grab their attention, because the contrasts are now the new normal. It is important to use them sparingly and avoid creating environments with too much visual clutter. [15]

## 3.4 Visual language

Relating to the aforementioned contrasts, visual language refers to communication using visual elements. [24] Using contrasts to guide the player is a form of visual communication, but it can go even further than that. In addition to creating a contrast in the environment, if one creates a clear visual language on what certain patterns mean, it is possible to do so much more than just guide the player around the environment. For example, color coding. If the designer establishes a clear rule that all the enemies in the game sport a vivid red color and the environments are designed so that red color always pops out really well, the player can identify red objects from the environment easily and during the same split-second they can also identify that they are also enemies. However, in this scenario the designer has to be careful about coloring anything else red, because they have established a rule to the player that all enemies are red and that red color signals about danger. By keeping the rules consistent, designers can communicate incredible amounts of information to the player without speaking a single word. [10] [11] [13] [14]

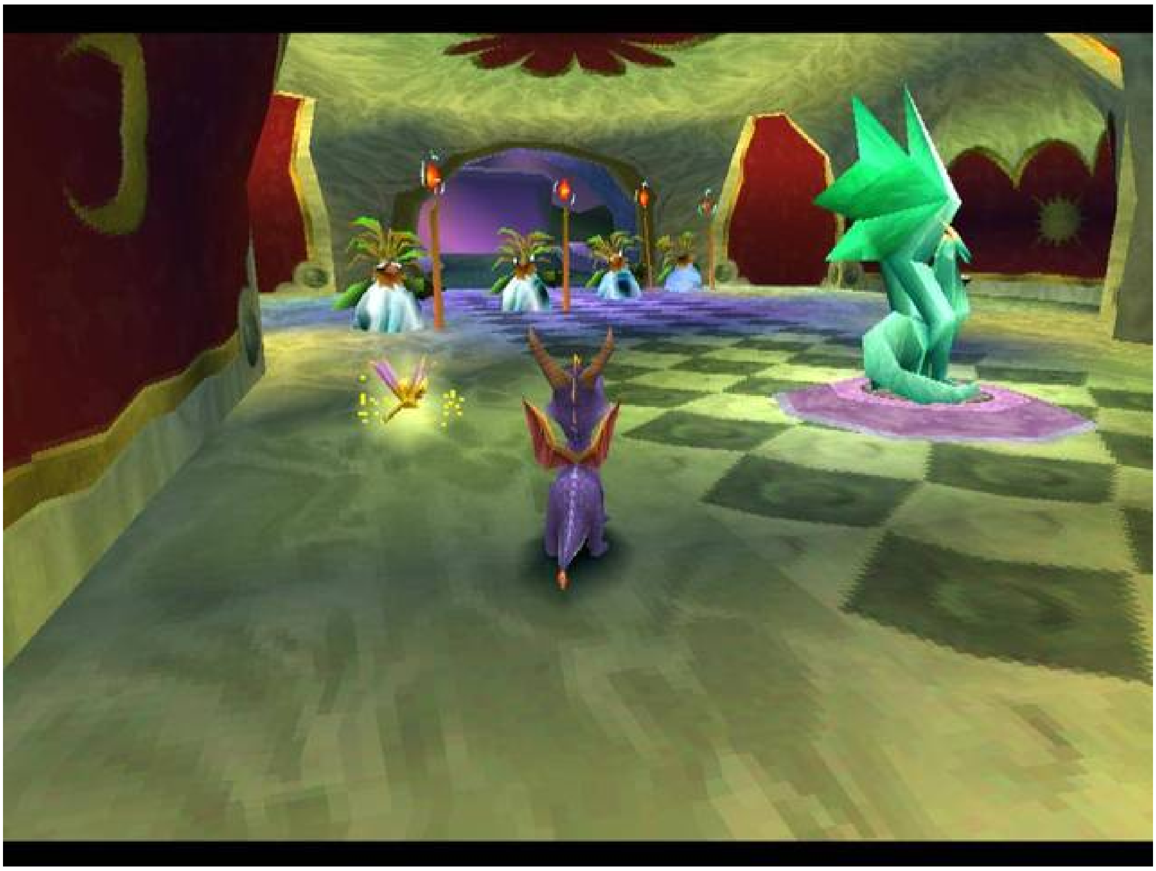


Figure 27. A screenshot of Spyro the Dragon.

For example, Spyro games have an early on established rule that Spyro’s flame breath does not affect anything metallic, however metallic objects can be attacked by charging at them. This applies to the environment and the enemies as well. Another established rule is that any large enemies cannot be defeated by charging at them, but unless they are wearing metallic armor, they can be defeated with Spyro’s flame breath. The games proceed to mix and match different types of environmental mechanics and enemies that follow these set rules and players learn to play around those rules quickly, without needing to explain them explicitly.

## 3.5 Composition and framing

Composition is the art of placing terrain and objects in a specific way to present a clear **view** for the player. At certain points in a game, the designer might want to draw the player’s attention towards something, for example to guide the player through the environment, as discussed earlier. And sure, contrasts are necessary to make things pop out of the environment, but by composing the contrasts properly into the player's view the designer can make absolutely sure that the player won't miss them. Designers can also use composition for other purposes as well, such as showing off the game's art, providing implicit storytelling or teaching the player a new mechanic. The point of composition is to make the player's view as harmonious as possible and easy to take in. [18]

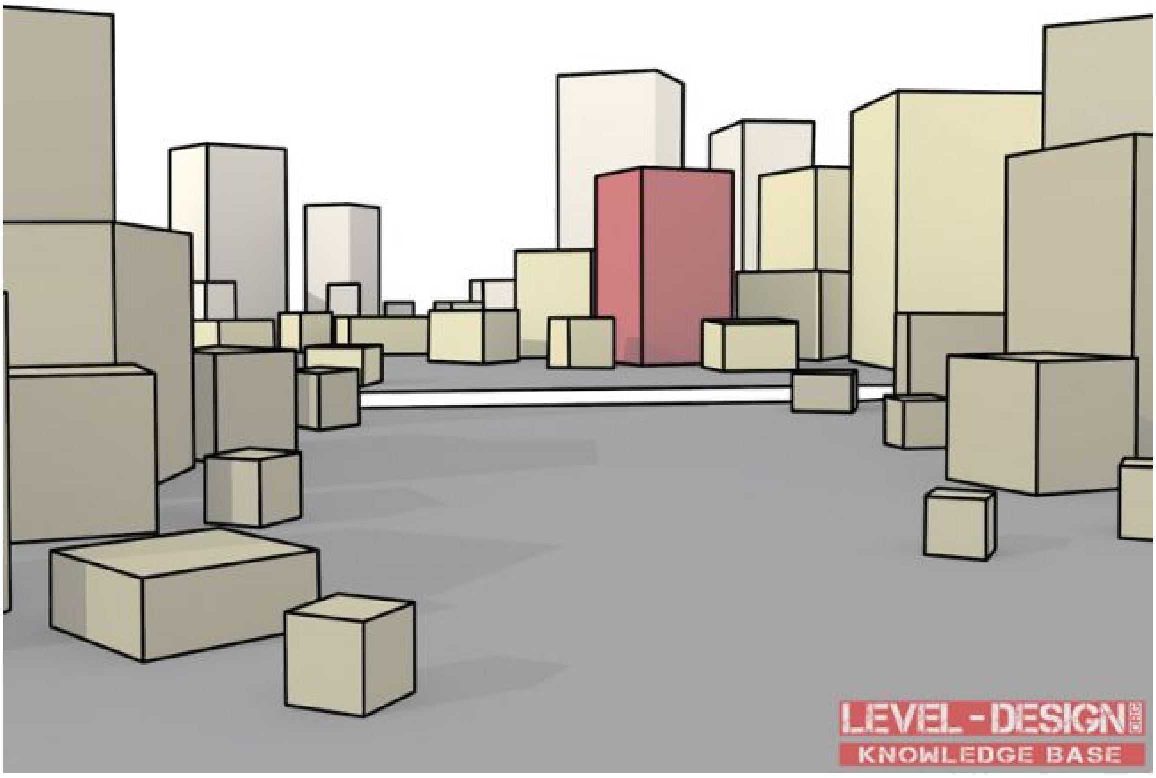


Figure 28. An example of a focal point.

First, the designer will have to determine the **focal point**, also known as a dominant, because it should be the most dominating element in the view. Focal point is the thing the designer wants the viewer to focus on in their composition and everything else in the view should guide the viewer's eye to it. It is possible to have more than one focal point in a single view, but the same rule of contrasts applies here as well; too many focal points in one view will cause the view to become chaotic and unclear.

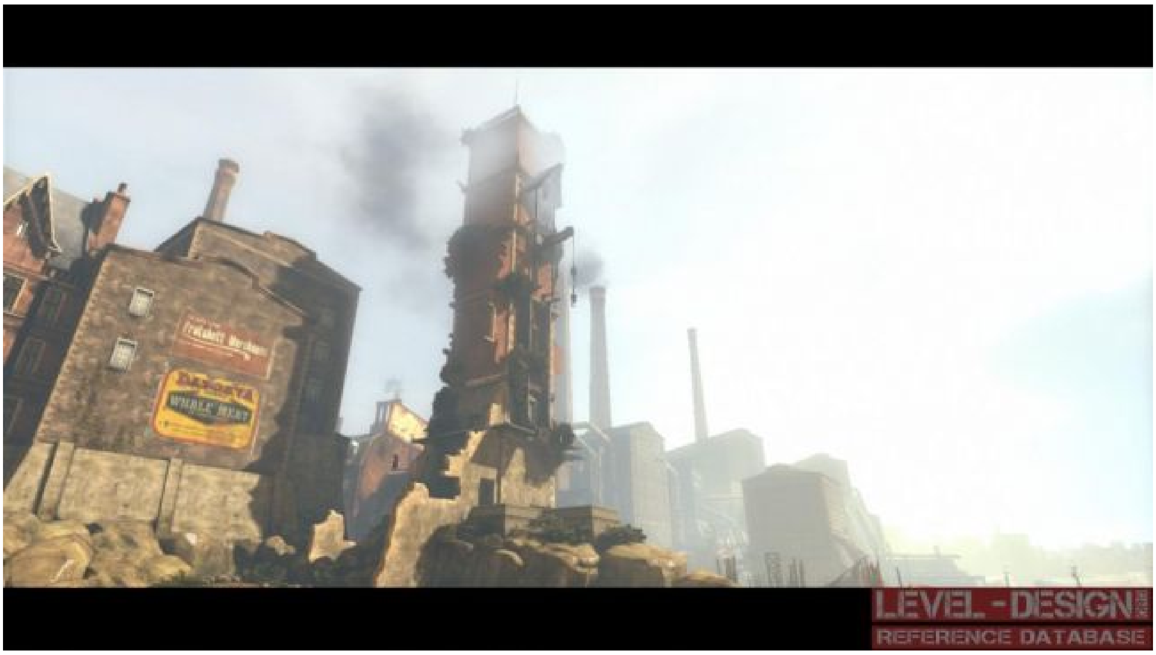


Figure 29. A screenshot of Dishonored.

Designers should also make sure the focal points in the player's view view pop out properly. They should contrast the rest of the environment with their color or shape. When composing, designers should be mindful of the background; focal points should be given a background that will emphasize their silhouette. Also, designers should be aware of the foreground; it is important to make sure there is nothing blocking the player's line of sight to the focal points. Designers should make the focal points as attractive to the human eye as possible. [18]

But what if the player is not even looking at the view that the designer composed? Unfortunately, games are an interactive media, so designers cannot control the player's view at all times. We can however place some assured key views within levels where all players are bound to arrive sooner or later. These are called **observation spots**. By placing linear paths which the player must traverse in order to move to the next area, the designer can compose a specific view during the linear path or at the end of it, because all players are bound to end up in the same observation spot. [18]

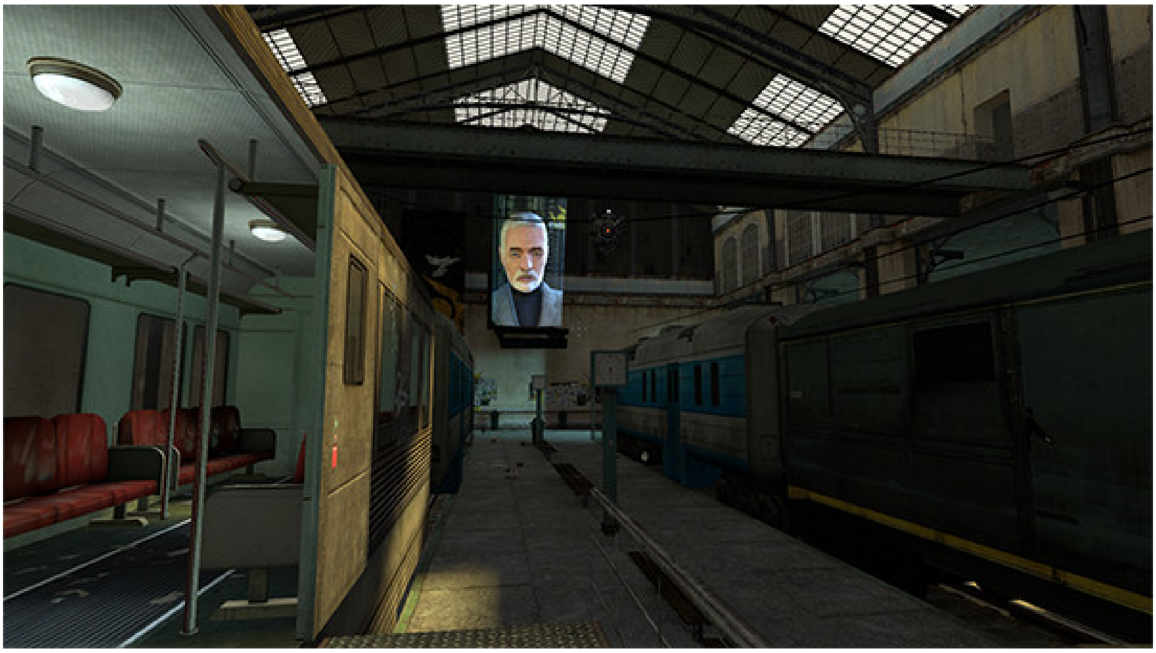


Figure 30. A screenshot of Half Life 2.

In this scene from Half Life 2, the player is arriving at a train station and is exiting the train. The player is in complete control of their character and view. However, in order to enter the train station, the player will have to turn 90 degrees to the left and walk between the narrow platform between the two trains. There is no other way to go, so the view here is absolutely assured for each player, allowing the level designer to compose this view so that the player's attention will be aimed towards the large screen at the top. [19]

Designers should also try to think if there are other points in the game where all players will share the same view. For example, when players start a level, they will most likely all start from the same position. This is the designer's chance to compose a very specific view at the start of a level. Designers can show what the level's goal is, show some elements that define the level or show some interesting narrative content. [19]

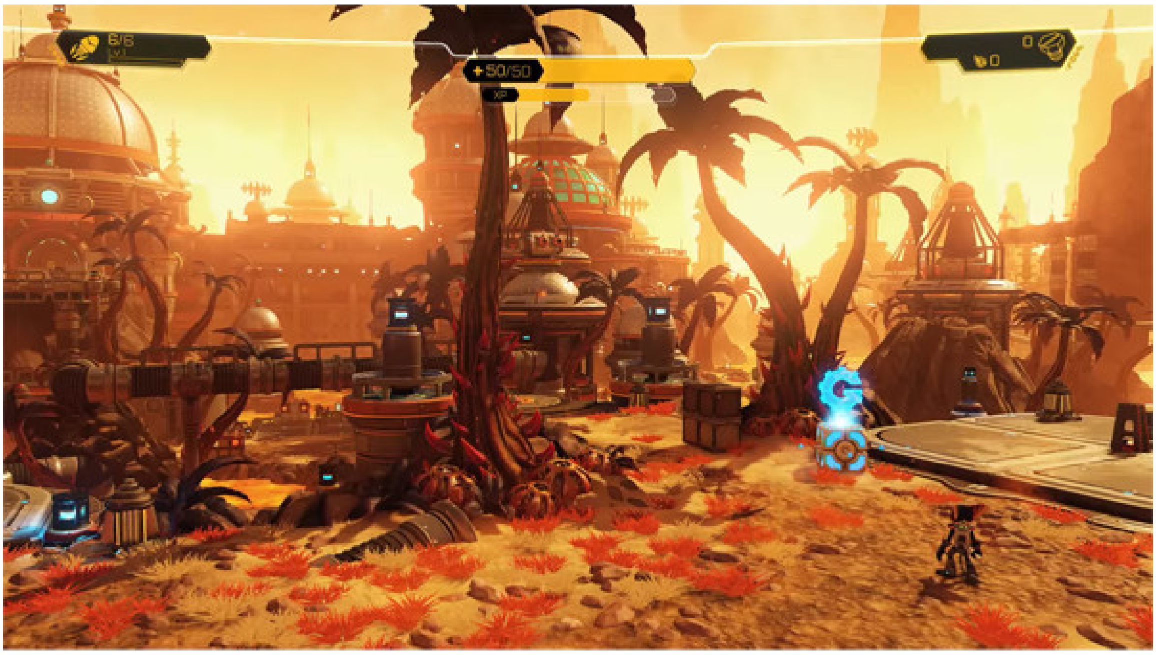


Figure 31. A screenshot of Ratchet & Clank (2016).

In Ratchet & Clank games, the player travels from planet to planet during their adventure. Whenever they arrive on a new planet and exit their ship, the view they get is always the same. All planets in Ratchet & Clank games have a distinctive **opening view**, usually directing the player's attention towards the level's goal and providing a complete overview of the level's theme. [19]

To sum it all up, designers should look for opportunities in their levels to provide the player with composed views. Composed views can be used to help guide the player or to showcase the level's narrative or visual elements. Designers can use one or two focal points in these views to capture the player's attention and make sure they pop out properly. There's so much more to composition than just this, but for the sake of keeping things relevant to this thesis, this will have to suffice. [7] [15] [18] [19]

## 3.6 Pacing

**Pacing** is referred to as the **ratio of intensity to rest** over the course of a segment in the game. What this means is that if the game never changes its level of intensity, it essentially has no intensity at all. It is all about giving the player exciting things at just the right moments, while letting them rest occasionally. But how does one know when to let the player rest and when to give them the time of their life?

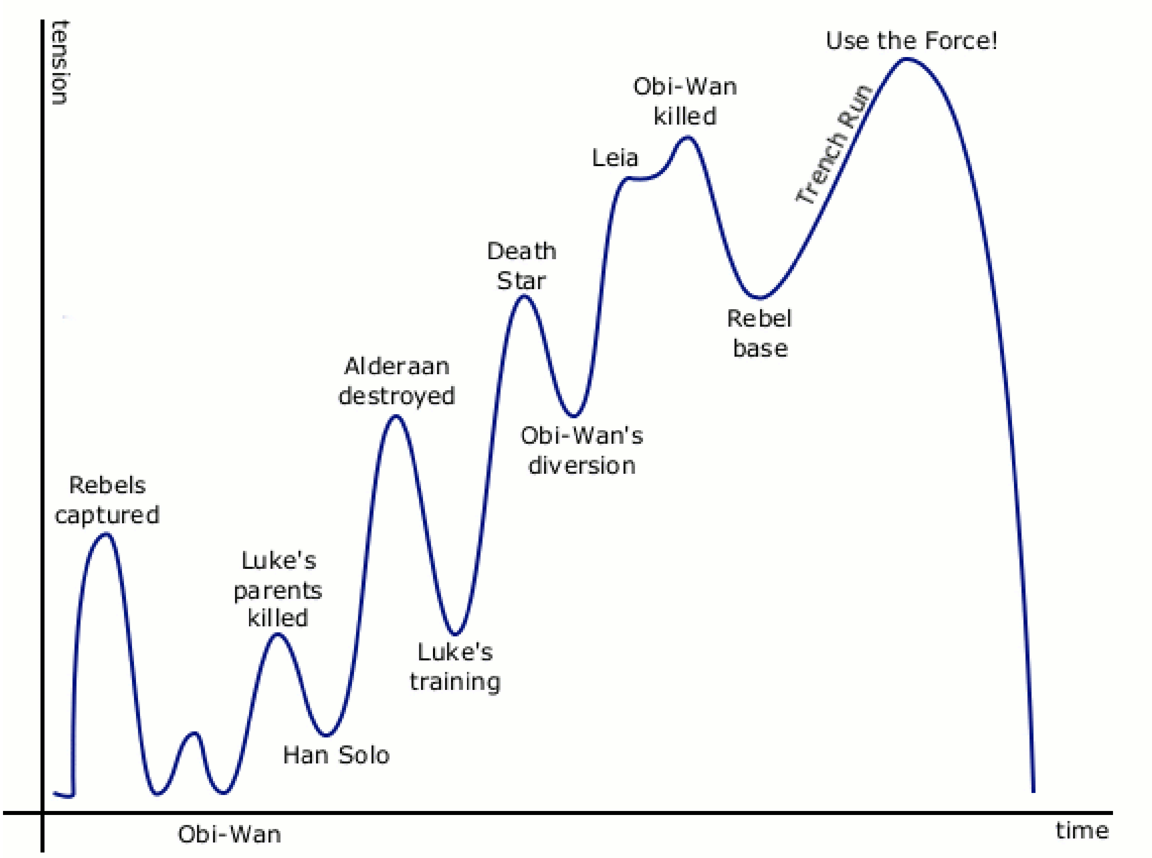


Figure 32. A graph of the pacing in Star Wars: A New Hope.

There are a lot of good examples about successful pacing in media, but the pacing of the story in Star Wars: A New Hope seems to be the most commonly referred one. The movie starts out with an intense action scene, grabbing the viewer’s interest within the first few minutes. After that the movie gives the viewer a rest, a calmer scene where some new characters are introduced. From there on the movie gradually ramps up the intensity while shifting between intense and calm scenes, eventually climaxing in an epic space battle and then giving the viewer their final rest and closure for all the action. Similar intensity curves are used in all medias; open up with a bang, settle down from there, slowly build up for the climax while keeping the viewer interested with small doses of intensity and then provide closure for the experience as a whole. [20] [21]

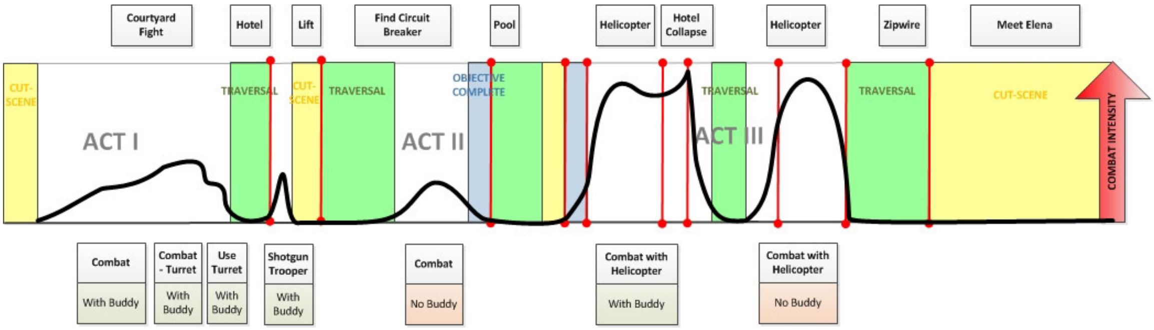


Figure 33. A graph of the pacing in Uncharted 2: Among Thieves.

Pete Ellis has used the pacing of a level in Uncharted 2 as an example of how to apply this into games. [9] The level starts from a stealth setup, the player is close to a lot of enemies, but they have not noticed the player. Once the player engages the enemies in combat, the intensity slowly ramps up and climaxes once a turret vehicle drives into the fray and the player’s objective is to neutralize it. Once they have done so, the last wave of enemies shows up, but the player can dispose of them with the turret vehicle rather easily, slowing down the intensity smoothly to a rest period.

Figure 34. A screenshot of Uncharted 2: Among Thieves.

After the intense combat scenario at the start, the player mostly does some climbing and fights occasional enemies as they make their way through a tall hotel building, providing the player with different, less intense gameplay. After the calm period, the player is rewarded with some story exposition in the form of a cutscene, bringing the intensity to a complete halt. But soon after that the final part of the level starts, which is the most intense one.

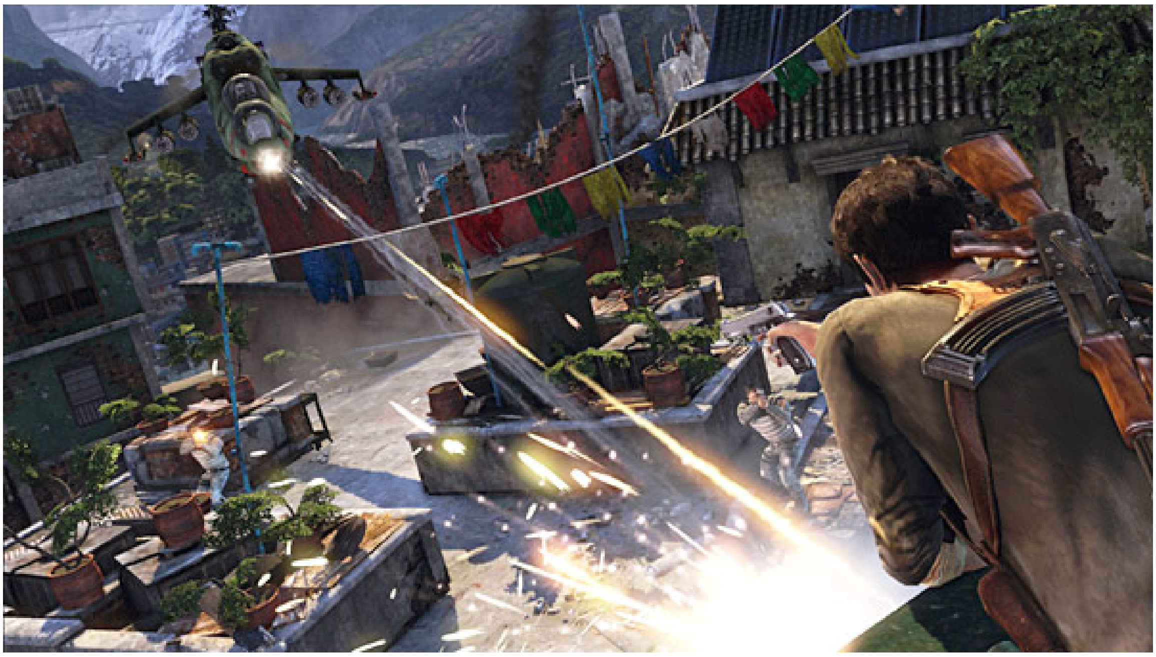


Figure 35. A screenshot of Uncharted 2: Among Thieves.

The player is attacked by a helicopter, which also keeps destroying the environment around them as they try to find cover, creating a whole another level of urgency to the situation, even though it does not make the game any more difficult gameplay-wise. However, this last part is so intense that it has to have a short rest period in the middle before the player returns to fight with the helicopter. Once the player confronts the helicopter and destroys it with grenade launchers, the intensity dies down for good and the player is rewarded with an even longer cutscene, providing a longer period of complete rest after the intense finale of the level.

Notice how the level does not follow the pacing of Star Wars directly hand-in-hand, but still incorporates all of the basic principles. If all media were paced exactly the same way, patterns would arise and that would result in a number of problems.

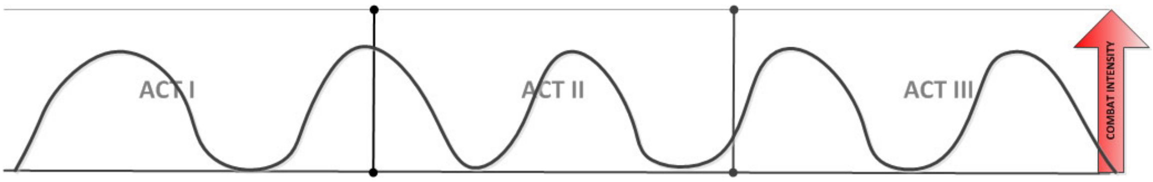


Figure 36. An example of repetitive pacing.

Something designers should avoid when creating pacing for a game is clear patterns. If the player can always predict what's going to happen next, the experience might become boring. Good level design should always surprise the player and introduce something new and exciting all the time. Introduce a new game mechanic or a new way of interacting with familiar game mechanics. Break the pacing a bit to avoid creating clear patterns. And remember to gradually ramp up the difficulty and complexity of the game or level as well. As the player learns new things about the game and slowly masters the game's mechanics, they're going to want more complex challenges that utilize those mechanics. Designers shouldn't let their games become repetitive or boring. [9]

One thing I should note is that pacing a more open level is really difficult. I tried searching information about it online, but I could hardly find any, since apparently it is a really big challenge even for veterans. It makes sense, since one can predict player behavior much better when pacing a linear experience. But when the designer lets the players roam free in a more open environment, it is much more difficult to predict when they would need rest and when to ramp up the intensity. [22]

## 3.7 Conclusion

In order to design an engaging level for a game, there's a lot of things designers have to consider from the get-go. The designer has to know the constraints and the role of the level, which will help them determine the level's contents and the rest of the level's design. There are multiple ways to go when beginning the actual level design, none of which is universally the correct one. Designers just have to pick the one that works for them through trial and error.

Once designing the level, there are a ton of things to take into account every step of the way. Designers should make sure the navigation is as intuitive as possible by using clever player guidance tricks. It is advisable to visual language in the game environment to communicate crucial information to the player without actually speaking a word. Good level design utilizes contrasts and focal points to compose clear views of the level for the player, so they can learn the level's layout as easily as possible. Designers should pace the level's contents properly in order to make sure the player is never too overwhelmed or bored with their experience. And most importantly, playtest the level intensively. No one can learn whether or not their level is netting the desired outcomes if they never watch actual players play their level.

With these tools, I dove into the world of level design to create the best possible environments for Biting Mascot’s 3D platformer; Captain’s Tail.

# 4 Captain’s Tail level design process

My assignment was to design the first level for Biting Mascot’s 3D platformer Captain’s Tail. Before I started designing the level, I had to determine my high-level constraints. For my **deadline**, I was reserved three weeks to finish the whole level, including planning, white-boxing and iteration. Our **franchise constraints** derived directly from our game’s design pillars, which we had decided upon during pre-production;

1. **Joy of movement and agility**, meaning we want to make controlling the character as fun as possible and we don’t want to break the player’s momentum

2. **Pirate-themed adventure**, meaning we want to make the piracy theme apparent everywhere and we want the experience to feel like an adventure filled with surprises and things to discover

3. **Bright mood and humor**, meaning we want the game to have a light-hearted mood and we don’t want to invoke feelings of stress, anxiety, disgust or frustration

4. **Fun for all ages**, meaning we want the game to appeal to both casual and highly skilled players, we have to keep both kinds of players in the flow during gameplay

5. **Meaningful collectibles**, meaning that the player should feel genuinely rewarded when finding collectibles

Our game has two specific **target audiences**. Young adults from ages 20 to 34, who have played similar games before and already find them appealing and also children from ages 3 to 14, who may not have played 3D platformers before, but could find these kinds of games appealing. Both audiences have players from all skill levels and we must make sure that they all find the game engaging and can stay in the flow. We had no **client requirements**, because we are making this game all by ourselves and we don’t have a publisher at the time I am writing this thesis. As for the **engine constraints**, I used Unity and its ProBuilder tool to create these levels, which didn’t exactly limit me in any significant ways.

Once I had my high-level constraints determined for all levels, I dived into the actual level design process of the first level.

## 4.1 Level 1 planning and sketching

I started designing the level by thinking about **the level's purpose**. It's the first level in the game and since we have no separate tutorial level, this level will be the player's first hands-on experience with the gameplay. Once the player sets foot in this level, they would have no idea how the game works or what the core mechanics are. I made it my first priority that the first level should be about teaching the game's core mechanics to the player while simultaneously keeping more skilled players in the flow by providing optional challenges. It would act as the basic tutorial level, without feeling like a tutorial level.

I realized that the biggest challenge would be that players learn new things at a different pace and I would have to take that into consideration as well. Based on my observations in previous projects, some players try out every mechanic on their own before they even start progressing in the game, while some players only do what the game demands of them. I also thought about how we're going to teach mechanics to the players if the levels would be more open. The more open an environment is, the less accurately we can predict where players will go.

I started thinking about the concept of **funneling**. We can make the level's areas more open and still predict where the players will go if we funnel the open areas together with more linear segments. This means that the player would have to cross a certain linear path before they can move on to the next open area. This way we can predict that all players will end up on that linear path at some point if they want to progress in the level. We can place the tutorials on these more linear paths, ensuring that all players will stumble upon them during the level and learn more about the game.

**The theme** of the first level came from an idea we had for the first level's name; “Bay Six”. It would act as a pun for teaching the player the “basics” and so the level's environment would have to look like a bay area. In addition to that, we decided that the first levels of the game would share a Caribbean island theme, meaning large beaches, lush jungles and the occasional cave on the water's edge. We decided that the first level would be mainly uninhabited. This means that the level's geometry would have to include mostly natural, asymmetrical shapes and nothing resembling of any larger civilization.

**The goal** of the first level was not decided until later on, which in retrospect caused me some difficulties and some later reworks for the level. At the time I was simply concepting the level, I had little to go on as the level's goal. We had been talking about that the first level's goal would be a pirate ship that the main character would redeem for themselves at the end of the level, meaning that the level would have to end at a shore, where a ship could naturally float around. I kept this option open as I continued concepting the level. I also wanted to make the level loop in on itself somewhat, making it possible to easily travel between different areas of the level without having to run long distances back and forth. This also affected the goal's placement within the level.

I thought about **the desired player experience** for the first level a lot. I wanted the players to learn the game's basic mechanics quickly, but I didn't want to overwhelm them with a constant flow of new information. I also wanted the players to be able to intuitively keep up their momentum and choose their paths based on their skill level. I also did not want the open areas to feel too overwhelming, since it should always be clear where to go in order to proceed in the level. With this in mind, I thought about a structure where I teach the players some mechanics in a short, linear segment and then offer a more open area where I can test the player's understanding of these mechanics. After the more open area, I would funnel the level back into a more linear segment, teach some new things and then test the players again in a more open area and so on. In theory, this would allow for better pacing of the tutorials in the level and also offer chances for players of different skill levels to choose their own paths in the frequent open areas, keeping players in the flow.

As for **the scale of the level**, we wanted to aim for slightly longer than the length of the level we had in the Unreal Engine demo. We decided that we would aim for roughly 10-minute long levels, give or take a few minutes. However, estimating a level's optimal length during planning and sketching can be difficult, especially early on in development, as I learned later on.

This level would be the first in the game's overall **level progression**, so I couldn't assume that the player knows anything about the game. I would have to teach every mechanic to the player before demanding that the player understands them. The challenges within the level would also have to be mild while also rewarding the player a lot for passing those minor challenges, to keep the player's motivation strong.

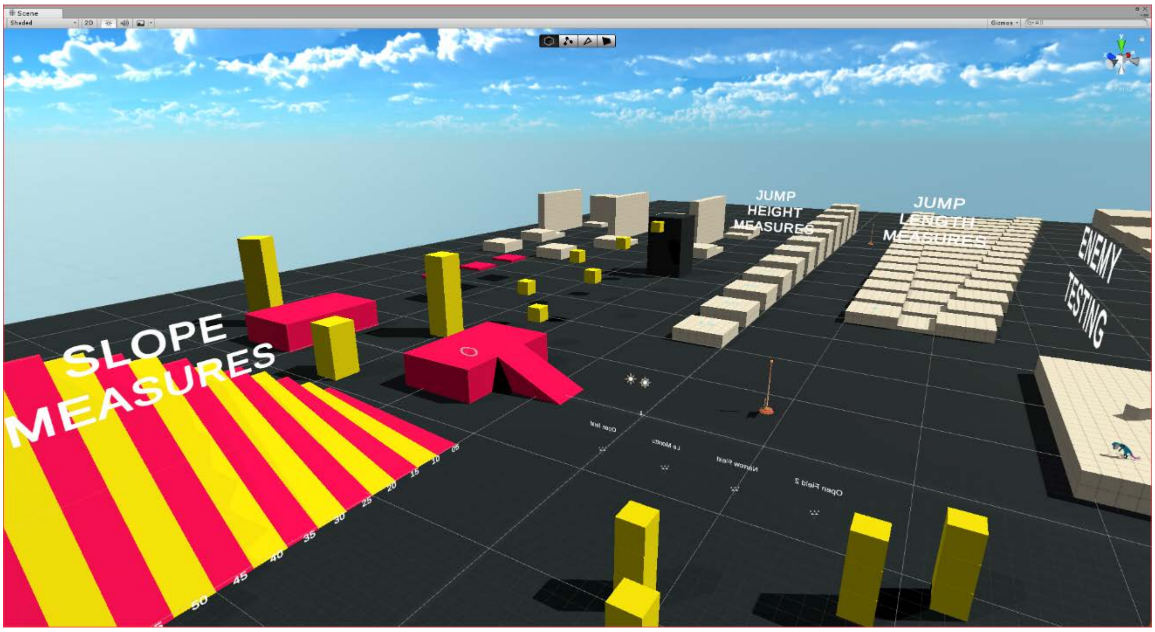


Figure 37. Our test room, where I tested and created different measures for the game.

Before I started sketching or white-boxing anything, I created measures of our game's **metric constraints**. I made notes of our main character's jump lengths and heights, both optimal and maximum measures. Optimal being the most comfortable jump height or length for jumping, double jumping or sprint jumping; not too long or too short. These measures help me create comfortable gaps in the environment without having to measure the best possible measures over and over again. Maximum measures refer to the absolute maximum distances or heights the player could reach with each jump. These measures helped me estimate how high or far to place objects that I don't want the player to reach from certain places.

We didn't spend too much time thinking about **the history of the environment** or other narrative elements, because our game is very light on the narrative side. We did decide that the bay area featured in the first level would be a pirate smuggling cove, which explains why there are enemy pirates and treasure to be found in the level.

I didn't exactly have a **visual trademark** in mind for the level at the planning stage, other than that it was a level based around a bay area. It was not until iteration when we got a much stronger identity for the level in the form of a better visual trademark.

As for the level's **set-pieces**, I based them around the level's contents. There were six key gameplay beats the level would include;

1. Basic mechanic tutorial. This area would teach about movement, jumping, double jumping and collectibles.

2. Test understanding of basic mechanics, some light challenges and optional tougher challenges utilizing the mechanics taught before this.

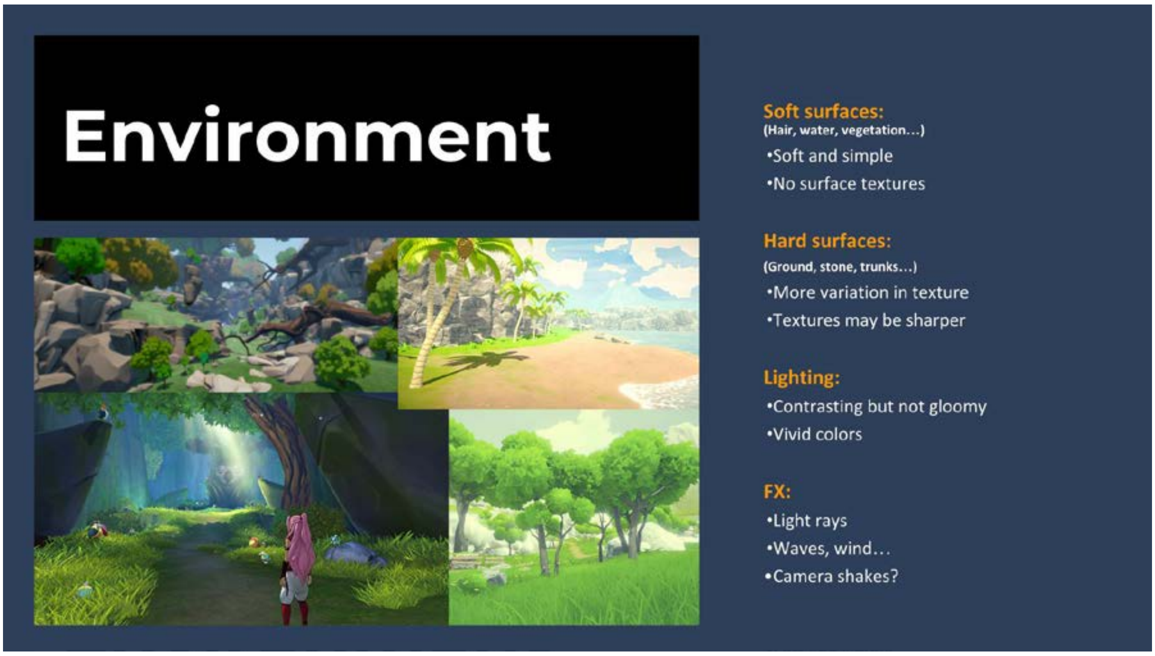
3. Combat and sprint tutorial. This area would teach the player how to attack, how to defeat enemies and how the game's sprinting works.

4. Test understanding of basic mechanics, some light challenges and optional tougher challenges utilizing the mechanics taught before this.

5. An open field with plenty of optional challenges for those who have mastered all of the mechanics taught in this level.

6. The finale, a linear segment with some light challenges based on the mechanics taught in this level.

I decided to form pairs of the level's areas and give each pair a unique look and feel within the level. Areas 1 and 2 would take place in the island's bay area, with a huge cliff on one side and the open sea on the other. Areas 3 and 4 would take place in a jungle as the player reaches the other end of the bay. This jungle would take the player to areas 5 and 6, which take place on the huge cliff next to the bay area. By making the level's end take place on the cliff next to the level's starting area, the level would loop in on itself, making travelling between different points of the level much easier without having to backtrack long distances.



Environment

Soft surfaces:
(Hair, water, vegetation…)
- Soft and simple
- No surface textures

Hard surfaces:
(Ground, stone, trunks…)
- More variation in texture
- Textures may be sharper

Lighting:
- Contrasting but not gloomy
- Vivid colors

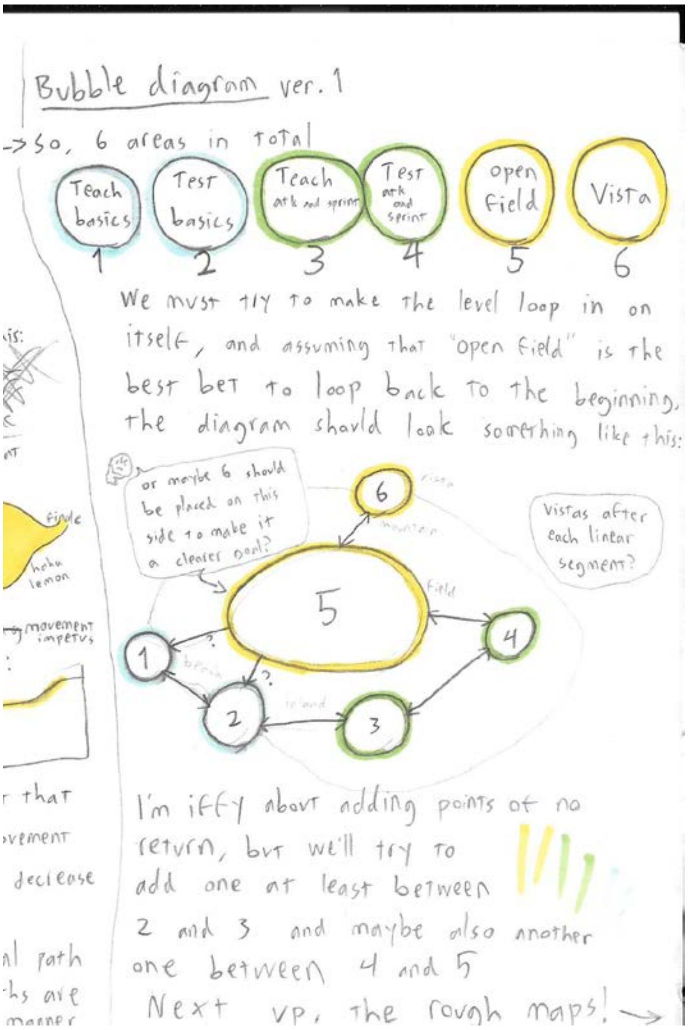
FX:
- Light rays
- Waves, wind…
- Camera shakes?

Figure 38. A page of Captain's Tail's art bible, made by Biting Mascot's lead artist.

Our artists were in charge of **the looks** of our levels, so I merely tried to follow what I found in the game's art bible. In retrospect, I should have used Pinterest or Flickr more actively, since I encountered a lot of mental blocks later on and I could have drawn inspiration out of Caribbean scenery images.

I decided I would utilize the level's geometry as my main tool for **player guidance**. This would be the most cost-efficient and versatile solution, since almost all other methods would require additional artwork or programming from the rest of the team.  I would also use our game's collectibles and other objects to support the guidance, but I wanted the geometry of the level to at least create the groundwork for other supporting methods.

As I started planning I tried to utilize both Pete Ellis' **visual timeline** and Mike Stout's **bubble diagram**, just to see which would work better for designing these levels. I found it difficult to plan or predict the level's pacing curve in the visual timeline, due to the open areas of the level. Instead, I found Stout's bubble diagrams much more viable for this type of work. As I mentioned before, I divided the level into six areas, so I used those six areas to construct the bubble diagram. Connecting the bubbles before even designing their contents more specifically helped me plan the level's structural goals much easier. These goals included that the level should revolve around a bay and that it should loop in on itself.



Bubble diagram ver. 1

-> So, 6 areas in total

We must try to make the level loop in on itself, and assuming that "open field" is the best bet to loop back to the beginning, the diagram should look something like this:

I'm iffy about adding points of no return, but we'll try to add one at least between 2 and 3 and maybe also another one between 4 and 5

Next up, the rough maps! ->

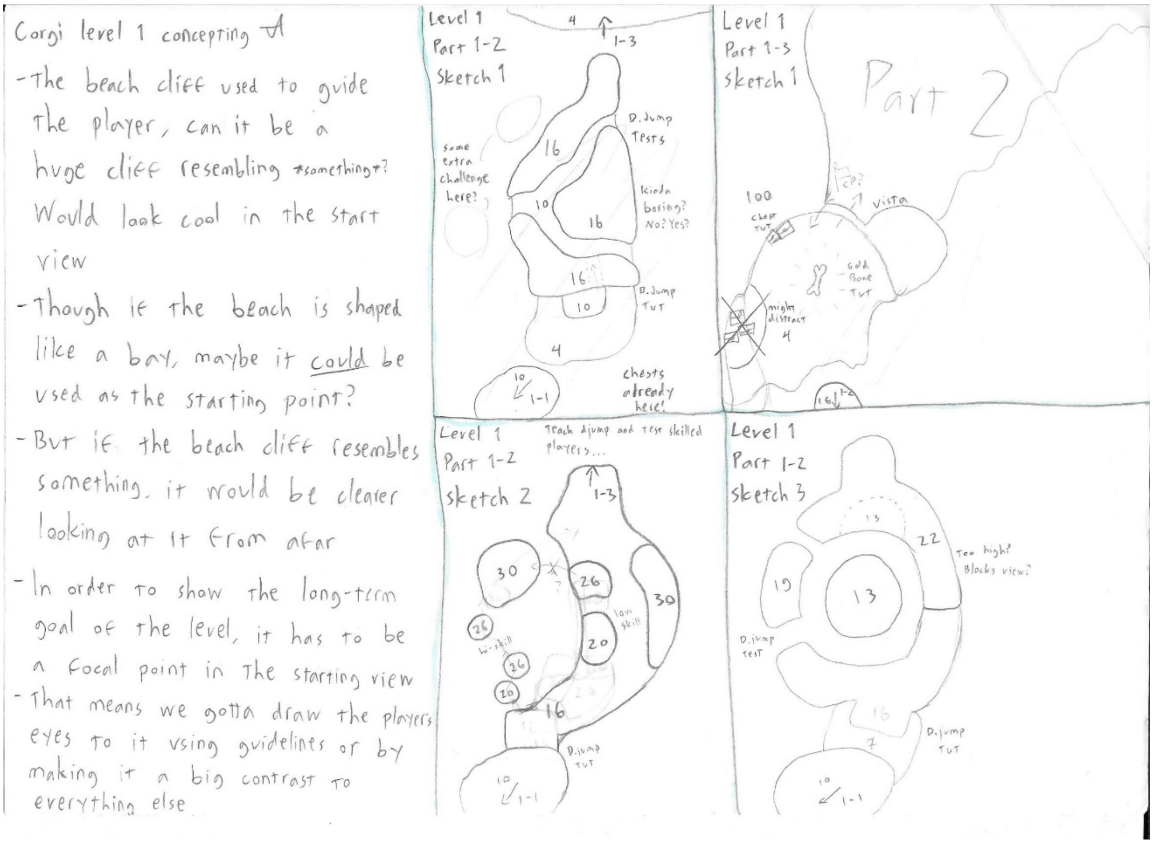
Figure 39. Initial bubble diagram and notes for the level.

Once I had my bubble diagram ready, I started sketching the rough maps of each area.

### 4.1.1 Area 1

However, as soon as I started making **rough maps** of the first area, I realized that if the player starts at the beach of the bay area, the goal of the level is not visible at the start of the level. I thought that his could be a problem, because the player would not have a clear motivation during the level, they would not know what is it that they’re looking for. So instead of placing area 1 at the beach of the bay, I decided to move area 1 to some smaller islands, away from the main island. The player would then get a much clearer view of the main island and would have to travel there using those smaller islands. This would also work well in terms of the first area’s purpose, which is to provide a linear section, where we teach the player the basic mechanics.

Regarding the starting view of the level, I thought about testing a similar start view camera angle as Ratchet & Clank games used. By providing the player with a specific camera angle that frames the main island and the goal of the level, they would already have a much clearer picture about what to expect from the level and where they should go. This would also work well when starting the level from the smaller islands next to the main island.



Corgi level 1 concepting

- The beach cliff used to guide the player, can it be a huge cliff resembling *something*? Would look cool in the start view
- Though if the beach is shaped like a bay, maybe it could be used as the starting point?
- But if the beach cliff resembles something, it would be clearer looking at it from afar
- In order to show the long-term goal of the level, it has to be a focal point in the starting view
- That means we gotta draw the players eyes to it using guidelines or by making it a big contrast to everything else

Figure 40. Level 1, area 1, initial rough maps.

I did multiple sketches of each area, but the basic concept stayed the same. The level would start at an open area, where the player can freely try out the controls if they wish to do so. The open area would quickly funnel into a small ledge, where the player must use jump to proceed. After the initial jump lesson, the player would be offered some optional ledges and platforms with rewards to give the player contexts of different situations where jumping may be used and reward them for learning their lesson. Soon after the initial jumps the player would have to use double jump to proceed and likewise offer optional double-jump related platforming after learning the mechanic.

Once the player has been taught about jumping and movement, the area would connect with the main island. I wanted to give the players a grand new view of the bay area once they reach it, so I decided to make the players walk through a tunnel beach cliff before reaching the main bay area, so that I could compose a specific view for the players once they pass the tunnel.

### 4.1.2 Area 2

I initially wanted to focus area 2, the actual bay area, on testing the player's understanding of the basic movement and jumping mechanics, nothing else. However, as I tried to come up with ideas how I would test the player without having to teach them anything new, I found myself at a loss multiple times. Instead of trying too hard to make do with so little, I decided that I could use some additional mechanics to make the actual bay area more interesting. I decided that I would use area 2 to teach the player about attacking and enemies as well while also testing the player's understanding of basic platforming. By intertwining these two goals, the area could offer much more different kinds of content, keeping the level's gameplay and pacing interesting.

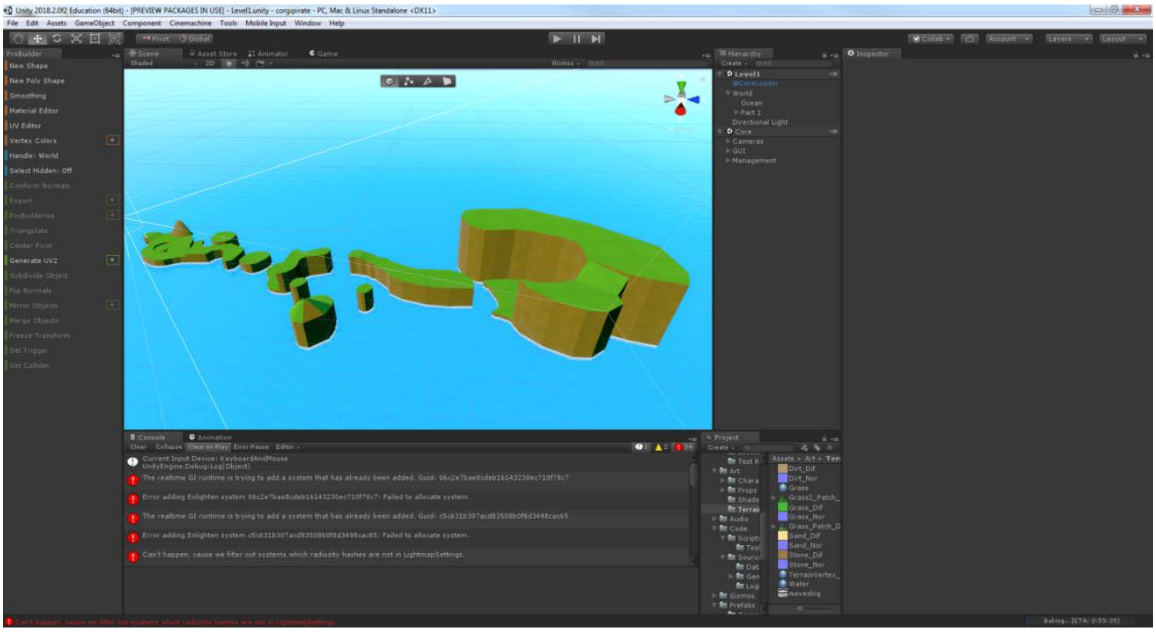
I also included some optional side paths in area 2 which utilize our more advanced mechanics, like the wall-run feature. This is because the team wanted to make sure that first-time players would not necessarily be able to discover everything within the levels on their first playthrough. We also did not want to hard-lock any content, so instead we made optional challenges based on mechanics that the player will learn in later levels and thus, they will have a reason to return to previous levels to utilize the new mechanics they have learned.

While sketching the first two areas, I experimented with two different approaches to making rough maps. At first, I tried sketching the areas based on the roles they need to fulfill, letting the level's structure unfold based on those needs. Then, I tried sketching the area's structure first and then fill it with things the area has to contain, such as tutorials, optional challenges and so on. I found the latter approach much more difficult to use and it took me a lot of time. Coming up with a space that I'm trying to fill with interesting content afterwards resulted in a lot of empty, poorly-used space. Therefore, I decided I would use the former approach when making these sketches, unless a more efficient approach presents itself.

As I made multiple sketches of each area, I realized how much time I am sinking into them and how little they can offer me in the long run. I decided to settle on my sketches sooner than before and just playtest the level as soon as possible to get the best outcomes. One can only get so far with plans on paper. This is why as soon as I finished concepting area 2, I started white-boxing area 1.

## 4.2 Level 1 white-boxing, playtesting and iteration

This was my first time using Unity's ProBuilder tool to white-box our levels. Before starting this project, I did make myself familiar with it and made some very short test levels, just to see how it works. I managed to white-box area 1 fairly quickly and moved on to area 2.



```
Current Input Device: KeyboardAndMouse
UnityEngine.Debug:Log(Object)
The realtime GI runtime is trying to add a system that has already been added. Guid: 06c2e7bae8cdeb16143230ec710f78c7
Error adding Enlighten system 06c2e7bae8cdeb16143230ec710f78c7: Failed to allocate system.
The realtime GI runtime is trying to add a system that has already been added. Guid: c5c631b307acd83508b0f8d3498cac65
Error adding Enlighten system c5c631b307acd83508b0f8d3498cac65: Failed to allocate system.
Can't happen, cause we filter out systems which radiosity hashes are not in LightmapSettings.
```

Figure 41. Level 1, area 1, first white-box draft.

I continued white-boxing area 2 directly from my rough maps, adjusting some heights and distances as I was able to perceive them better in a 3D space. Once I finished white-boxing area 2 about halfway through, we began playtesting the level. During playtesting we got some valuable feedback about the level and I began fixing and reworking segments of areas 1 and 2 right away.

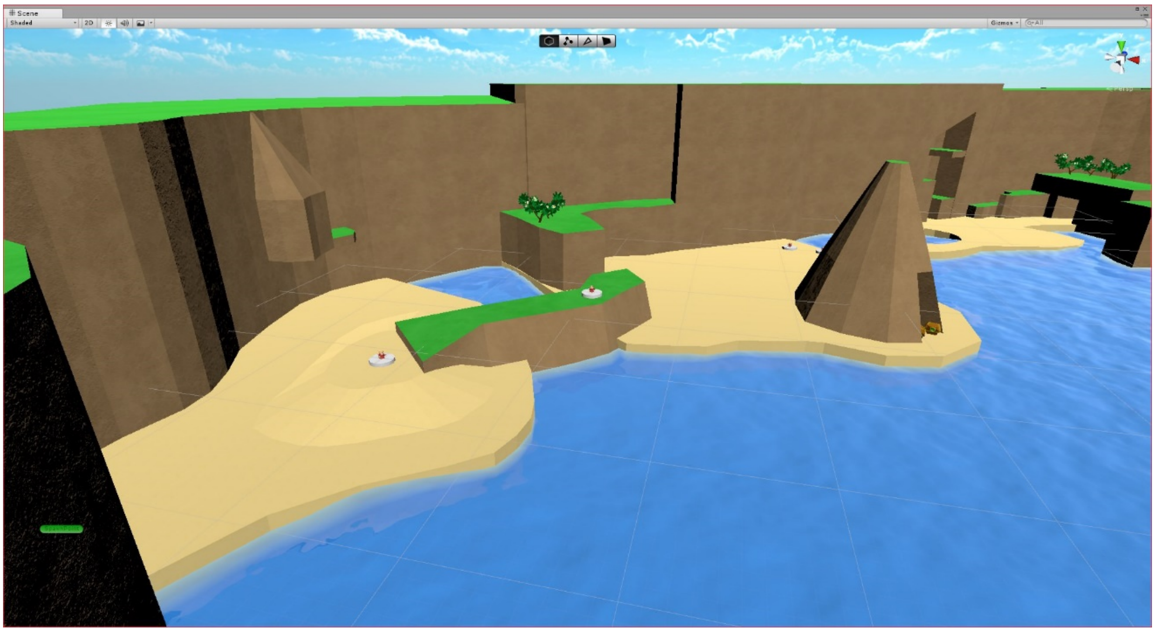


Figure 42. Level 1, area 2-1, first white-box draft.

The first thing that came up during playtesting was the difficulty curve of the level. In area 2 I introduced enemies in more complex situations too quickly for new players and many got overwhelmed. I decided to chop up the complex situations into smaller pieces and introduce them one by one in more safe environments, so players will have more time to get familiar with the game’s mechanics and concepts. Another thing that came up was that the enemy encounters already felt repetitive over the short course of the level. To fix this, I decided to focus on the quality of the enemy encounters, rather than their quantity. I also thought about dividing the level’s areas into clearly hostile and non-hostile areas. What this means is that some areas of the level would feature enemies while other areas would be completely enemy-free, focusing more on platforming and exploration. This would also require that whenever the player enters a hostile area within a level, the area’s enemies would have to be introduced clearly right away to signal to the player that this is a hostile area.

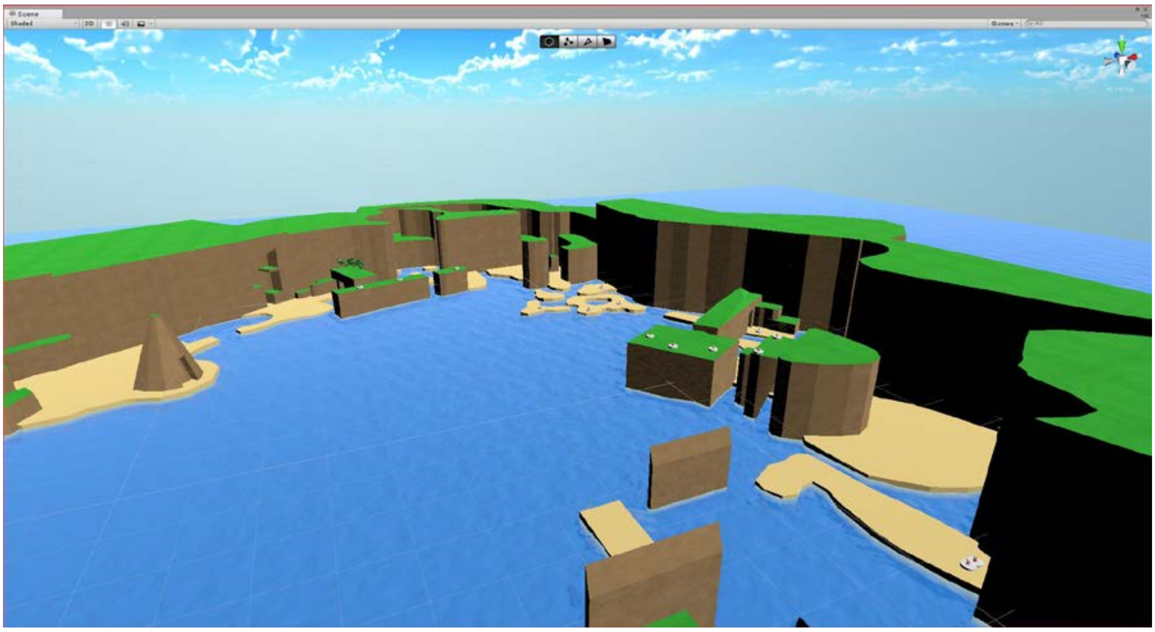


Figure 43. Level 1, area 2, first white-box draft.

Once I finished making iterations for the existing areas, I finished white-boxing area 2. As I took a look back at the finished area 2 I realized that all the higher structures I had created as side-paths in the beach really obstruct the player's vision and make it harder for the player to visualize where they're supposed to go to proceed. I did not take this into account when making rough maps, because it's much harder to perceive 3D space from a top-down sketch. At this point I did not have enough time to try to think of a re-work for this issue, so I just left them as they were and decided to focus the player guidance in area 2 on landmark props and collectibles. Later on I removed some of the taller structures and designed entirely different content in their place.

### 4.2.1 Area 3

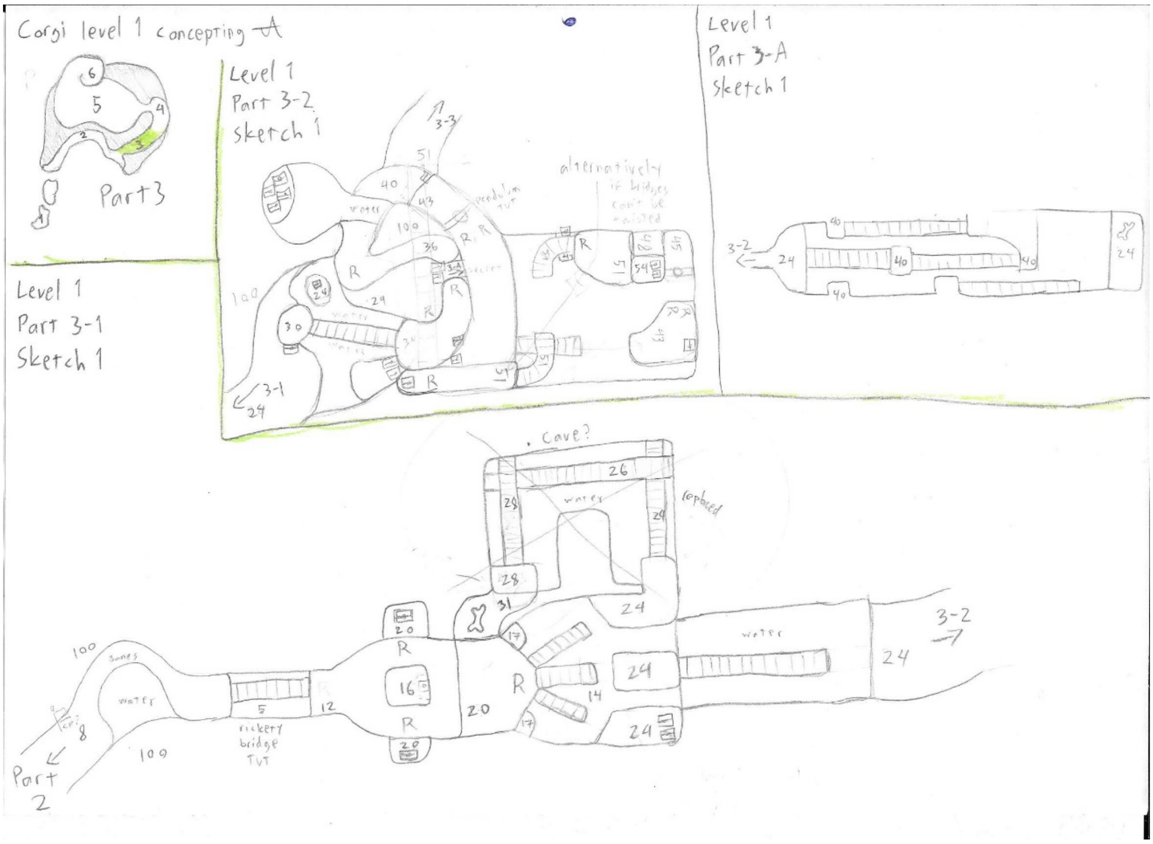


Figure 44. Level 1, area 3, initial rough maps.

Because I moved the combat tutorials to area 2, area 3 would now only act as the sprint tutorial, which is fine, because there are a lot of ways we can utilize sprint and the previous mechanics together. I ended area 2 by funneling it in-land towards the more jungle-themed area 3. I decided that we will teach sprinting to the player by utilizing rickety bridges, which start crumbling away as the player steps on them. The only way to travel across these bridges is to sprint across them fast enough so that the player won't fall with the bridge planks.

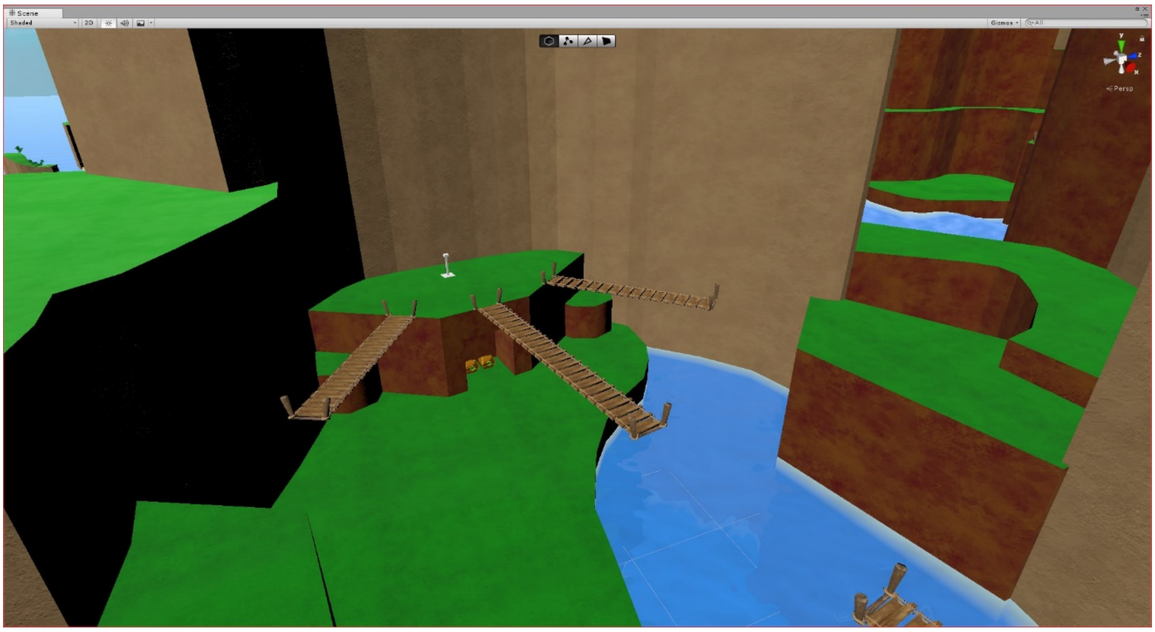


Figure 45. Level 1, area 3-1, first white-box draft.

At this point I realized that since there are plenty of paths intertwining here and there, it may be much tougher for the player to distinguish the critical path of the level. I started thinking about other ways to guide the player on the critical path than just the level geometry. I talked about it with one of our artists and we decided that **a lantern prop** could act as a universally usable focal point. A lantern would offer two kinds of contrasts to draw the player's eye in most environments; the lantern's bright orange color would contrast with the rest of the environments colors and its light source would create a lighting contrast. In addition to that, if the lanterns are only placed alongside the critical path, the player would learn the correlation between the lanterns and the critical path; follow the lanterns to reach the end of the level.

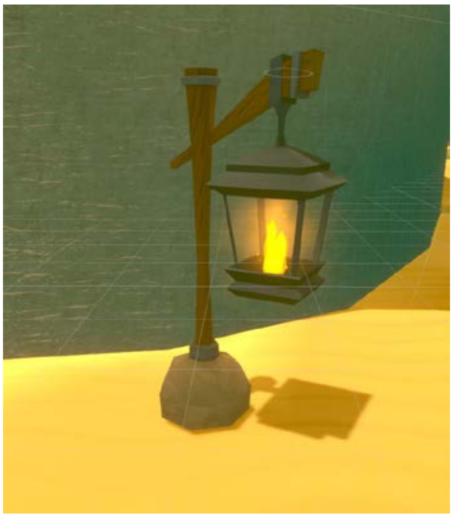
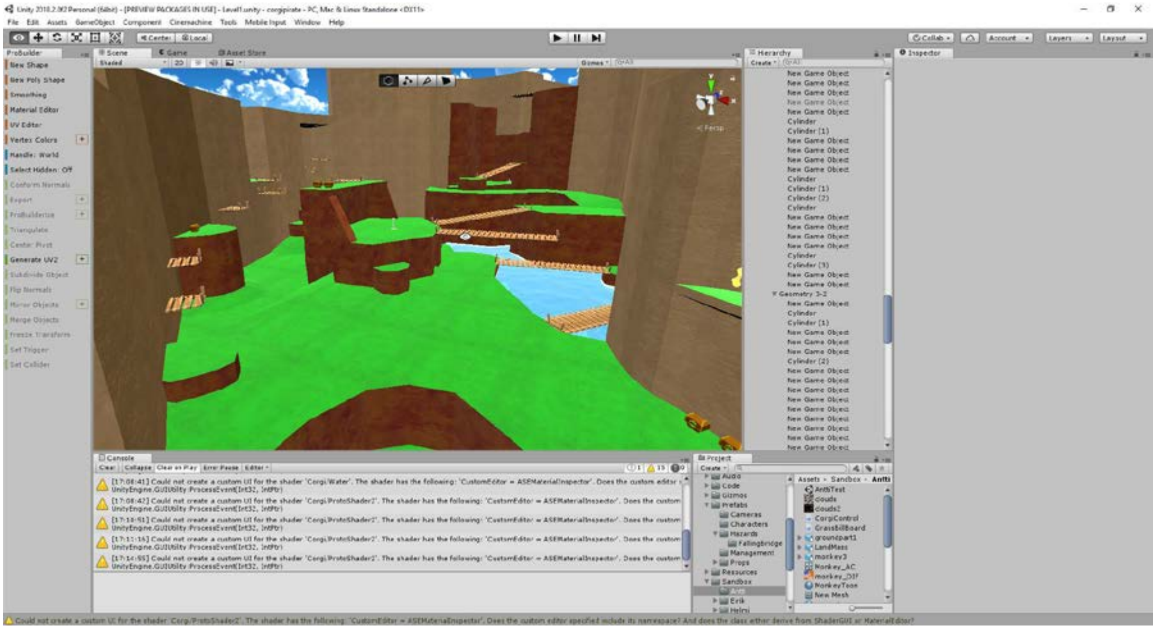


Figure 46. The lantern prop used in the level to guide the player.

As I sketched area 3, I slowly came to realize that the level is quite large as it is and creating three more areas of similar size would probably result in a level that's too long. I decided to combine areas 3 and 4 into one, so that area 3 would teach the player about sprinting and also test their understanding of it through challenges, hoping that it would not overwhelm the players. In addition to that, I decided that adding another area after the cliffside would definitely stretch the level's overall length. So area 4 will now refer to the cliff part of the level, which is also the finale.

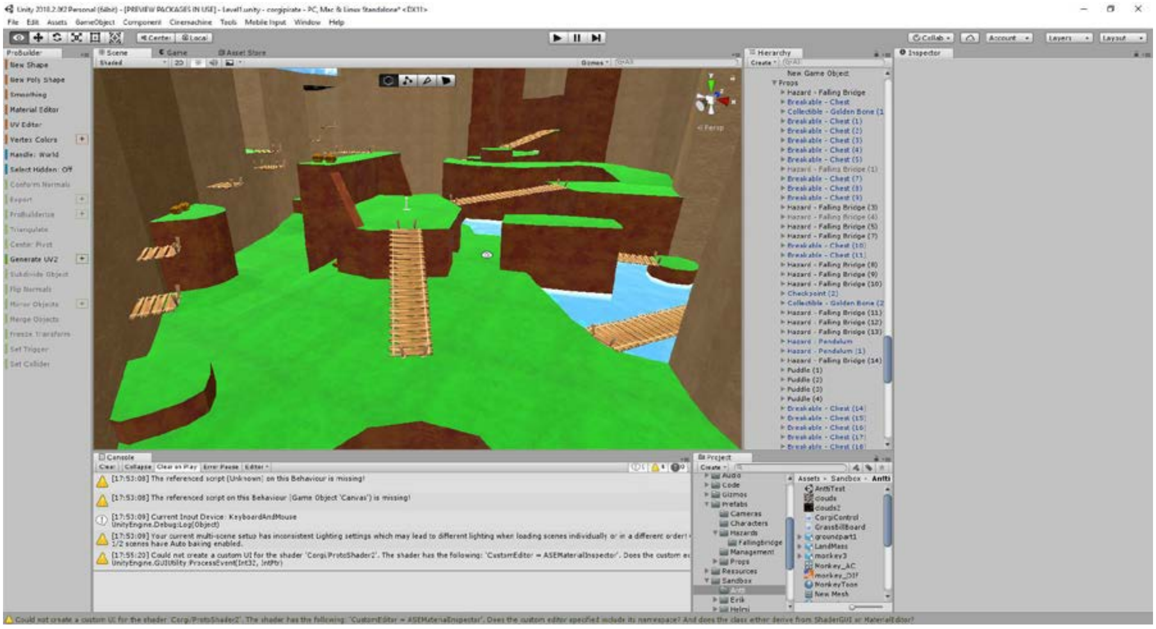


Unity 2018.2.0f2 Personal (64bit) - [PREVIEW PACKAGES IN USE] - Level1.unity - corgipirate - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Cinemachine Tools Mobile Input Window Help
Center Local
ProBuilder
Scene Game Asset Store
Shaded 2D
Gizmos
Hierarchy
Inspector
Collab Account Layers Layout
Console
Clear Collapse Clear on Play Error Pause Editor
Project

```
[17:08:41] Could not create a custom UI for the shader 'Corgi/Water'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom editor
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
[17:08:42] Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
[17:10:51] Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
[17:11:16] Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
[17:14:55] Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom editor specified include its namespace? And does the class either derive from ShaderGUI or MaterialEditor?
```

Figure 47. Level 1, area 3-1, second white-box draft.

As I tried to design more open segments into area 3, I ran into problems with how to design open areas that don’t feel empty. But once I tried to tackle this, the segments ended up getting cluttered and filled with too much content. I consulted Critical Force’s level designer Nicky Pelupessy for help on this matter. He suggested that I should keep the player’s options in mind whenever designing an area. Once the player enters an area, they will try to take in what they see in the initial view. Based on the options they can perceive, they will form a plan on how to proceed, where to go first, what the check out in which order and so on. If the area doesn’t provide much options, it won’t require much thought and the players will just move on fairly quickly. However, if the area provides too many options, the player might get overwhelmed and they might be unable to form a clear plan of progression for themselves. The optimal amount of options in an area depends heavily on the type of game and the type of players the designer is designing for. With this in mind, I started iterating area 3’s open segments. Below is a picture of the end result.


Unity 2018.2.0f2 Personal (64bit) - [PREVIEW PACKAGES IN USE] - Level1.unity - corgipirate - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Cinemachine Tools Mobile Input Window Help
Center Local
Collab Account Layers Layout
ProBuilder
New Shape
New Poly Shape
Smoothing
Material Editor
UV Editor
Vertex Colors
Scene Game Asset Store
Shaded 2D
Gizmos
Hierarchy
Create
Inspector
Project
Assets › Sandbox › Antti
Console
Clear Collapse Clear on Play Error Pause Editor

```
[17:53:08] The referenced script (Unknown) on this Behaviour is missing!
[17:53:08] The referenced script on this Behaviour (Game Object 'Canvas') is missing!
[17:53:09] Current Input Device: KeyboardAndMouse
UnityEngine.Debug:Log(Object)
[17:53:09] Your current multi-scene setup has inconsistent Lighting settings which may lead to different lighting when loading scenes individually or in a different order!
1/2 scenes have Auto baking enabled.
[17:55:20] Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom e
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
Could not create a custom UI for the shader 'Corgi/ProtoShader2'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom editor specified include its namespace? And does the class either derive from ShaderGUI or MaterialEditor?
```

Figure 48. Level 1, area 3-1, third white-box draft.

While playtesting the newer areas, I noticed that limited collectibles effectively keep track of how many secrets or challenges the player has overcome. Limited collectibles refer to the collectibles within a level that do not respawn when collected, even when the level is reloaded. By placing limited collectibles at the end of optional routes and challenges, the players have a visual indication of how many challenges they have beaten and how many there still are to find. Rewarding the player with limitless collectibles at the end of optional challenges cannot pull this off so effectively. In terms of level design, this can help the player feel a stronger sense of progression.

### 4.2.2 Area 4

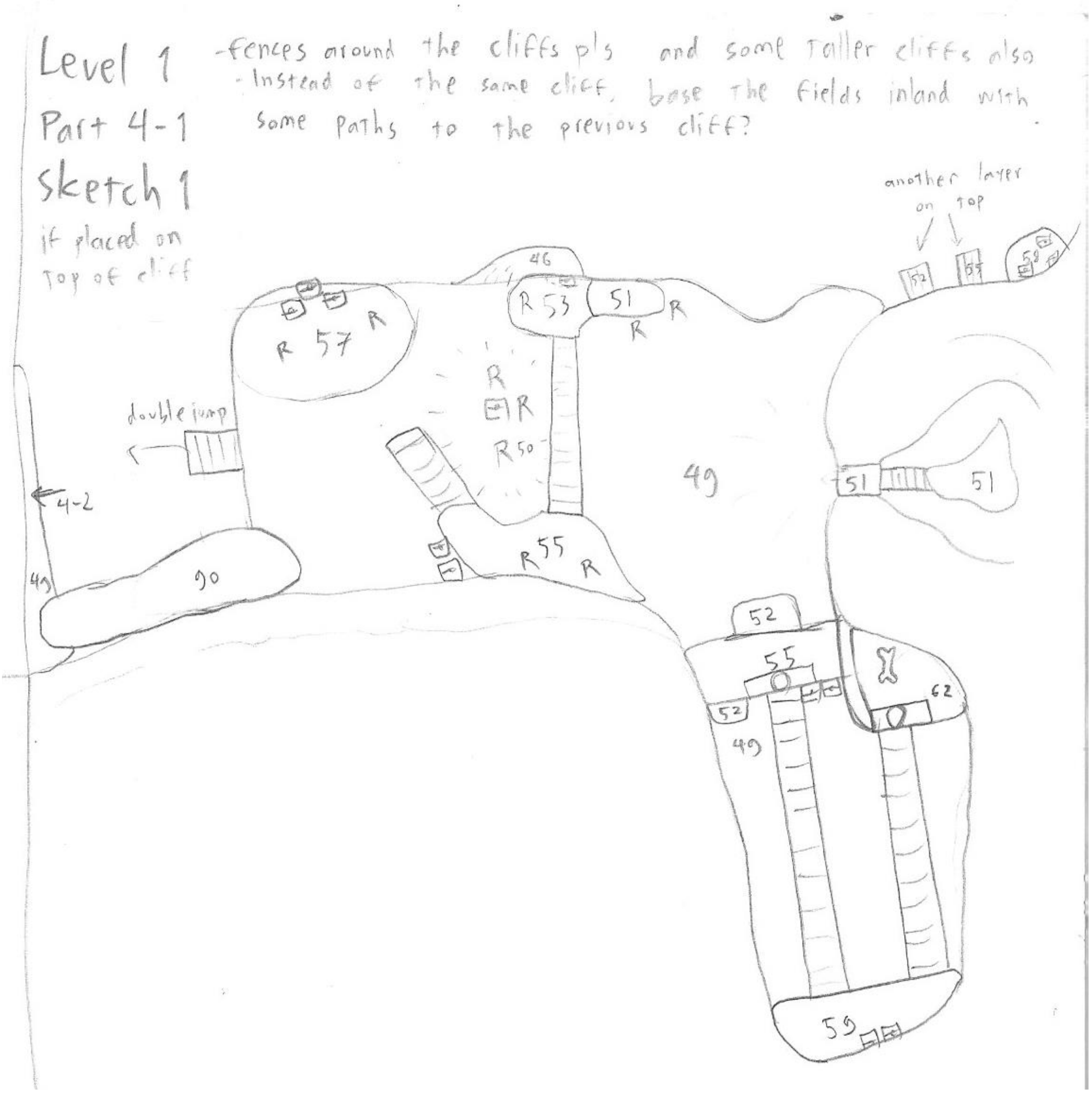

Level 1
Part 4-1
Sketch 1
if placed on
Top of cliff
-fences around the cliffs pl's and some Taller cliffs also
-Instead of the same cliff, base the fields inland with some paths to the previous cliff?
another layer on top
double jump
4-2
49
90
R 57 R
46
R 53
51
R
R
R
R 50
R 55 R
49
51
51
52
55
52
49
62
59
52
55
58

Figure 49. Level 1, area 4, initial rough maps.

Once I finished white-boxing and iterating area 3, I moved on to make rough maps of area 4, the cliffside. The length of the area was more or less decided at this point, because it had to follow the shape that the bay area in area 2 had created. Area 4 was supposed to be a more open area with plenty of optional challenges utilizing all the mechanics taught to the player at this point. With this in mind, I made area 4's critical path fairly linear and simple while creating optional paths all around it. This would allow players to play through area 4 rather quickly if they just want to proceed, keeping the level's overall length in check, but also allow players to go and explore the areas around the critical path.

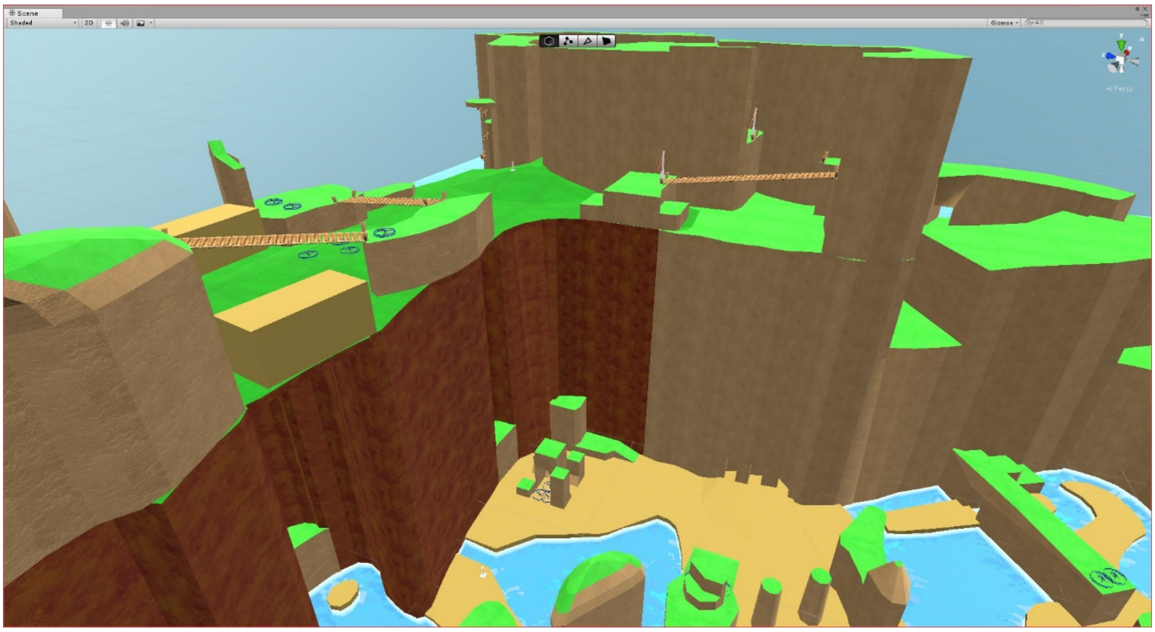
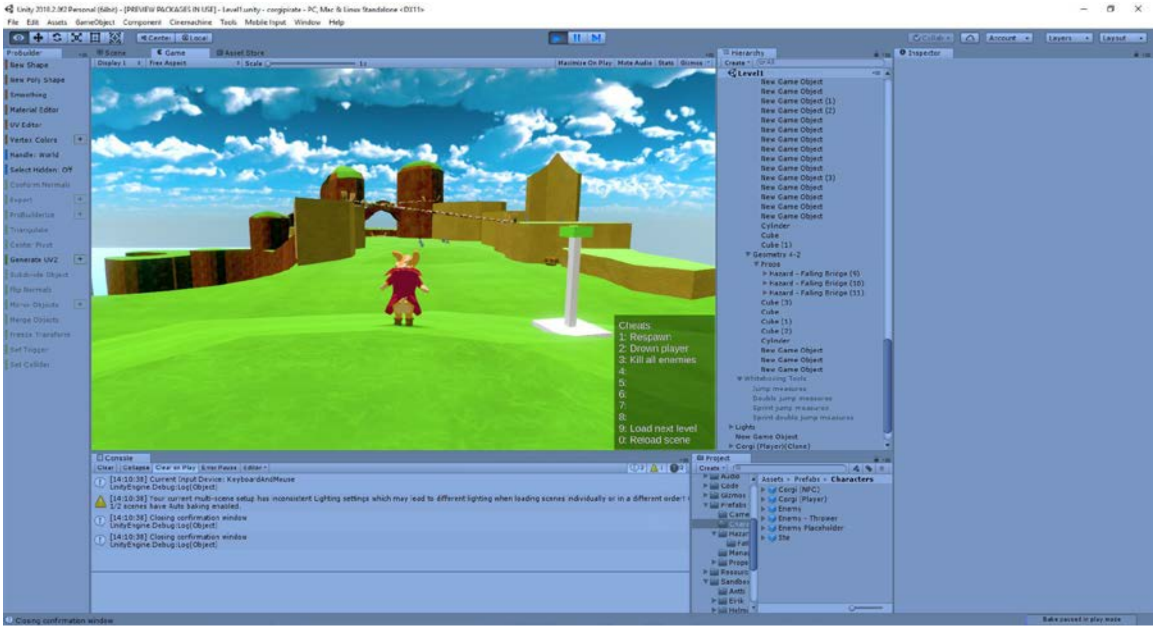


Figure 50. Level 1, area 4-1 shown above, area 2-2 shown below.

I started experimenting new techniques when sketching area 4. Instead of finishing the entire area before white-boxing it, I tried sketching the area one segment at a time and white-boxing it immediately once the sketch is done. This helped me with the sketching of the following segments, because I would already know what kinds of problems came up when white-boxing the first segment and if it would affect the rest of the segments. Sketching and white-boxing hand-in-hand tended to work much better rather than focusing on one or the other completely before moving on.

Another thing I found useful was making the player's next mid-term goal clearly visible whenever they reached their previous mid-term goal. For example, once the player reaches a checkpoint, there should be a clearly visible focal point in the critical path that would draw the player's attention. Once the player reaches the focal point, the view from there would also include another focal point in the critical path to point the player in the right direction constantly. This way the player would intuitively travel from focal point to the next, keeping them on the critical path of the level. These focal points could be either gameplay-related objects, like checkpoints or contrasts in the environment, like geometrical shapes or guidelines and so on.



Unity 2018.2.0f2 Personal (64bit) - [PREVIEW PACKAGES IN USE] - Level1.unity - corgipirate - PC, Mac & Linux Standalone <DX11>

Cheats:
1: Respawn
2: Drown player
3: Kill all enemies
4:
5:
6:
7:
8:
9: Load next level
0: Reload scene

```
[14:10:38] Current Input Device: KeyboardAndMouse
UnityEngine.Debug:Log(Object)
[14:10:38] Your current multi-scene setup has inconsistent Lighting settings which may lead to different lighting when loading scenes individually or in a different order!
1/2 scenes have Auto baking enabled.
[14:10:38] Closing confirmation window
UnityEngine.Debug:Log(Object)
[14:10:38] Closing confirmation window
UnityEngine.Debug:Log(Object)
```

Figure 51. Level 1, area 4-1, first white-box draft.

In the picture above, once the player reaches the checkpoint they're standing next to, there is a small tunnel between two tall pillars, creating a frame that draws the player towards it. The third pillar on the right was later removed to further emphasize the effect of the key focal point in the view. I wish I had realized this technique sooner, because it turned out to work really well in later levels too, if properly done.

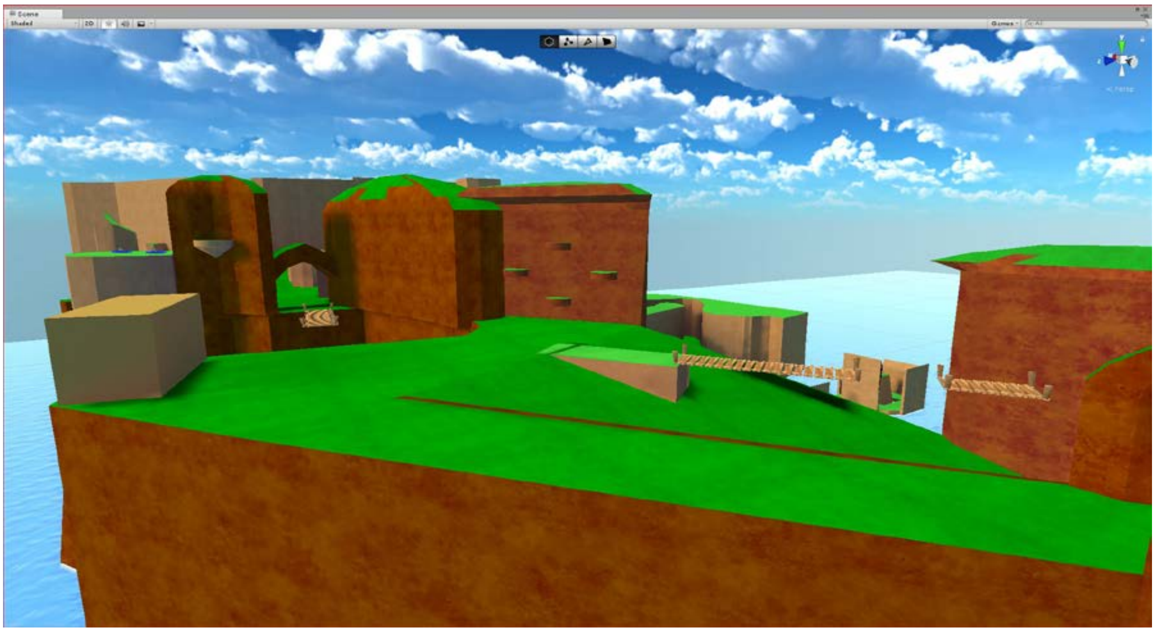
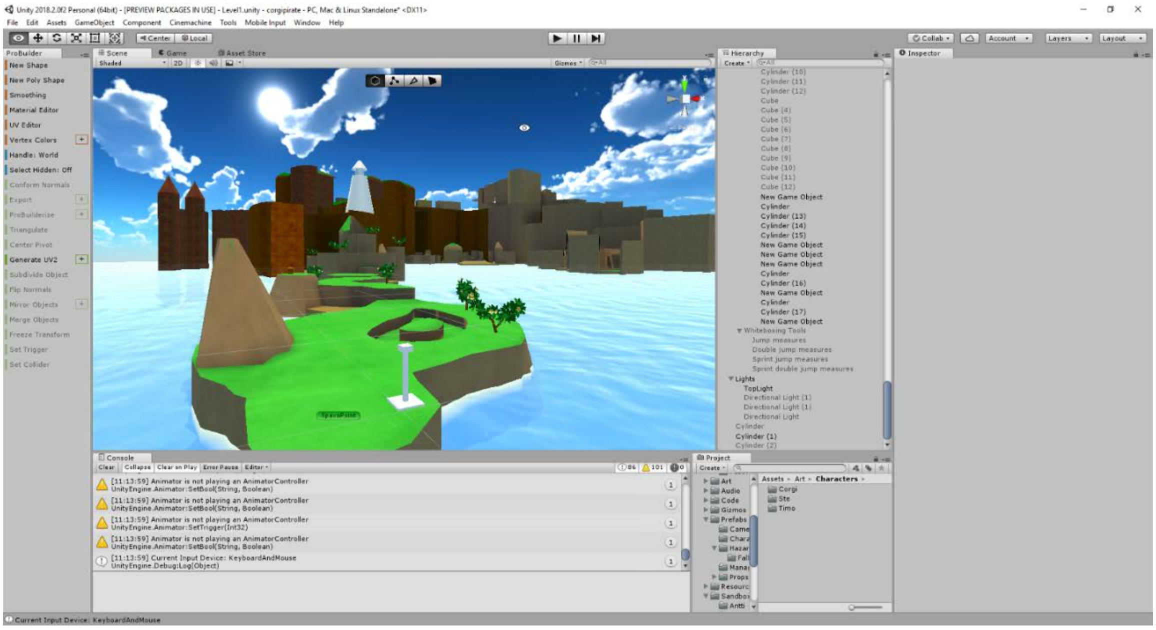


Figure 52. Level 1, area 4-2, first white-box draft.

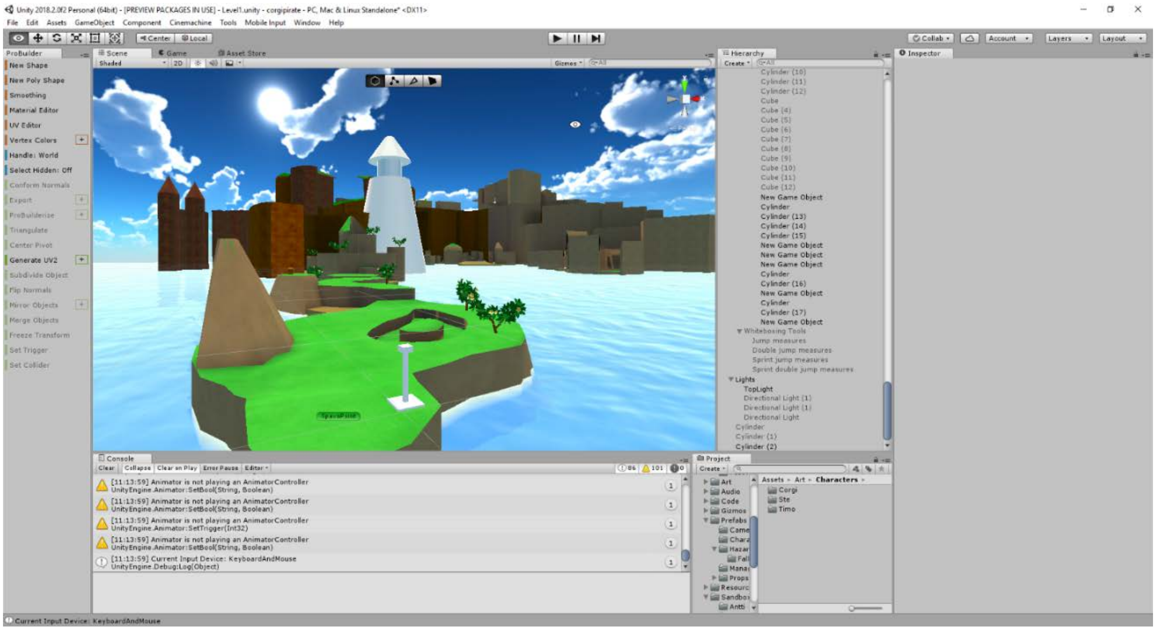
Since we were about to wrap up the level, I realized that our level still does not have a strong visual trademark, other than being based around a bay area. But considering that our later levels would most likely also include much of the same elements which were featured in this level, I thought that it would not be enough of a visual trademark. I began brainstorming with our lead artist about what would fit the level's theme and act as a strong visual trademark for the level. Eventually we decided that the level's visual trademark would be a lighthouse, which would also act as the level's end goal, serving two purposes at once.


Unity 2018.2.0f2 Personal (64bit) - [PREVIEW PACKAGES IN USE] - Level1.unity - corgipirate - PC, Mac & Linux Standalone* <DX11>
Console
Clear
Collapse
Clear on Play
Error Pause
Editor

```
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetBool(String, Boolean)
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetBool(String, Boolean)
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetTrigger(Int32)
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetBool(String, Boolean)
[11:13:59] Current Input Device: KeyboardAndMouse
UnityEngine.Debug:Log(Object)
```

Figure 53. Level 1, starting view from area 1.

Since area 4 would end on top of the cliff, we logically tried placing it there. However, it didn’t really pop out from the environment as well as we hoped. The lighthouse would have to be recognizable from almost anywhere in the level in order to be a level-defining landmark that would also help the player navigate towards their end goal. We tried different ways of making it pop out more and scaling proved most efficient.


Unity 2018.2.0f2 Personal (64bit) - [PREVIEW PACKAGES IN USE] - Level1.unity - corgipirate - PC, Mac & Linux Standalone* <DX11>
Console
Clear
Collapse
Clear on Play
Error Pause
Editor

```
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetBool(String, Boolean)
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetBool(String, Boolean)
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetTrigger(Int32)
[11:13:59] Animator is not playing an AnimatorController
UnityEngine.Animator:SetBool(String, Boolean)
[11:13:59] Current Input Device: KeyboardAndMouse
UnityEngine.Debug:Log(Object)
```

Figure 54. Level 1, starting view from area 1.

By making the lighthouse a lot bigger, it became a much stronger contrast in the environment and could be seen from almost anywhere. We were also able to include it in the starting view of the level, establishing the level's goal as soon as the player starts the level. And with this, level 1 was finally finished, ready to be sent over to our level artist. Or so we thought.

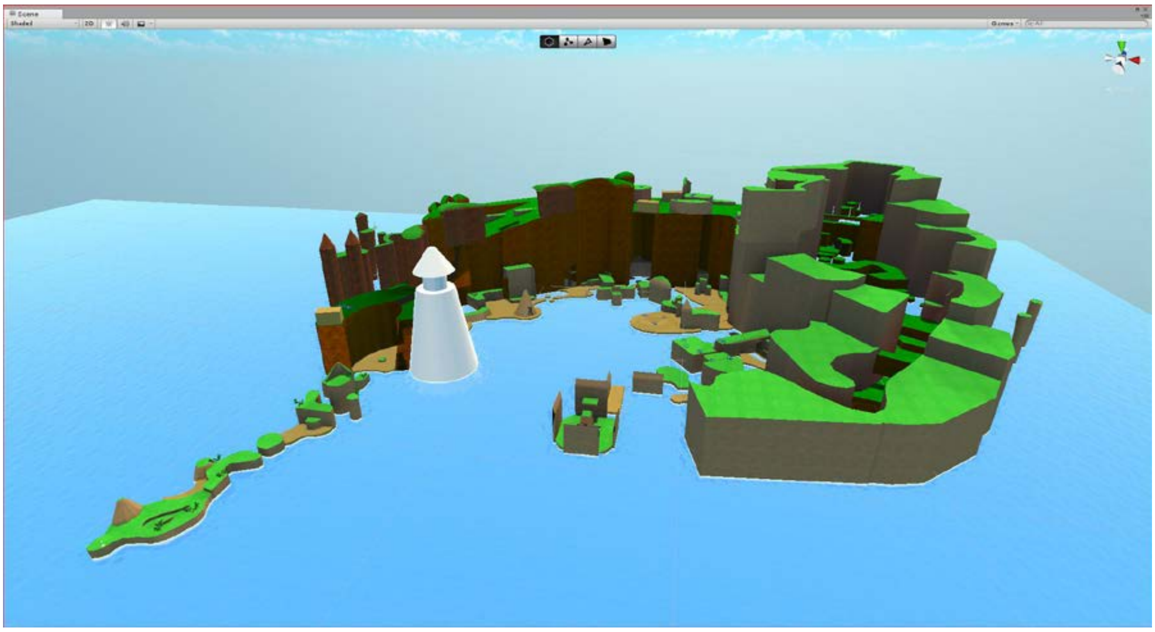
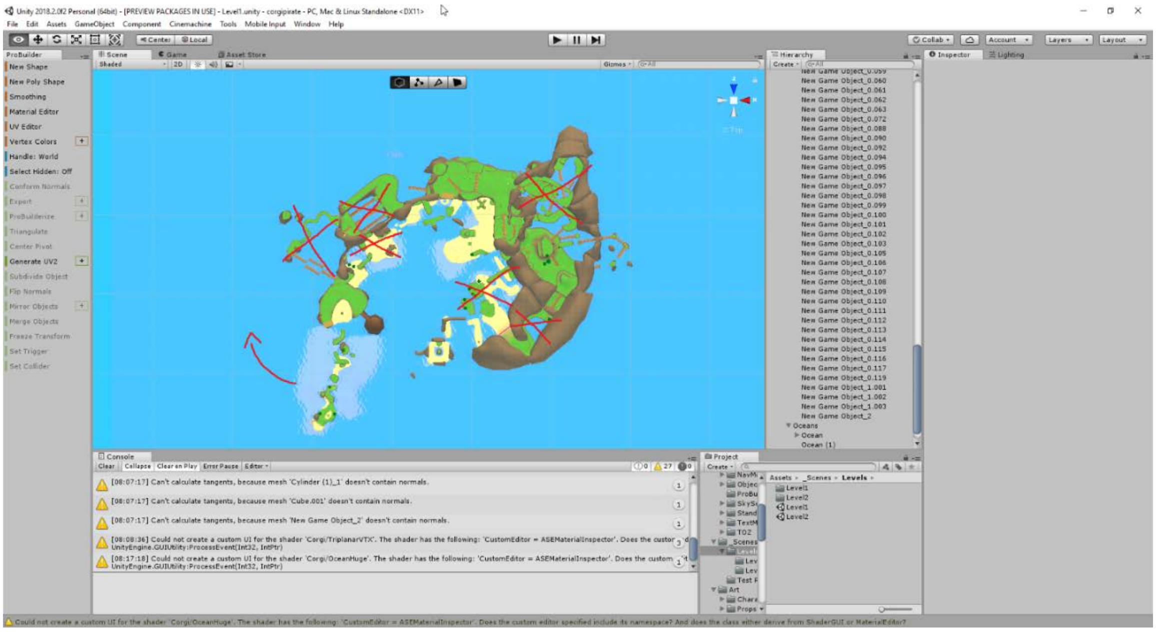


Figure 55. Level 1, fully white-boxed.

## 4.3 Level 1 division

Once the level was thought to be finished, with merely some smaller iterations left to do, we started playtesting it intensively. During these playtests we found out that the level was far too large for its own good. A common observation was that most players actively explored every nook and cranny in the level, until at some point they started just ignoring all the explorable areas and just wanted to finish the level. This usually happened at the end of area 3. Playtimes in the level ranged from 11 minutes to a whopping 29 minutes. This was much too long from what we had originally envisioned.



```
Unity 2018.2.0f2 Personal (64bit) - [PREVIEW PACKAGES IN USE] - Level1.unity - corgipirate - PC, Mac & Linux Standalone <DX11>
```

```
[08:07:17] Can't calculate tangents, because mesh 'Cylinder (1)_1' doesn't contain normals.
[08:07:17] Can't calculate tangents, because mesh 'Cube.001' doesn't contain normals.
[08:07:17] Can't calculate tangents, because mesh 'New Game Object_2' doesn't contain normals.
[08:08:36] Could not create a custom UI for the shader 'Corgi/TriplanarVTX'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custor
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
[08:17:18] Could not create a custom UI for the shader 'Corgi/OceanHuge'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
```

```
Could not create a custom UI for the shader 'Corgi/OceanHuge'. The shader has the following: 'CustomEditor = ASEMaterialInspector'. Does the custom editor specified include its namespace? And does the class either derive from ShaderGUI or MaterialEditor?
```

Figure 56. Top-down view of level 1 and notes on shortening the level.

I spent a long time thinking what would be the most optimal solution for this issue. Above is a picture of how I planned to cut and change the different areas of level to make it match the 10-minute target length. I tried removing the less important areas in the level and focus on the key elements. However, cutting and moving the areas would still require a lot of work in order to reconnect them into a comprehensive whole. All solutions I came up with would still require reworks in the level and with the schedule we had, I was reluctant to do so.

Eventually, we decided that the most cost-effective solution would be to slice the existing level into two levels. The slice could be made quite naturally, because areas 3 and 4 focus on almost entirely different mechanics than areas 1 and 2. This would require redesigns for the end segments of both halves, but it was not that big of an issue. Level 1 would focus on the bay area and level 2 would take place in the forest and cliff segments. Level 1 would still retain the lighthouse trademark, because a beach would be a much more logical location for it. We would have to decide a new trademark for level 2.

Once the whole team agreed to this, I divided the level into two and started working on each level's new end segments.

## 4.4 Level 1 conclusion

I cut off level 1 at the point where the beach would head in-land. I decided that the level would end at the other end of the bay area, giving the level a much stronger identity by focusing on just the bay theme.

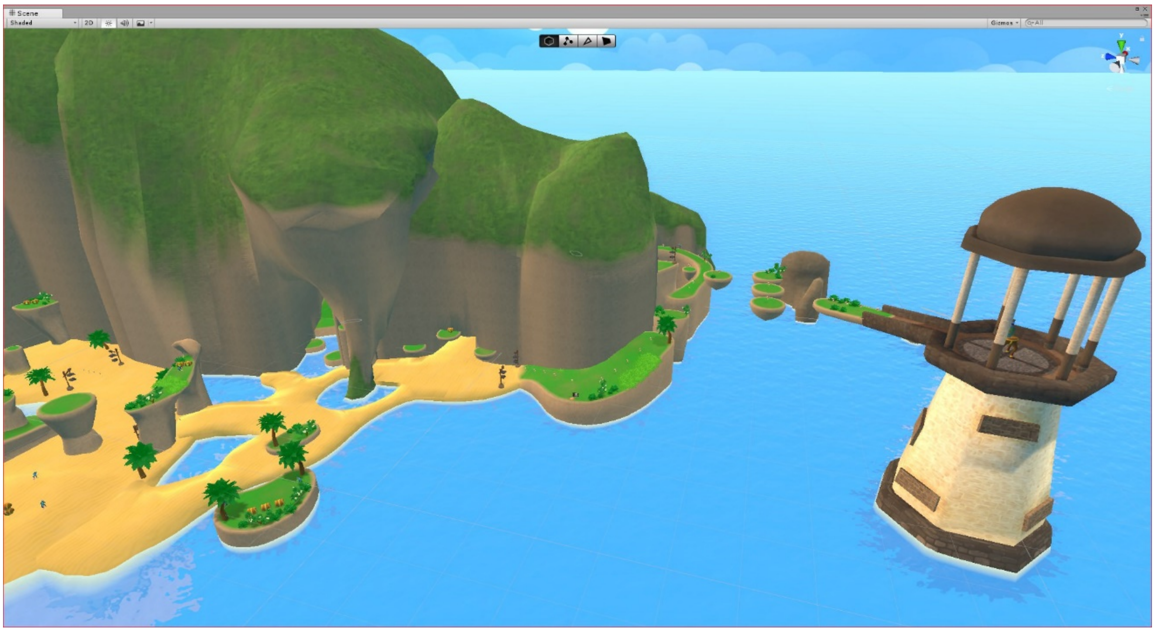


Figure 57. Level 1, part 2-3, final version.

Since level 1’s end was not going to be near the start of the level anymore, but rather at the other end of the bay area, I decided to move the lighthouse there. I had to design a path that would lead the player from the beach level to the top of the hulking lighthouse. I did not have much time to do this, so I just quickly planned a linear path with fresh platforming challenges and some side-paths.

Once that was done, level 1 didn’t require any other changes to be finished, however some iterations were still made later on based on playtest feedback. Those will be explained in the retrospect section in more detail.

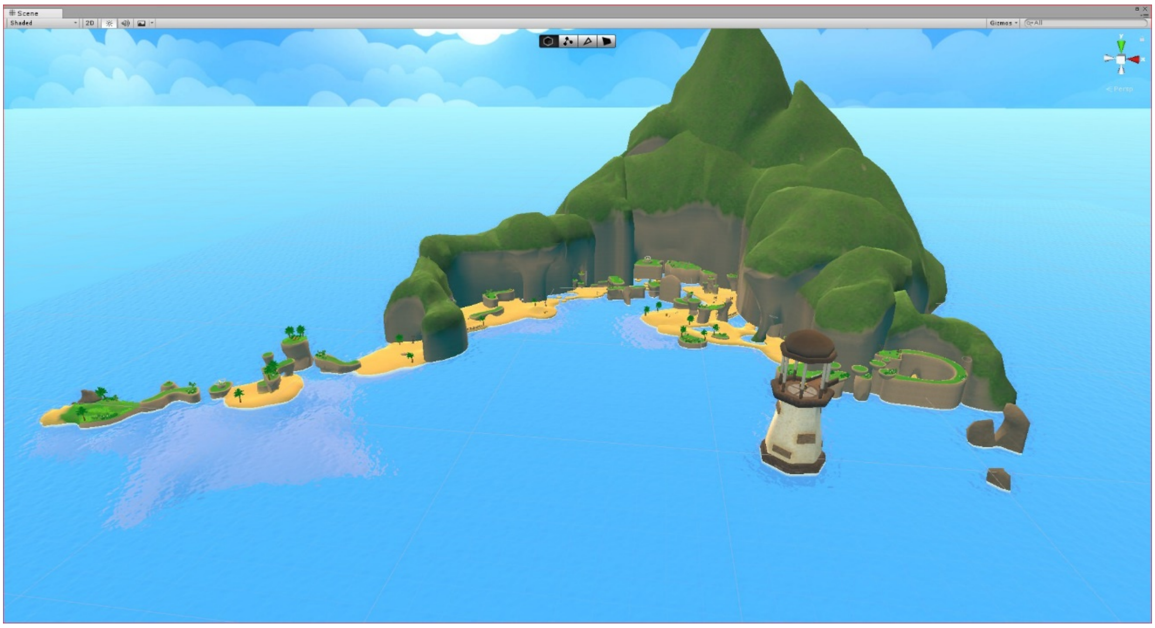


Figure 58. Level 1, final version.

Level 1, Bay Six, revolves completely around the bay area, ending at the lighthouse on the other end of the bay. The level teaches the player about jumping, double jumping, attacking and basic enemies. The level starts off with a linear segment, ensuring that the player learns the basic mechanics of the game before opening up to wider segments with more incentive for exploration. The actual bay area provides more side paths and optional challenges on higher ground for skilled players, while keeping the critical path clear on the beach. The end of the level funnels into a more linear segment again, providing some final platforming challenges before the player reaches the lighthouse and completes the level. Level geometry, lantern props, bone collectibles and the bright-yellow beach are used to guide the player on the critical path.

## 4.5 Level 2 conclusion

Level 2 required slightly more redesign than level 1. If level 2 was going to start from the former area of level 1, I figured it would need a better starting area too. One that would welcome the player properly to the forest theme before introducing the sprint tutorials and whatnot. In addition to that, I also had to redesign the end for level 2. Again, I discussed about level 2’s end goal landmark with our lead artist. Since the later areas of level 2 were going to be on a high cliff, we figured that the landmark could be a large rock formation in the ocean.

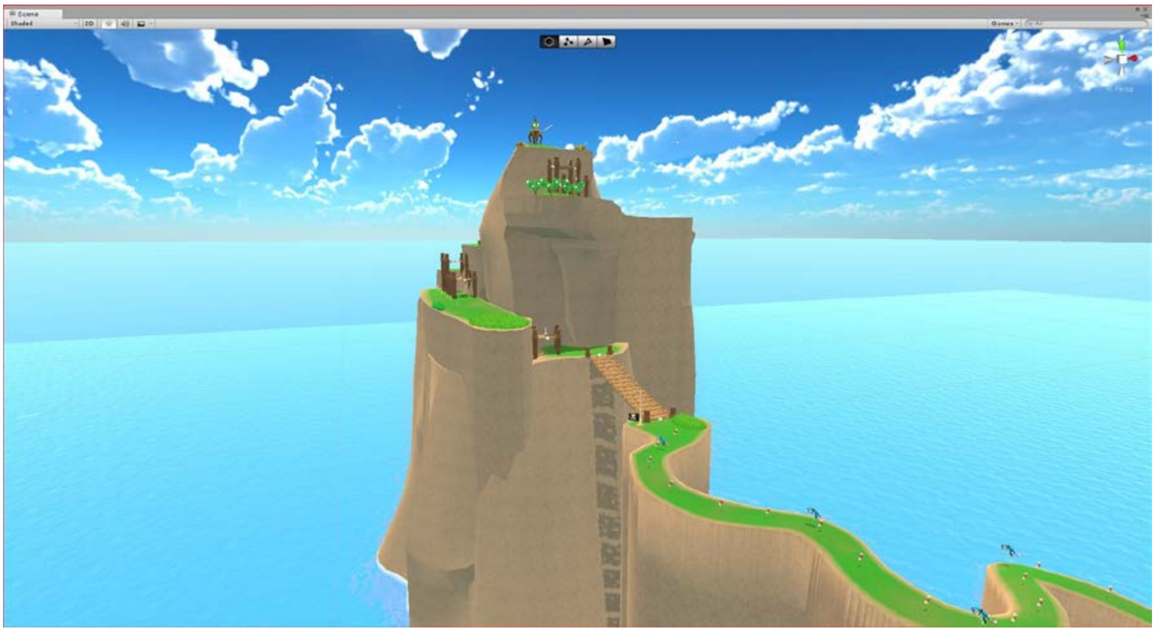


Figure 59. Level 2, area 2-3, final version.

Thus, I based the end of level 2 around a mountain that the player would climb around before reaching the end of the level at the summit. Again, I made the end of the level fairly linear as a test of the player’s understanding of the mechanics taught in the level. I had to consult one of our programmers to create a custom camera system for this segment, to ensure the intuitiveness of progression.

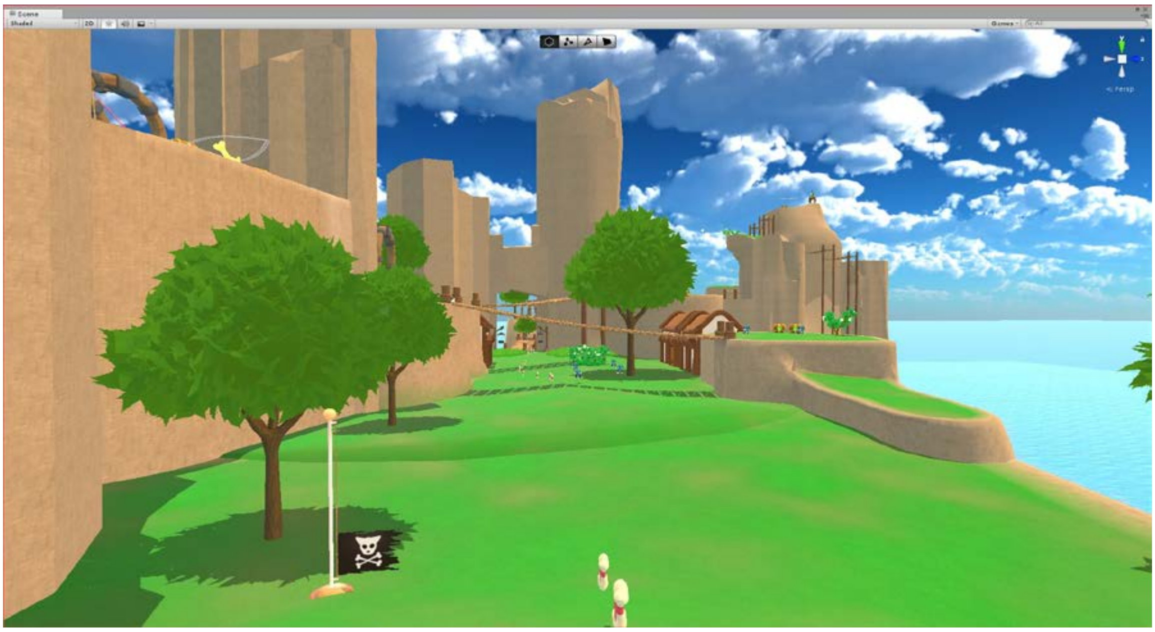


Figure 60. Level 2, entrance to area 2-1, the level's end can be seen in the back.

I also tried to compose some views in the level so that the player would be able to see the mountain and the goal on top of it before reaching it. Unfortunately, due to the level's shape, I was unable to include the mountain in the level's starting view properly without even more redesigns.

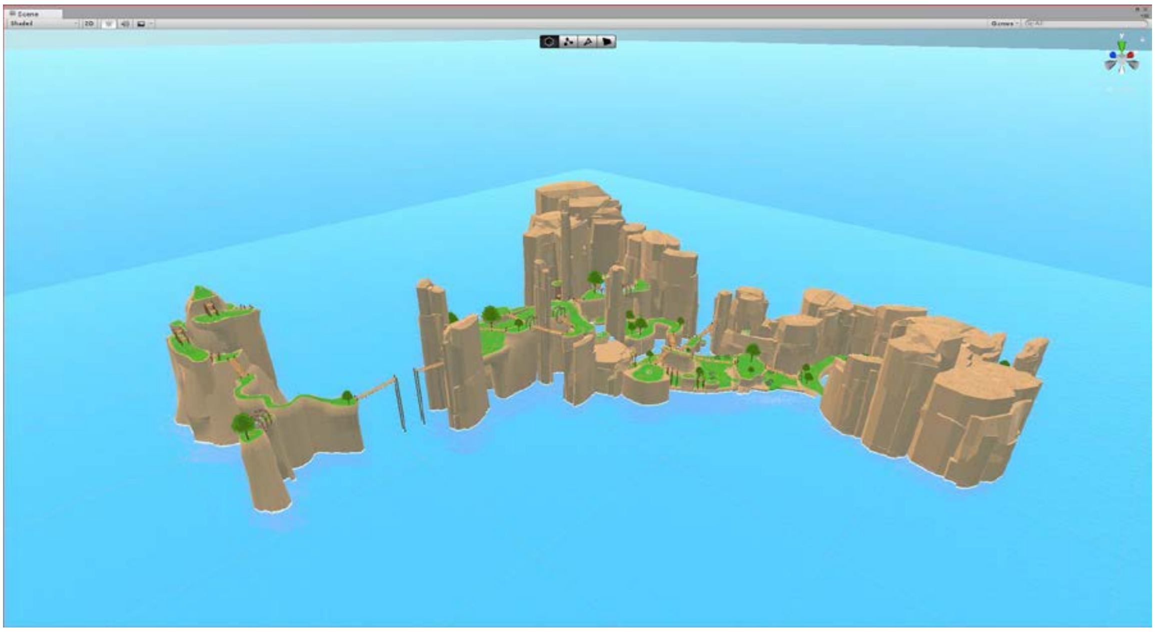


Figure 61. Level 2, final version. Some walls have been hidden for a clearer view.

Level 2, Barkbados Island, starts off from a thick forest with plenty of side paths to explore, providing a much different environment than level 1. The level teaches the player about sprinting and rickety bridges early on before providing plenty of optional challenges based on the mechanics taught to the player. As the level progresses from the forest area to some seaside cliffs, the level gets much more linear and eventually ends on the top of a tall mountain. Despite being much more open than the previous level, the critical path is again being highlighted with the use of geometry, lanterns and bone collectibles. In addition to these, level 2 also uses its rickety bridges to highlight different paths to the player.

After these redesigns and some final playtest iterations, levels 1 and 2 were finally complete and the length of each level was approximately 7 minutes. In addition to that, playtesters were able to navigate much easier in these shorter levels and did not get impatient at any point.

# 5 Retrospect

This process was a great learning experience for me. If I could redo both levels from the ground-up, I would approach them entirely differently than I did at the beginning of this assignment. There are a lot of small things that I learned during this process, but for the sake of this thesis, I focus on just the most important ones in this segment.

## 5.1 Constraints and checklists

Having a checklist of constraints at the very beginning is definitely useful. I found myself returning to those questions quite often especially in the concepting phase. Making sure I know what my level is about beforehand helped me from getting sidetracked with the level's content. The questions I found myself relying upon the most often were the level's purpose, desired player experience, set-pieces and player guidance. Later I realized that I should have paid more attention to the level's scale and visual trademark. It's really tough to estimate a level's scale from on-paper sketches, so I think it's important to white-box as soon as I have at least something planned. That way I can already playtest the level's scale and see how soon it escalates. I should also have thought more about the level's landmarks and trademarks in the beginning. The landmarks in both levels turned out to be really important and it would have helped if I had taken them into consideration much sooner.

## 5.2 Visual timeline and bubble diagram

When it comes to open levels like the ones in Captain's Tail, I found the visual timeline really difficult to use. I think that similar to movies, a planned timeline of events and pacing only works with mostly linear levels, where the designer can predict much more efficiently when and where the player is going to perform certain actions. That said, I think I still should have paid more attention to the level's pacing. I am still not quite sure how pacing should properly be planned in more open levels, but I am definitely going to experiment different techniques for it in the future.

But even if Pete Ellis' visual timeline was not as applicable as I had hoped, I found Mike Stout's bubble diagram quite useful in the concepting phase. It helped me determine the shape I had in mind for the level and keep it consistent through the entire process. Dividing the level into smaller areas also helped in both communication and the hierarchy of the level's construction during white-boxing. It also helped make each part of the level relevant, since each area had an assigned role to them that I was able to rely on when concepting and white-boxing. I am absolutely going to use bubble diagrams in the future as well and maybe even experiment my own variations of it based on the type of game I'm making.

## 5.3 Theoretical information

I searched the internet for credible articles and videos on the subject of 3D platformer level design, where to get started and what to look out for. As I researched, I came to the conclusion that there is not that much credible information on genre-specific level design, especially about 3D platformers. First-person shooters were the most common genre to find genre-specific level design information for and even most sources that focused on level design in general used first-person shooters as a base example. While there are some things that can be applied from genre to genre, most genres might be so different from one another that they follow an entirely different set of rules. This led me to the conclusion that there may be a demand for more genre-specific level design guides.

Despite that, most of the theoretical info about level design I dug up during my research came in handy at one point or another. I tried to keep all of it in mind every step of the way, but I always noticed later on whenever I had been neglecting some of them. For example, the navigation in the level was really difficult at first; people kept wondering about where to go and some accidentally started going through the level backwards without realizing it. This was because I had not used all the tools I had in my player guidance kit, such as landmarks and lighting. Sometimes areas seemed really cluttered and players were overwhelmed with options. This was because I had not used composition properly to make harmonious views where all important elements are shown clearly to the player. There might have been even more problems if I had not done the research at all. Turns out some things in level design do carry over from other genres, though it may be difficult at first to determine which ones.

But no amount of theoretical information is going to save me if the game does not feel fun for the player. I learned through playtesting that no matter how much sense the level's design makes in theory, if the player does not find the experience fun, the designer is to blame, not the player. Desired player experience is important to keep in mind at all times when designing levels, but the best results can be reaped through constant playtesting and iteration. It's also important to start testing early, so the applied theories can be confirmed before starting to trust them blindly. Even after multiple iterations, I am still not entirely happy with the levels myself, but that just means I have to reserve more time for playtesting and iteration in the future.

## 5.4 The process

I feel like I spent too much time on the planning and concepting phase. While I think it is important to determine the level's purpose and other defining elements at the beginning, I don't think I should've used as much time on it as I did. In addition to that, I spent many days on concepting, because I kept remaking new versions of the existing rough maps at first. In retrospect, I think it was pointless to make different versions of the same area on paper, since I only white-boxed one of them. I also made the rough maps too detailed at first, when I should have just moved on to white-boxing as soon as I had a base concept ready. I tried skipping the sketching phase completely at one point, but I was hindered by mental blocks while trying to white-box an area from scratch. I discovered that I need at least some rough maps to base my white-boxing on, but I should not spend too much time on them. I found it most useful to just settle on a rough map of a single segment of an area as soon as possible, white-box it and continue concepting from there again. Concepting and white-boxing seamlessly helped me eliminate some problems on both ends and I am going to keep doing that in future projects as well.

I found the metric measures really useful while white-boxing. For example, for jump measures I just created rectangles representing the character's different jump heights and lengths. I could then just compare gaps to the rectangles and see how manageable certain jumps in the level would be without needing to test it all the time. Easily referable measures saved me a lot of time in an already busy schedule.

I noticed that by focusing just on the level's functional content I ended up designing only logical challenges and explorable areas into the levels, but not necessarily any interesting structures or any kind of recognizability. At least not in the concepting phase. I think that I should not only think about the different things I intend to teach and challenge the player, but also concept some memorable structures, key areas, landmarks etcetera, and then design how I can work with those together. In theory this would also give me a clearer goal to work towards when designing the individual parts of the level, instead of just putting a bunch of fully gameplay-focused content together and calling it done. In many 3D platformers, including this one, exploration is a key element. Ease of navigation and making parts of a level recognizable in these types of games is crucial.

## 5.5 Final words

After these two levels, I proceeded to design a third level for the demo of Captain's Tail before its release. I did my best to apply everything I had learned during this process. Initially I was going to talk about the third level in this thesis too, but I did not feel like it would have contributed to the subject enough to be relevant. I also want to focus on how to make my level design process more modular in the future, so that I can further improve my work efficiency. Still, this process has not only taught me a lot about 3D platformer level design, but also about level design and game design in general. I can feel the improvement in my bones and I cannot wait to apply what I have learned into more practice.

# 6 References

## 6.1 Information sources

1. Gamasutra article “Beginning Level Design” https://www.gamasutra.com/view/feature/131736/beginning_level_design_part_1.php

2. Techopedia’s definition of level design https://www.techopedia.com/definition/88/level-design

3. Wikipedia’s definition of level design https://en.wikipedia.org/wiki/Level_design

4. A youtube video “The Level Design Process (Design Talk)” https://www.youtube.com/watch?v=TyXvj-t2xks

5. World of Level Design article “Map Creation Guidelines” http://worldofleveldesign.com/categories/level_design_tutorials/map_creation_guidelines.php

6. Black Shell Media article “Fundamentals of Level Design: Where Does One Start?” https://blackshellmedia.com/2016/12/05/fundamentals-level-design-one-start/

7. 80 Level article “Introduction to Level Design for Games” https://80.lv/articles/introduction-to-level-design-for-games/

8. Envatotuts+ article “A Beginner’s Guide to Designing Video Game Levels https://gamedevelopment.tutsplus.com/tutorials/a-beginners-guide-to-designing-video-game-levels--cms-25662

9. World of Level Design article “Single Player Level Design Pacing and Gameplay Beats” http://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/level-design-pacing-gameplay-beats-part1.php

10. Purplepwnystudios blog post “"Visual Hierarchy for Game Developers: A Practical Guide to Making Important Stuff Seem Important"

http://purplepwny.com/blog/visual_hierarchy_for_game_developers_a_practical_guide_to_making_important_stuff_seem_important.html

11. The psychology of video games article “Why do color coded clues in level design work?”. http://www.psychologyofgames.com/2013/09/why-do-color-coded-clues-in-level-design-work/

12. Gamasutra article “Designing better levels through human survival instincts”. https://www.gamasutra.com/view/feature/134779/designing_better_levels_through_.php

13. Gamasutra article “ Color in games: An in-depth look at one of game design's most useful tools”. https://www.gamasutra.com/blogs/HermanTulleken/20150729/249761/Color_in_games_An_indepth_look_at_one_of_game_designs_most_useful_tools.php

14. How to not suck at game design article ” Color Theory For Game Design 2 of 4 – Glyphs And Neutrals”. http://howtonotsuckatgamedesign.com/2014/11/color-theory-game-design-2-4-glyphs/

15. A Youtube video “Why Nathan Drake Doesn't Need a Compass | Game Maker's Toolkit”. https://www.youtube.com/watch?v=k70_jvVOcG0

16. World of level design article “Functional lighting”. http://www.worldofleveldesign.com/categories/wold-members-tutorials/magnar_jenssen/functional-lighting-magnar-jenssen.php

17. A Youtube video “Splinter Cell (Pandora Tomorrow) – Level design”. https://www.youtube.com/watch?v=CGQY3eLMjsY

18. Gamasutra article “Composition in Level Design” https://www.gamasutra.com/blogs/MateuszPiaskiewicz/20140817/223513/Composition_in_Level_Design.php

19. Envatotuts+ article “Level Design: Views and Vistas” https://gamedevelopment.tutsplus.com/articles/level-design-views-and-vistas--cms-25036

20. A Youtube video “Pacing – How Games Keep Things Exciting – Extra Credits” https://www.youtube.com/watch?v=5LScL4CWe5E

21. A Chaotic Stupid article “Trinity 6 – Intensity Ramps” http://www.chaoticstupid.com/trinity-6-intensity-ramps/

22. Gamasutra article “How to control the pacing of an open world game” https://www.gamasutra.com/view/news/173239/How_to_control_the_pacing_of_an_open_world_game.php

23. A GDC talk “Ten Principles for Good Level Design” https://www.youtube.com/watch?v=iNEe3KhMvXM

24. Wikipedia’s definition of “visual language” https://en.wikipedia.org/wiki/Visual_language

25. Wikipedia’s definition of “Game mechanics” https://en.wikipedia.org/wiki/Game_mechanics

26. Wikipedia’s definition of “Gameplay” https://en.wikipedia.org/wiki/Gameplay

## 6.2 Image sources

1. A screenshot of Mega Man 10, a side-scrolling 2D platformer game. https://www.mobygames.com/images/shots/l/430526-mega-man-10-xbox-360-screenshot-strike-man-s-stage.jpg

2. A screenshot of The Land of the Scurvy Dog. https://img.itch.zone/aW1hZ2UvMTAzMDAyLzQ5NDEzNy5qcGc=/original/l3te5D.jpg

3. A screenshot of Uncharted 3: Drake’s Deception. https://www.mobygames.com/images/shots/l/813537-uncharted-3-drake-s-deception-playstation-4-screenshot-drake.jpg

4. A screenshot of KillZone: Mercenary. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part1-11.jpg

5. A screenshot of Dark Souls 3. http://i.imgur.com/pgcqPXG.jpg

6. A screenshot of Super Mario 64. https://vignette.wikia.nocookie.net/super-mario-64-official/images/a/a5/Super_Mario_64_Star_get.jpg/revision/latest?cb=20160525153840

7. A screenshot of Dead Space 3. https://vignette.wikia.nocookie.net/deadspace/images/9/96/DS3_screenshot_Feeder_01.jpg/revision/latest?cb=20120728200254

8. A screenshot of Super Meat Boy. https://vignette.wikia.nocookie.net/supermeatboy/images/9/94/Screenshot_1.jpg/revision/latest?cb=20100909200030

9. A screenshot of Portal. http://www.gamershell.com/static/screenshots/1/11183/240384_full.jpg

10. A screenshot of the King's Row level in Overwatch. https://dotesports-media-cdn-tqgiyve.stackpathdns.com/wp-content/uploads/2018/08/11175946/bc6111bb-db54-442a-a960-a7e03c0d29d5.jpg

11. A screenshot of Half Life 2: Episode Two. https://www.mobygames.com/images/shots/l/264328-half-life-2-episode-two-windows-screenshot-two-kitchen-dwellers.jpg

12. A screenshot of Crash Bandicoot 3: Warped. https://coubsecure-s.akamaihd.net/get/b189/p/coub/simple/cw_timeline_pic/9668221dc6f/2d26886707495e9a9f4c6/big_1498422357_image.jpg

13. A screenshot of Ratchet & Clank. https://vignette.wikia.nocookie.net/ratchetandclank/images/6/69/Blackwater_City.png/revision/latest?cb=20130225172749&path-prefix=es

14. Concept art from Jak & Daxter. https://www.gamasutra.com/features/20020710/jakfig3.gif

15. Rough sketches from Captain's Tail. Self-taken image.

16. An example bubble diagram by Mike Stout. https://cms-assets.tutsplus.com/uploads/users/1159/posts/25662/image/BubbleDiagram01.png

17. The "three-act structure" narrative model. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part1-03.jpg

18. Pete Ellis' visual timeline of KillZone: Mercenary's first level. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part1-02-large.jpg

19. A screenshot of Mirror's Edge. https://aaronhsyong.files.wordpress.com/2014/10/mirrors-edge-red-hints.jpg

20. A screenshot of The Last of Us. https://cdn.atomix.vg/wp-content/uploads/2014/07/14667031945_5946a45e48_o.jpg

21. A screenshot of The Legend of Zelda: Breath of the Wild. https://d.wattpad.com/story_parts/547994861/images/151c3caac02819e211551235202.png

22. A screenshot of Left 4 Dead 2. https://youtu.be/k70_jvVOcG0?t=2m43s

23. A screenshot of Journey. https://adventuregamers.com/images/screenshots/23827/journey-game-screenshot-10-b.jpg

24. A screenshot of Journey. https://mygaming.co.za/news/wp-content/uploads/2015/04/Journey-Screenshot-1.jpg

25. A screenshot of Jak & Daxter. https://www.ps4home.com/wp-content/uploads/2016/03/Jak-and-Daxter-1.jpg

26. A screenshot of Super Mario 64. https://vignette.wikia.nocookie.net/supermarioitalia/images/7/76/Bob-omb_Battlefield_64.png/revision/latest?cb=20130714191735&path-prefix=it

27. A screenshot of Spyro the Dragon. http://s3.amazonaws.com/gamefndr/screenshots/000/010/486/medium/457ae4ea0ecaff6950a3bec81030ea66.jpg?1371748515

28. An example of a focal point. https://gamasutra.com/db_area/images/blog/223513/Compo_perspective.jpg

29. A screenshot of Dishonored. https://gamasutra.com/db_area/images/blog/223513/Counterpoint-dishonored_1.jpg

30. A screenshot of Half Life 2. https://cms-assets.tutsplus.com/uploads/users/1159/posts/25036/image/Vista_halflife2_trainsandmonitor.jpg

31. A screenshot of Ratchet & Clank (2016). https://cms-assets.tutsplus.com/uploads/users/1159/posts/25036/image/rcps4gasparopeningview.jpg

32. A graph of the pacing in Star Wars: A New Hope. https://jonathanbarkhuizen.files.wordpress.com/2016/07/pacing_01_star_wars.gif

33. A graph of the pacing of the level “Desperate Times” in Uncharted 2: Among Thieves. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part3-02-large.jpg

34. A screenshot of Uncharted 2: Among Thieves. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part3-04.jpg

35. A screenshot of Uncharted 2: Among Thieves. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part3-13.jpg

36. An example of repetitive pacing. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/images/pacing-gameplay-beats-part2-16-large.jpg