UQAC
Université du Québec
à Chicoutimi

Motor-difficulty evaluation framework of 2D platformer games' levels.

par Simon Lescieux

Mémoire présenté à l'Université du Québec a Chicoutimi en vue de l'obtention du grade de Maître ès science (M.Sc.) en informatique

Québec, Canada

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The continual growth of the video game industry has led to a significant increase in the scale of development of new projects, requiring considerably more resources to complete. In an attempt to counteract some of the resource requirement inflation, we propose a motor difficulty evaluation mathematical model as a development tool to avoid difficulty targeted playtesting. The proposed model deviated from other currently researched approach by evaluating difficulty in a deterministic manner and without player data. We define the notions of reachability and danger of a game level based on the properties and possible actions of the player's avatar. Reachability and danger are used to describe two difficulty evaluation metrics: one based on the relative area coverage of reachable to danger, and one based on the perimeter of overlapping areas. The metrics are put to test in levels taken from the game *Super Mario Bros.* and the results of both metrics are compared to each other as well as with the Nishikado difficulty. We underline current limitations of the model and their effect on the results and provide elementary solutions. We conclude by proposing a possible timeline of future research to expend our model into a general commercial applicable product.

RÉSUMÉ

La croissance continue de l'industrie du jeu vidéo a conduit à une augmentation significative de l'échelle de développement de nouveaux projets, nécessitant beaucoup plus de ressources pour être achevés. Dans une tentative de contrecarrer une partie de l'inflation des besoins en ressources, nous proposons un modèle mathématique d'évaluation de la difficulté motrice comme outil de développement afin d'éviter les tests de difficulté ciblés. Le modèle proposé se différencie des autres approches actuellement étudiées en évaluant la difficulté de manière déterministe et sans données sur les joueurs. Nous définissons les notions d'accessibilité et de dangerosité d'un niveau de jeu en fonction des propriétés et des actions possibles de l'avatar du joueur. L'accessibilité et le danger sont utilisés afin de définir deux mesures d'évaluation de la difficulté : l'une basée sur la couverture relative de la zone accessible au danger, et l'autre basée sur le périmètre de l'intersection des zones. Les métriques sont mises à l'épreuve dans des niveaux tirés du jeu *Super Mario Bros.* et les résultats des deux métriques sont comparés entre eux ainsi qu'avec la difficulté Nishikado. Nous soulignons les limites actuelles du modèle et leurs effets sur les résultats et proposons des solutions élémentaires. Nous concluons en proposant un calendrier possible de recherches futures pour étendre notre modèle à un produit applicable commercialement général.

INTRODUCTION

Since their inception, video games have continuously grown in popularity throughout the world. For instance, the video games industry's revenue in 2020 was more than triple that of the film industry and the music industry[1], as shown by Figure 1. In terms of year-over-year growth, an increase in revenue of around 20% was seen from 2019 to 2021.[2] These numbers were in part achieved through an important movement of diversification of games, with the express objective of reaching out to new audiences and markets. These diversification initiatives can be seen on multiple levels, such as game genres and platforms.

One of the video game developer's principal objectives is to provide the player with an enjoyable experience. In the case of a game designed for entertainment purposes, the experience can also be described as providing engaging mechanics and an appropriate level of difficulty. It is still not clear as to what engaging and appropriate exactly means in this context. Since every player is different with its own set of capabilities and experiences, these notions are fundamentally subjective. The same could be said for serious games: they require information to be presented in a way that is understandable and digestible.

Regardless of their type, all games provide a set of challenges. A game's challenges vary based on its genre, objective, and target audience. For example, casino games such as *slot*

[1] https://www.statista.com/statistics/1132706/media-revenue-worldwide/

[2] https://theesa.ca/wp-content/uploads/2021/11/esac-2021-final-report.pdf

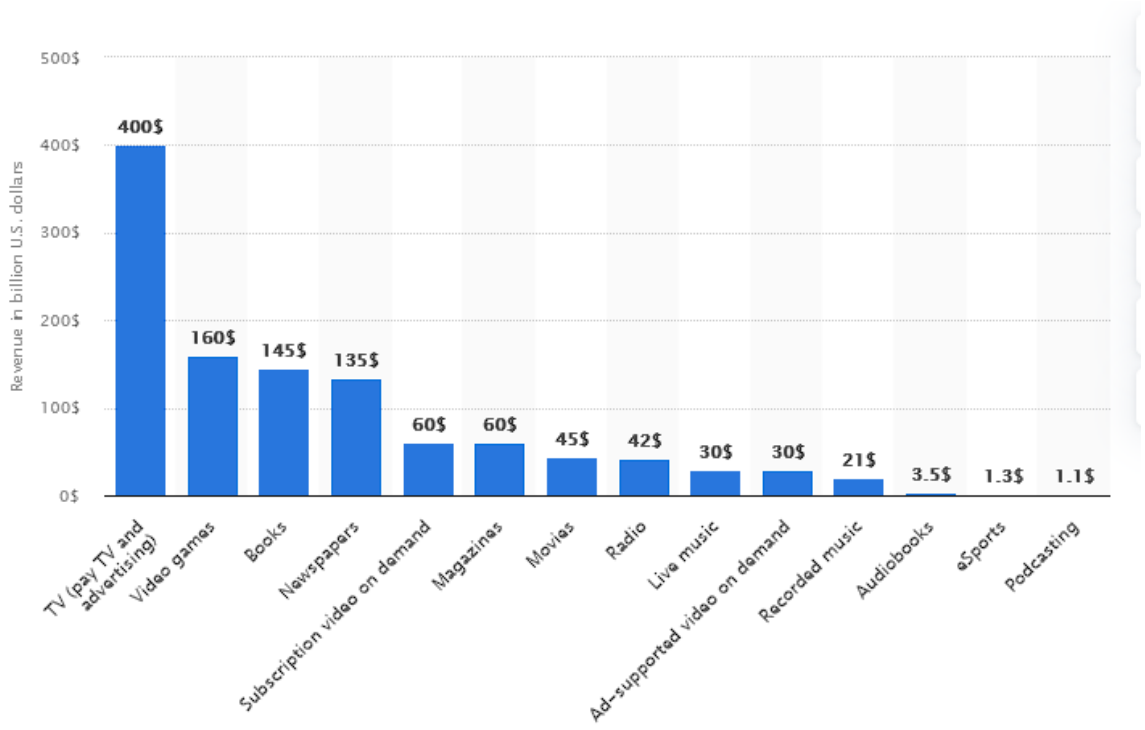


Figure 1: Estimated media revenue worldwide in 2020, by category. Taken from `statista.com/statistics/1132706/media-revenue-worldwide`**.**

machines most often get their challenge from elements of randomness and unknowns, or *alea*.Caillois (1958) Meanwhile, rhythm games get their challenge from a more perception and dexterity-based set of mechanics. The range of challenges available allows the video game industry to appeal to a vast number of players.

As players progress through a game, they acquire greater understanding of the game and its controls, thus allowing players to achieve better performance with time and practice. In other words, this process gradually makes any game easier as the player progress through the game. This reveals two important facts about video games: first, the difficulty of any game changes based on the time and effort invested by the player. In other words, that "difficulty" is subjective and cannot be used as an absolute point of reference. Second, the game should provide an increasing level of challenge to keep the difficulty perceived by the player at least constant or increasing. For example, this progression could be made throughout the game in a linear manner where the slope would be the estimated gain of competence by the player.

A practical way of managing the increase of difficulty in a video game is to divide it into a series of relatively small parts called levels. These levels are generally sequential, and each proposes a set of challenges. For each level, the difficulty can be approached in two major ways: keep each level's challenge constant and up the challenge in-between levels, or increase the challenge amidst the levels themselves. Remark that these techniques are not mutually exclusive. Either way, this design allows for a somewhat modular development approach as each level can be built in parallel, allowing for easier modifications and challenge adjustments by minimizing game-wide consequences.

As it is currently understood, every element of a game that participates in achieving their goal of providing a good experience is part of the flow of that game. The flow can be described as the state a player is in as it experiences the game. A major component of the flow is immersion, which is heavily driven by difficulty. Although difficult to engineer, a state of flow is a necessity for the success of every game. As shown in **??**, a game that is trivially easy will quickly become boring and unenjoyable. The same can be said about a game that offers an overwhelmingly difficult experience, only due to frustration.

Since difficulty plays a significant role in the overall enjoyment of a game as it creates and maintains the player in a state of *Flow* Nakamura et Csikszentmihalyi (2014), it is imperative that the difficulty curve be developed with particular care. The *Flow* describes the state of mind of the player in which its immersion is maximized. As stated previously, the fundamental subjectivity brought on by the players makes this task especially demanding on the developers. Balancing the difficulty of a game is to bring a set of individually complex elements such as the number of enemies as well as their reaction speed, their pattern, the level's topology and more to an equilibrium.

This is most often achieved through a process of trial and error, and not in a straightforward fashion. In other words, designers and developers create a multitude of iterations of the same parts of the game. In-between each iteration, a playtesting session is held, where some players

unrelated to the project and developers alike try out the game in its current state and provide opinions and feedback about their experience. The information that comes out of those playtest sessions is used to inform the adjustments to be made and the next set of iterations. The cycle of development to playtest back to development generally continues up until the very last moments before the official launch of the game. The last few iterations of the game to be sent to playtest are often referred to as *Beta* versions. This iterative process takes up a substantial amount of time and resources as it leads to parts of the game having to be reworked, sometimes multiples times over.

Of course, experienced developers are a great asset when it comes to designing levels. That experience allows them to intuitively reach a better balance and identify possible problems before or without playtesting. Even then, external tools can provide additional information about a level and the game along multiple axis, notably difficulty. The more information available to the developers, through experience or other means, the more time can be saved by virtue of needing less playtesting to be done to achieve a similar result Volkovas et al. (2020). Some currently available tools, like machinations, provide insightful information. However, that information is mostly based around game mechanics, dynamics, and evolution of the game economy Van Rozen et Dormans (2014). This kind of information is significantly less useful when it comes to evaluating the difficulty of a level than for something resembling topological analysis, where the levels are studied on a physical scale, notably through terrain collisions.

The objective of this work is to present ongoing research surrounding the development of a theoretical framework aimed at evaluating the challenge presented by 2D games. The framework is currently restricted to games that proposes two game bricks “move” and “avoid”, forming the “driver” metabrick Djaouti et al. (2008). In plain words, this means the focus is put on games where the player controls an avatar and must avoid certain elements, may they be obstacles or enemies. Some examples of games fitting that gameplay categorisation would

be *Super Mario Bros.*[3] or shooters like *R-Type*[4] and the *Gradius*[5] series. Some more recent examples are *Super Meat Boy*[6] and *Celeste*.[7]

The work is organized as follows. First, chapter 1 presents certain notions and definitions that are required for understanding the proposition in later chapters. chapter 2 provides academic background and surveys previous works in the field. The mathematical model of difficulty evaluation is depicted in chapter 3, along with two challenge evaluation metrics. chapter 4 gives the implementation of our framework adapted for the game *Super Mario Bros*. The methodology and results are presented in chapter 5. chapter 6 revolves around a discussion of the results and of the overall research process. This section will also cover limitation of the framework in its current state along with low resolution solutions. This master's thesis is concluded with a summary of this research and possible avenues for further research.

[3] `https://mario.nintendo.com/history`
[4] `https://www.nintendo.com/fr-ca/store/products/r-type-dimensions-ex-switch/`
[5] `https://www.nintendo.com/fr-ca/store/products/arcade-archives-gradius-switch/`
[6] `http://supermeatboy.com/`
[7] `http://www.celestegame.com/`

CHAPTER 1

PREREQUISITES AND DEFINITIONS

For the remainder of this work, certain notions and definitions are necessary to understand the problematic. These notions will be explained in this section.

1.1 VIDEO GAME

The concept of video game has no one true and clear definition, but there is one thing that is evident: a video game is a game. The definition of video game can thus be reduced to a modified definition of game. A *game* is defined by Caillois (1958) as an activity that provides entertainment, is defined in time and place, has no foreseeable outcome, has no meaningful real life consequences, is bound by rules, and is distinct from reality.

In another definition, provided by Crawford (2003), a *game* is a set of dichotomies. Creative expression in pursuit of beauty is art, and is entertainment in the pursuit of money. Non-interactive entertainment is movies, plays, books while interactive entertainment is a plaything. A plaything with no goal is a toy, while a plaything with a goal is a challenge. A challenge without competitors is a puzzle, while a challenge with competitors is a conflict. Finally, a conflict that doesn't allow the competitors to attack each-other is a competition, while one that does is a game Crawford (2003). Although conceptually rich, these definitions are difficult to adapt to a mathematical analysis as they describe the action of play more than the object of

the game. For this purpose, the definition proposed by Chauvier (2007) will be adopted in this research. A *game* is defined as a tuple of six elements:

- **rules** bound the actions the player can take and define a set of game states the game can be in;
- **game states** are any configuration of the game elements, some unique states are the *game over* and the *success* states;
- **actions** are tools to navigate from one game state to another;
- **the player** is the actor responsible of the actions of playing;
- **objectives** motivate and guide the player through the game, give it purpose;
- **challenges** oppose the player in its pursuit of objectives creating non-trivial tasks.

1.2 LEVEL

Historically, video game levels were the solution to limited hardware memory and processing power, allowing the hardware to deal with limited, isolated portion of the game. A level would simply be the most organic partition of a game, each level following a loading-screen. A definition of a level could hence be summarized as a sub-part of a game where the player can progress, wedged in-between loading-screens. With the advances in technology, the need to separate the game in distinct levels has diminished over time and the traditional definition became obsolete. We generalize the previous definition to a space available to the player in order to accomplish its objectives. The downside of this new definition is that the bounds of a level are now blurred. For example, in open world games in the likes of *Cyberpunk 2077*[1], it is not clear what spatial delimitation would compose a level since the argument could be

[1] `https://www.cyberpunk.net/us/en/`

Figure 1.1: First level of *Super Mario Bros*.

made that the entire playable area composes a single level due to the lack of distinct physical barriers, rendering the partitioning meaningless.

In the case of *Super Mario Bros.*, a level is defined as a navigable space bound to the left by the starting position of the player, and to the right by either an end-of-level flag pole or an encounter with a boss enemy. Each level is designed to be traversed from left to right where some backtracking is possible, in order to retry a failed jump for example. Furthermore, since the approach presented in this work focuses on motor difficulty, that is to say the difficulty related to moving and avoiding obstacles, the only relevant information about a level is its topology. The topology of a level is the collection of platforms detailing the area of the level that can be accessed by the player as well as any obstacle present in this accessible area.

In Figure 1.1, the level is bound by the starting point of Mario to the left, and then end-of-level flag to the right.[2]

1.3 DIFFICULTY

The term "difficulty" in the broad sense refers to the amount of effort a player has to exert to overcome an obstacle. A game can present a varied array of obstacles, and different obstacles may require a different approach, which implies that there are multiple types of difficulty as well Chauvier (2007). For example, obstacles may require different combinations of dexterity, perception, analysis, or luck. This leads to the following classification of difficulty, as presented in Levieux (2011), Bopp et al. (2018), and Denisova et al. (2020):

[2]All images of the video game *Super Mario Bros.* are taken from `https://ian-albert.com/games/super_mario_bros_maps/`.

1.3.1 SENSORY DIFFICULTY

Sensory difficulty describes the effort the player has to exert in order to obtain information about the state of the game. There are generally two senses employed while playing video games: vision and hearing. The sense of touch can sometimes also be used in some virtual reality games, but remain a much rarer occurrence. We can imagine sensory difficulty to be an axis, where one end is the complete unavailability of information to the player, and the other is the complete availability of information. Ideally, the sensory difficulty of a game should be somewhere relatively centered on that axis, as either ends are undesirable for the majority of games. An absolute absence of information is a black screen with no audio, while having all the information is similar to directly reading the source code. Sensory difficulty, as opposed to the other types of difficulty, is not a simple scale of "easy" on the left to "hard" on the right. A fairer assessment would define both extremes as "hard" with a center space as "easy". This implies that there are two ways to increase sensory difficulty: hide information from the player and overwhelm the player with information. In practice, this can be achieved through visual and audio cues. For example, information can be hidden by reducing the visibility of the player. The player may be placed in a tight, corridor-like environment where the player can only see as far as the upcoming corner, or in a dark cave with little to no light. In opposition, an overabundance of information can be created through environmental sounds, like the rustling of leaves or the signing of birds to partially cover the sounds of approaching enemies. An example application of such techniques can be found in the dungeons of *Elden Ring*[3], where the environment is dark and labyrinth-like. This specific application allows the game to hide enemies and traps in full view. Figure 1.2 depicts a tunnel where enemies are hidden around corners, away from the player's limited field of view.

[3] `https://en.bandainamcoent.eu/elden-ring/elden-ring`

Figure 1.2: Screenshot taken from *Elden Ring*.

1.3.2 LOGICAL DIFFICULTY

Logical difficulty describe the effort required by the player in order to effectively use the available information. This type of difficulty can most often be found in strategy based games, such as the *Sid Meier's Civilization*[4] and the *Age of Empire*[5] series where the player has to manage and employ available information to defeat their opponents. Since the information is at the core of the logical difficulty, increases and decreases in difficulty can be implemented through modifying the accessibility of the information. Similarly to the sensory difficulty, the logical difficulty is an axis where 'easy' is centered on the axis instead of one of its extremes. In practice, this means that too little information will require the player to extrapolate the missing information, or parse the available information to extract the most impactful information in case of an overabundance of information.

One other way logical difficulty manifests itself is in a player-versus-player environment. The

[4] `https://civilization.com/`
[5] `https://www.ageofempires.com/`

Figure 1.3: Screenshot taken from *Sid Meier's Civilization VI*.

act of facing another player can create logical difficulty where there was none in a single player version. The difficulty created by player behaviour can take many forms like positioning, timing, deception and even provocation. In other words, players can use information about the game to amplify or reduce other types of difficulty to their advantage. For example, *Call of Duty*[6] does not provide much logical difficulty in the single player mode of the game. However, in the player-versus-player mode, entire strategies come to life and strategies are even more explicit in team-versus-team modes where every player assumes a specific role and occupy strategical interest points.

Figure 1.3 demonstrates the abundance of information available to the player in the game *Sid Meier's Civilization VI*.[7] In this example, the quantity of information can be overwhelming and thus present a higher difficulty than if the amount of information was limited to only the information that would be immediately valuable.

[6] `https://www.callofduty.com/ca/en/`
[7] `https://civilization.com/`

1.3.3 MOTOR DIFFICULTY

Motor difficulty describes the level of spatial and temporal precision that the player must demonstrate while performing an action. Motor difficulty was the focus of the first video games like *Pong*[8] and *Space Invaders*[9]. The simplest representation of this type of difficulty is through the avoidance of static obstacles. Famous examples of such obstacles are bottomless pits and spikes, commonly found in games like *Super Mario Bros.*[10], *Sonic the Hedgehog*[11] and *Super Meat Boy* [12](Figure 1.4). Another way of implementing motor difficulty is by linking it to sensory difficulty in rhythm games, where the player has to execute certain actions following a precise timing.

Figure 1.4: Screenshot taken from *Super Meat Boy*.

Motor difficulty is also versatile as a game can require very precise execution of actions, or be lenient on precision and instead demand a very quick succession of actions. For example, a

[8] https://www.atari.com/
[9] https://www.taito.co.jp/en/
[10] https://mario.nintendo.com/history/
[11] https://www.sega.com/games/sonic-hedgehog
[12] http://www.supermeatboy.com/

first person shooter game like *Call of Duty* requires accuracy from the player during gun fights, while beat-em-up games like *Batman Arkham* will reward the player for pressing almost any combination of buttons in quick succession. Both games offer satisfying combat styles that are unique to their genre by applying the principles of motor difficulty in different ways.

1.3.4 EMOTIONAL DIFFICULTY

Emotional difficulty describes the effort the player must put into certain choices because of emotional attachment. This attachment can be directed at any object within the game that may be subject to consequences of a decision. In most cases, this object will be a character, but it may also be an item, a game mechanic, or a story plot. The difficulty can also be felt when the game as a whole expresses a heavy emotional subject such as depression or loss without having the player fight those emotions.

An example of this type of difficulty can be found in *Detroit: Become Human*.[13] The player is repeatedly faced with choices that affects the story, the world and other characters. Figure 1.5 depicts one of those choices.

1.4 CHALLENGE

Up to this point, the use of the terms "difficulty" and "challenge" have been used in similar contexts, but from this point on, they will be used to express different concepts. The easiest way to put it is that difficulty is the perceived challenge. In counterpart, this means that the challenge of a game refers to its objective level of opposition to the player. An obstacle in the player's way generates a set challenge, and that challenge can have a varying impact on the player's experience based on the player itself. We call that impact the difficulty. The difficulty is always based on the challenge as per our previous definition of challenge. This research

[13] https://www.quanticdream.com/en/detroit-become-human

Figure 1.5: Screenshot taken from *Detroit: Become Human*.

focuses on evaluating the challenge and not the difficulty since the idea is to cut out the human factor (and its inherent subjectivity) of the process.

CHAPTER 2

STATE OF THE ART

2.1 RESEARCH QUESTION

In order to situate our research among the current scientific literature, an overview of similar research of significance is presented in this chapter. The literature was explored in an attempt to find information pertaining to our research question: *How can one calculate motor difficulty of a game level without player involvement?* This question can be divided into three important elements, each bringing specificity to the potentially relevant research landscape.

The first element is the calculation of difficulty. The main goal of this research is to provide a methodology to identify the difficulty of a level based on elements found in a given level. This implies that the mean to achieve an evaluation of the difficulty needs to be structured by set rules and instructions so that it can be reproduced with high fidelity in varied applications. The goal behind the calculation as well as its use cases are, in this case, not seen as a factor impacting the relevance of related research.

Two important research fields touching game difficulty evaluation are procedural content generation (PCG) and dynamic difficulty adjustment (DDA). In both cases, it can be interesting to estimate the difficulty of a level for similar purposes. PCG algorithms can use the evaluated difficulty of a newly generated level and compare it to a threshold or range such that the level

presents a desired challenge to the player. DDA can use the evaluated difficulty of a level before and after a given modification to decide whether to keep or discard the modification. Research done in these fields thus might provide an attempt at evaluating game difficulty.

The second element of importance is the distinction of motor, or dexterity-based difficulty from other types of difficulty. As mentioned in chapter 1, each type of difficulty has its own applications and influences. It therefore seems reasonable to assume that each type of difficulty needs an evaluation approach specifically tailored to that type. Since this research focuses solely on motor difficulty, mentions of other types of difficulty is ignored in the related research.

The third and last element of the research question is the non-involvement of players. The main differentiating factor of this research is the idea that difficulty can be evaluated objectively, without players. This means that the traditional process of play-testing does not fit the description of this element. That is not to say that the player needs to be completely removed from the equation. The difficulty metric can be based on player data and statistics, simply not a direct evaluation by the players. That being said, the less present the player, the more accurate and objective the resulting evaluation will be.

2.2 SEARCH STRATEGY

We obtained relevant studies by searching Google Scholar using specific keywords based on the research question. The keywords were picked to be general enough to limit the research set to the field of difficulty analysis of games. We then included additional keywords related to the first element of the research question, limiting further the research set to difficulty evaluation. The resulting keywords are: *game*, *difficulty*, *calculation*, *computation* and *evaluation*. Some variations of these words were used, such as *calculations* and *calculating*.

The specificity of motor difficulty and the involvement of the player are not taken into account

at this point into the literature survey. They will constitute a later stage of manual curation. The reason for this decision is the possibility that the differentiation of difficulty types might not be explicitly made in the research, and searching for it specifically might invalidate otherwise relevant research. As for the third element, the involvement of the player in the evaluation is not something that can easily be quantified using automated keywords. The involvement of players naturally sits on a continuous axis, while keywords are discrete by definition. Keywords are criteria; the subject either fulfills the criteria or not.

2.3 INCLUSION CRITERIA

Inclusion criteria were used to select relevant studies. The studies must be (I1) peer-reviewed publications, (I2) in French or English, (I3) published between January 2004 and June 2022, and (I4) related to the research question keywords. The publications that fulfill the inclusion criteria can be found in Table 2.1. The table also provides a short explanation as to why certain articles wont be reviewed based on the second and third elements of the research question.

2.4 ANALYSIS CRITERIA

The research that was found relevant through the previous criteria is detailed following the evaluation criteria. These criteria serve as tools to identify and highlight important information in relation to this work as well as help denote similarities and differences between the referenced work as well as our proposition.

The first analysis criterion is the level of abstraction at which the level of difficulty is evaluated. The abstraction levels refer to the composition of the evaluated subject. The highest level of abstraction is the entirety of the game while the lowest is the smallest individual element of the game, such as pixels. The level of abstraction of the evaluation impacts its usage as well as its complexity. Moreover, evaluations of different levels of abstraction might be complementary

Reference	Reviewed?	Explanation for exclusion
Desurvire et al. (2004)	No	The study presents heuristics for evaluating playability. Although a few do touch on difficulty, they provide a qualitative evaluation but no quantitative evaluation.
Mourato et Santos (2010)	Yes	The research will be reviewed.
Aponte et al. (2011a)	No	The study proposes a methodology to calculate the difficulty of a game using a statistical approach. The probability used are however obtained from logs generated through playtesting.
Aponte et al. (2011b)	No	The study focuses on evaluating difficulty progression as the player's abilities develop through the game.This research thus heavily rely on information about the player.
Fraser et al. (2013)	No	The study identifies factors which impact game difficulty. These factors can be used by developers to inform their decisions and provide a sort of qualitative evaluation of difficulty.
Mourato et al. (2014)	Yes	The research will be reviewed.
Fraser et al. (2014)	No	A follow-up study on Fraser et al. (2013), provides a measure of the impact of the identified factors. This measure is however not directly tied to the difficulty, but more so to the player's performance.

Table 2.1: Relevant research found.

and weave a more complete evaluation while others might be conflicting.

The second analysis criterion is how removed is the player from the difficulty evaluation. The reviewed research have already distanced themselves from playtesting as per the research question's third element. This criterion is to identify the type of player data used as well as how it is used. As the point of this research is to completely get rid of the player in the evaluation process, the further removed the player is the better. For example, a player's recorded tendencies in face of an obstacle are favourable compared to a player's general opinion of said obstacle. In other words, we aim to subtract the subjectivity tied to the player out of the difficulty evaluation equation.

2.5 RESEARCH REVIEW

Both pieces of research found to be relevant by the aforementioned research question and research methodology are part of a series of four articles Mourato et Santos (2010); Mourato et al. (2012, 2013, 2014) part of a doctoral research on procedural content generation. The second and third publications of the series focus heavily on conceptual analysis of PCG and provides little information on difficulty, much less difficulty evaluation. The other two articles (first and fourth) of the series are however very similar to our research both in concept and implementation. The latter part of this chapter goes over both relevant articles and emphasize the differences with our research.

2.5.1 MEASURING DIFFICULTY IN PLATFORM VIDEOGAMES

The first article of the series Mourato et Santos (2010) provides a way to measure the difficulty of a game level by using probabilistic methods. Each obstacle of a level is considered in isolation from the rest of the level so that difficulty is measured individually. The difficulty of an obstacle is stated as the probability that a player will not succeed to traverse the obstacle. That probability is calculated based on three possible outcomes of an obstacle: the player successfully traverses the obstacle, the player unsuccessfully traverses the obstacle and tries again, and the player unsuccessfully traverses the obstacle and gives up. These calculations are done for all static obstacles of a level, and a slightly modified version to approximate dynamic ones.

The article also goes over five situations in which static obstacles can find themselves in, each affecting difficulty in its own way. A general formula for difficulty calculation of static obstacles is proposed on the assumption that all static obstacles can be summarized to a jumping or gap obstacle, that is, that the obstacle can be traversed by jumping over it. It then presents the difficulty calculations applied to levels of *Super Mario Bros.* as well as *Little*

Big Planet. No control results of playtesting were presented to compare with the calculated difficulties, but the results were nonetheless adhering to the expected pattern of progressive difficulty increase across levels.

The first important aspect of the article is the use of probabilities to measure the difficulty. Using probabilities as the outcome of the difficulty evaluation allows for a very intuitive assimilation of the results, as they have intrinsic meaning. For example, a difficulty rating of 0.3 suggests that 30% of the players attempting an obstacle would fail to traverse the obstacle. This characteristic is useful when the results of the evaluation are fed into another piece of software such as a PCG system or DDA system. The intrinsic meaning of probabilities means that those systems can be developed without taking the context of the difficulty evaluation into account. For example, if a game level's difficulty is expressed as relative to other levels, then a PCG or DDA system would need additional information in order to make use of that difficulty evaluation.

The second notion of importance in the article is the individuality of each obstacle. This implies that a game level is conceptualized and treated as a series of obstacles. This notion allows for each obstacle to be evaluated independently from one another and then chained together in series. An example of this is briefly mentioned in Mourato et al. (2012). Consider all paths through a level, these paths branch and rejoin at one point or another so that there is a unique start and end to the level. Each path is populated with zero or more obstacles. In this scenario, each path's difficulty can be calculated from its series of obstacles. It is also possible to avoid redundant work when computing the difficulty of an obstacle that appears multiple times in the level. The calculation can occur once and the result reused as needed.

A significant drawback of this approach is its limitation when it comes to obstacles in close proximity. If two obstacles are close enough to one another, they might start interacting with each other in terms of difficulty. For example, if two jumps are required in quick succession in order to overcome two obstacles, the overall difficulty of the segment would reasonably be at

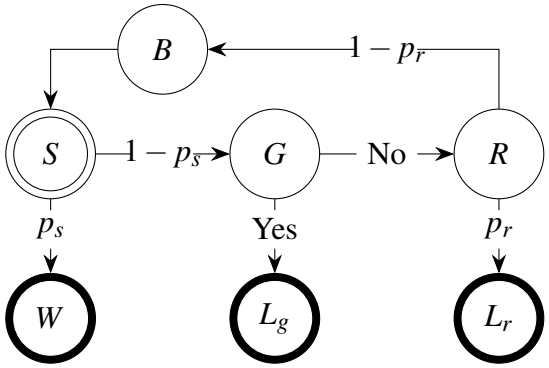


Figure 2.1: State chart of an obstacle as presented in Mourato et Santos (2010)

least as difficult as the sum of individual difficulties. Remark that this is only a hypothesis and it is possible that interactions between obstacles generate no noticeable impact on overall difficulty. The proposed approach nonetheless does not infirm or mention this possibility.

The third important point presented in the article is the difficulty calculation. Each obstacle is defined to accept one of three possible outcomes. The first outcome is the successful traversal of the obstacle by the player. The second outcome is the failed traversal of the obstacle by the player, leading to an immediate game-over state. The third possible outcome is the failed traversal of the obstacle by the player without leading to an immediate game-over state. This final outcome allows the player to reattempt the obstacle or not. The relationship between these outcomes is explained in Figure 2.1.

The states of Figure 2.1 are as follows: S is the starting point of the obstacle; G represents whether the obstacle automatically leads to a game-over; R represents the retry point of the obstacle; B is a series of other obstacles that might be required to traverse in order to return at the starting point of the obstacle. The states W, L_g, and L_r represent the three possible outcomes described previously. The probability of successfully traversing the obstacle on a single attempt p_s and the probability of renunciation p_r are used to traverse the graph. The difficulty calculation equation is extracted from this graph as the probability X:

$$X = p_s \sum_{i=0} [(1-p_s)(1-p_r)]^i, \tag{2.1}$$

where i represents the number of attempts for the obstacle. Remark that this equation does not take into account the possibility that more obstacles need to be traversed to return to the obstacle's starting point. This equation also does not directly reflect whether an obstacle automatically leads to a game-over state. This case can however be considered by setting the renunciation probability (p_s) to 1, mimicking the game-over scenario by preventing the player from retrying the obstacle. All other edge cases such as the empty level and the impossible level can be considered by setting probability values of 0 or 1 to specific variables of the equation.

The main problem of this approach is that information on the player is required in the form of the renunciation probability in a context where acquiring information of the player is near impossible. Whether the application is PCG or DDA, probing player behaviour in order to obtain something resembling an accurate evaluation of the player's renunciation probability would take a non-trivial set of data. Such a set of data can take time to put together even through automatic means, making the difficulty calculations unreliable in early levels of a game. Additionally, the probability of renunciation seems to be tied to the player as an individual, and not as a group.

Other than the renunciation probability, Equation 2.1 uses the probability of successfully

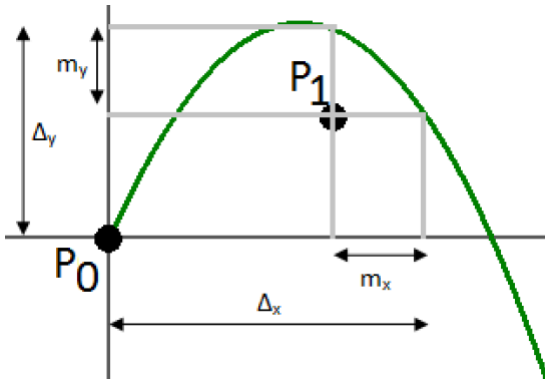


Figure 2.2: Visual reference of error margin as presented in Mourato et Santos (2010)

traversing an obstacle in one attempt. This probability can be inferred from the geometry of obstacle in focus. This is done by calculating the error margin of the jump necessary to overcome the obstacle. The error margin is defined as the acceptable deviation from the theoretical perfect jump and still succeed in traversing the obstacle. An obstacle with a high error margin suggests the jump has a relatively loose timing, while a low error margin suggests a jump needs to be be near "pixel-perfect" to overcome the obstacle.

Figure 2.3 presents a visual reference of the error margin. This example uses the projectile motion equation $P = P_0 + v_i t + \frac{gt^2}{2}$ to describe the path of a jump. P_0 and P_1 represent the origin platform's edge and the target platform's edge respectively. m_x and m_y thus represent the error margin both along the horizontal and vertical axis. If either of those error margin values are negative, the obstacle is impossible to traverse by virtue of the target platform being farther from the origin platform than the distance traversed by the farthest reaching jump. Remark that in order to keep the values in the range of probabilities, the error margin M of the obstacle is expressed as a ratio of the total distance traversed in both directions.

$$M = \frac{m_x \cdot m_y}{\Delta x \cdot \Delta y}$$

One drawback of this calculation method is that both the origin platform and target platform extend to infinity on either side of the obstacle. In other words, a large value of m_x does not

take into account that the jump could overtake the target platform entirely. For example, if two obstacles are placed in quick succession separated by a small platform, a far reaching jump from the first obstacle could land the player farther than the separating platform and into the second obstacle.

2.5.2 DIFFICULTY IN ACTION BASED CHALLENGES: SUCCESS PREDICTION, PLAYER'S STRATEGIES AND PROFILING

The second reviewed article Mourato et al. (2014) is a continuation and follow-up research of the previous reviewed Mourato et Santos (2010). The research is presented in two facets: the difficulty evaluation, and the player profiling. Although interesting, the second part of the research is left out of this review as it falls outside the scope of our research. The difficulty evaluation portion proposes an improvement on calculation of difficulty for static obstacles, as well as extends its application to dynamic obstacles. Static-dynamic hybrid obstacles are also discussed and their difficulty approximated to a combination of both types' individual difficulty. In summary, the study proposes a statistical model for difficulty evaluation based on player tendencies.

The new approach to difficulty evaluation of static obstacles relies on the positioning of the jump a player might take in order to overcome the obstacle. This notion was not taken into consideration previously, as the edge of the origin platform was chosen as the starting point of the jump. The idea behind this new perspective is that most players won't wait until the last possible moment to jump; most will attempt to jump before the edge. It can be expected to find some distribution centred on a distance d from the edge of the origin platform describing when players tend to jump. Using the theoretically perfect jump, it is possible to determine the earliest jump resulting in a successful traversal of the obstacle. The distribution can thus be bounded by the earliest possible jump resulting in success, and the edge of the platform. The probability of failure of an obstacle is then the portion of the distribution outside of those

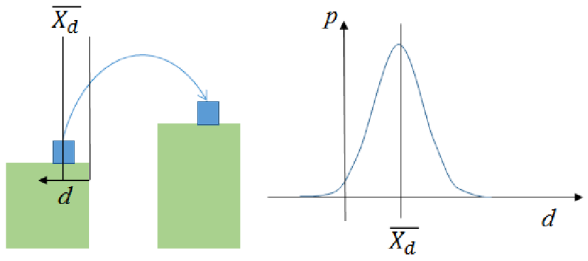


Figure 2.3: Expected distribution of static obstacle jump position as presented in Mourato et al. (2014)

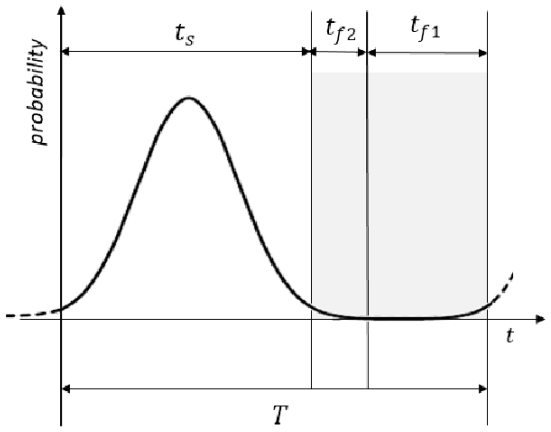


Figure 2.4: Expected distribution of dynamic obstacle as presented in Mourato et al. (2014)

bounds.

The issue with using a probability distribution in this way is that the distribution needs to be known. Unless a distribution can be proven to describe the general behaviour of players in the context of the obstacle, data needs to be gathered from playtest sessions in order to build the distribution. Even if a general distribution can be used, for example a normal distribution, the mean and variance of such distribution would still require data collection.

Dynamic or time-based obstacles difficulty is calculated in a similar fashion as the static obstacle: a distribution is assumed spanning over a fixed time interval of length T such that this interval represents one cycle of the obstacle. For example, a carnivorous plant from *Super Mario Bros.* would be evaluated starting from the moment when the plant just disappeared into a pipe, through the plant coming back out and ending when the plant disappears again. This length of time T is divided into three segments. The first is the span of time during which the

obstacle poses no danger to the player and is denoted t_s. The second is the span of time during which the obstacle poses immediate danger to the player, possibly leading to a game-over state and is denoted t_{f1}. Lastly, the third is the time required by the player to traverse the obstacle and is denoted t_{f2}. The traversal time is considered an extension of the dangerous interval as the player would end up in the dangerous time before completing the obstacle. These times are represented in Figure 2.4.

The distribution is centered somewhere in the safe time window. In the case of dynamic obstacles, the bounds are defined as the beginning and end of the safe time window. Considering the resemblance in evaluation between the dynamic obstacle and static one, their drawback is the same: the need for experimental values.

The hybrid obstacle consists of a combination of static and dynamic elements. For example, a moving platform in *Super Mario Bros.* has a jumping component like any other gap, and a timing component from the movement of the platform. The calculation of the hybrid obstacle's difficulty is a product of its static and dynamic individual difficulties. The results presented in the article support this hypothesis and imply that every obstacle can be evaluated only through its static difficulty and dynamic difficulty, individually or together. Remark that these calculations are made with probabilities, and so the resulting difficulty of a hybrid obstacle is always equal or greater than both individual difficulties.

The results presented in the article indicate that the approach presented generates promising difficulty approximations. The hypothesized distribution used in both static and dynamic calculations is however disproved to be a normal distribution by the Kolmogorov-Smirnov test. It is however observed that a normal distribution can in fact be used to represent each player's individual performance, and that the standard deviation is smaller for players who perform better on average. Apart for failing to abstract away from player data completely, the research proposes an interesting and promising approach to difficulty calculations, and may resolve its shortcomings through further research.

CHAPTER 3

MATHEMATICAL MODEL

This chapter presents the mathematical model for difficulty evaluation. The chapter offers a formal mathematical definition of a bi-dimensional video game level, describe the analysis process of a level, and define two metrics used for assessing the difficulty of levels.

It is important to note that our model currently relies on a static environment, that is with no moving elements. A level is thus, in this case, considered a "snapshot" or a still frame of a dynamic level. Also, only the collision geometry of a level is used in the analysis. In other words, only elements producing collision data are considered part of a level, and all other elements are omitted entirely. As a general rule, all visual and audio elements are not part of the level. See Ericson (2005) for a thorough overview of collision detection.

Regardless of the method used to retrieve the collision geometry of a level, the level can be expressed as a bi-partition of pixels: every pixel of a level is either part of a collision structure, or not. A level is thus expressed as the union of both types of pixels. Formally, we define C as the set of all pixels that are part of a collision structure, and C' as the set of pixels that are not part of a collision structure. Another way to describe the set of non-collision pixels is as the entirety of the level L, minus the set of collision pixels, or $C' = L \setminus C$. This representation of a level has the advantage of providing powerful mathematical tools to study the problem. An example level is depicted using this definition in Figure 3.1.

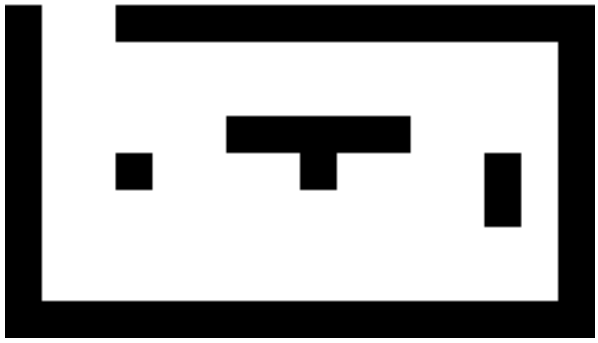


Figure 3.1: Example level, collision set C coloured in black and non-collision set C' coloured in white.

Horizontal velocity	30 p/s
Vertical Velocity	10 p/s
Gravity	-5 p/s^2
$\vdots$	$\vdots$

Table 3.1: Example subset of A for *Super Mario Bros*.

Apart from the level itself, the model accepts a set of properties associated with the player's avatar denoted A. The pawn is the player's manifestation within the game. This set can contain any gameplay element that might affect difficulty in one way of another. For instance, A could contain the pawn's starting position, movement speed, jump height, ability to double jump and so on. this is information necessary to calculate the difficulty of a level and cannot be acquired through the level itself. In the eventuality of a analysis of a dynamic level, additional sets of properties could be introduced for each dynamic element.

In the case of *Super Mario Bros.* the set contains the player's starting position, the ability to move horizontally on platforms, the ability to jump when on a platform, the maximal running speed, the running acceleration, the maximal jumping velocity and the gravitational constant as shown in Table 3.1.

The first step of the analysis is to define the area within the level the pawn can reach. This is done to identify the game elements that may have an impact on difficulty. The reasoning

behind this step is that an obstacle situated outside the reachable area does not impact the motor difficulty. For example, a static obstacle such as spikes does not impact the difficulty evaluation if they cannot be reached by the player. In this case, they may be use as decoration or for intimidation, but not as a motor difficulty influence. The reachable area is thus used to emphasize the important game elements.

From the properties of the pawn A, it is possible to find a function $R : L \longrightarrow [0,1]$ defined for any pixel $p \in L$ such that $R(p)$ returns the reachability of p. The reachability value of a pixel describes the ease with which the pawn can occupy the space of that pixel. The reachability value of 0 represent a pixel that cannot be occupied by the pawn under any circumstance, while the reachability value of 1 represent a pixel that can be reached with little to no effort. The vagueness in the definition of the upper bound permits to tailor the reachability function more easily to the specific case of study.

The second step of the analysis is to define the area within the level that can lead the player into a game-over state. This area represents the danger associated with various game elements. Similarly to the reachable area, it is possible to define a function $D : L \longrightarrow [0,1]$ from A for any pixel $p \in L$ such that $D(p)$ returns the danger of p. The danger value of a pixel represents the likeliness that the player will end up in a game-over state from that pixel. The danger value of 0 indicates that the pixel cannot lead to a game-over state, while the danger value of 1 indicates that the pixel is itself a game-over state. The danger value can, but not necessarily, be expressed in terms of probability of failure.

It is important to note that the danger function should not take into account the accessibility of an obstacle to determine it's danger impact. In other words, an obstacle must not produce a more meaningful danger area by virtue of being encountered by the player. This is covered by the interaction of both reachable and danger areas. We thus define the effective danger $D'(p)$ as the danger of given pixel modified by it's accessibility. Since both the danger $D(p)$ and reachability $R(p)$ increases the effective danger and that the effective danger must not exceed

either danger or reachability alone, we express the effective danger as a multiplication of both parts, $D'(p) = D(p) \cdot R(p)$. The effective danger is guaranteed to be in the interval $[0,1]$ as both parts of the multiplication are also in the same interval. Remark that both the reachability and danger functions must be defined on a case by case basis, as they heavily rely on the set of player properties A.

Both functions can also be deconstructed into multiple sub-functions to allow for specific tuning. For example, an obstacle may behave in a unique way and require its own danger sub-function. Although limited in use when only static elements are considered, this property allows for an easier expansion of the framework to dynamic elements. In the case of dynamic elements such as enemies, an enemy with a cyclical action sequence akin to a fixed patrol path would likely generate a different danger zone than an enemy possessing a developed artificial intelligence.

In summary, the framework takes in as input data the topology of the level as a combination of collision and non-collision pixels, as well as the set of properties of the player's pawn such as movement speed and jumping velocity. From these properties, the reachability function and the danger functions are deduced. The framework ultimately allows the definition of metrics used to generate difficulty estimations for the level in its entirety.

3.1 DIFFICULTY EVALUATION METRICS

Our modelling of a level along with its associated reachable and danger values provide a formal framework allowing one to define various metrics for the motor difficulty of 2D video games. We now suggest two possible metrics, but this list is no way exhaustive.

3.1.1 AREA-BASED METRIC

The first metric expresses the difficulty of a level as the ratio between the effective danger area and the reachable area. This ratio represents the amount of traversable space covered by danger. The intuition behind this metric is that a level composed of wide-area obstacle should propose a higher level of difficulty than a level with similar, but smaller obstacles. For example, in *Super Mario Bros.*, a gap of width five is more difficult to traverse than a gap of width two, since the wider gap requires more precision and timing than the narrower one. The area-based metric M_A is describes as:

$$M_A(L) = \frac{\sum_{p \in L} D'(p)}{\sum_{p \in L} R(p)} \tag{3.1}$$

where L is the level, $D'(p)$ is the effective danger of the pixel p and $R(p)$ is the reachability of the pixel p. It follows from the definitions of the effective danger and reachability that $D'(p) \leq R(p)$, and since both the numerator and denominator iterate over the set L in identical manners, the result of Equation 3.1 must be between 0 and 1 for any non-empty level. For example, the difficulty of any level entirely devoid of danger areas is always 0 while the difficulty of the empty level, or any level with no reachable area, is not defined. Figure 3.2 provides a visual representation of this metric.

3.1.2 PERIMETER-BASED METRIC

Computing the total area of danger and reachable zones represents, in all but trivial cases, an important time sink. Instead, one could measure the total length of the frontier separating danger and reachable zones in order to get a perimeter-based metric. Comparatively to how the area-based metric explores the idea of difficulty as overlapping areas of reachability and danger, the perimeter-based metric expresses difficulty as the transition from areas of reachability to

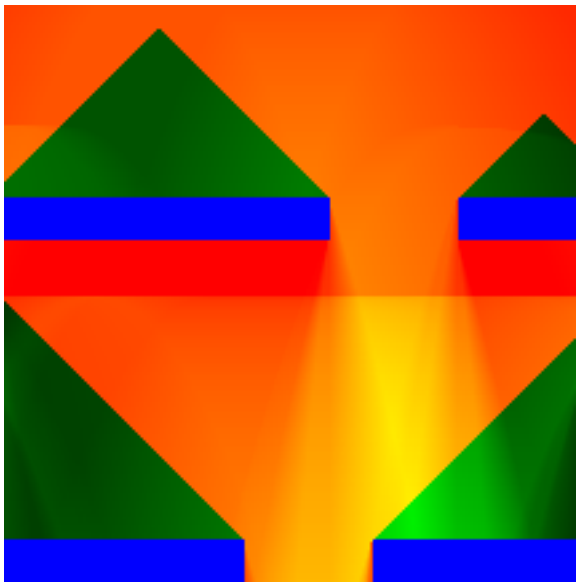


Figure 3.2: Visual representation of the area-based metric. Collision pixels are in blue, reachable pixels are in a green gradient, danger pixels are in red and effective danger pixels are in a yellow gradient.

0.2	0.1	0.0
0.1	p	0.0
0.0	0.0	0.0

(a) $p \notin B$**, since** $N(p) = 0$**.**

0.4	0.3	0.2
0.3	p	0.1
0.2	0.1	0.0

(b) $p \in B$**, since** $N(p) = 3$ **and** $D'(p) < \varepsilon$**.**

0.6	0.5	0.4
0.5	p	0.3
0.4	0.3	0.2

(c) $p \notin B$**, since** $D'(p) > \varepsilon$**.**

Figure 3.3: Examples of neighbouring pixels for $\varepsilon = 0.25$**.**

areas of danger.

To identify pixels that are part of the border separating reachable pixels from danger pixels, we define a function $N : L \longrightarrow \{0, 1, \ldots, 8\}$ that identifies the number of 8-adjacent danger pixels. A pixel is considered a danger pixel if it has an effective danger value $D'(p)$ of above a constant ε. The value of ε is determined on a case by case basis in relation of the danger function D to isolate trivial danger from important enough danger. Another requirement for a pixel to be part of the border set is that the pixel must not itself be a danger pixel. This is to avoid considering pixels that are completely within a danger area.

We also define all pixels part of the border area as part of the set B. The set is defined as follows: $B = \{p \in L \mid R(p) > 0, D'(p) < \varepsilon, N(p) > 0\}$. That is a pixel must have at least a single neighbouring pixel with an effective danger value above the threshold ε, while itself having an effective danger value below ε and being reachable. Three example cases are visualized in Figure 3.3. The perimeter-based metric M_P is thus expressed as:

$$M_P(L) = \sum_{p \in B} R(p) \tag{3.2}$$

where L is the level and B is the set of pixels composing the border between danger areas and non-danger areas. Compared to the area-base metric, Equation 3.2 is not bounded by the range $[0, 1]$, and returns the value 0 in the case of an empty level, a level entirely devoid of danger, and a level entirely devoid of reachable areas.

One could certainly argue that the two previous metrics are somewhat arbitrary since any difficulty value could theoretically be obtained by carefully choosing an appropriate difficulty metric. However, this flexibility is precisely the point of our model. It allows the level designer to define metrics tailored to the particular rules and constraints of its designed level. Still, the idea of creating comparisons between reachable and danger zones remains at the core of the approach. Figure 3.4 provides a visual representation of this metric.

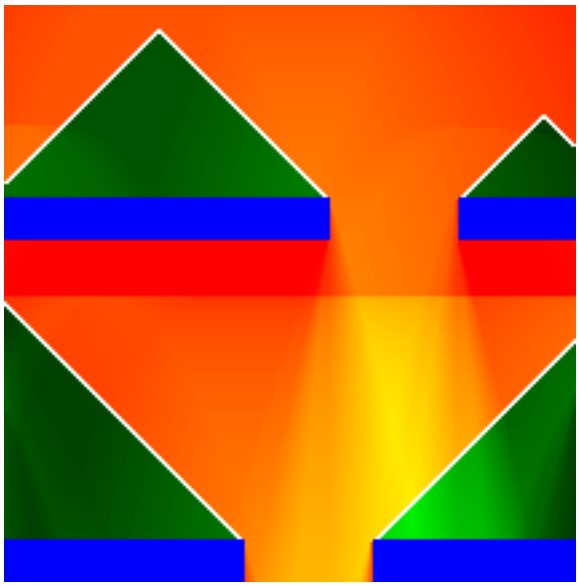


Figure 3.4: Visual representation of the perimeter-based metric. Collision pixels are in blue, reachable pixels are in a green gradient, danger pixels are in red and border pixels are in white.

CHAPTER 4

IMPLEMENTATION

In order to assess the validity of our model, we analyze the difficulty of the industry game *Super Mario Bros*.[1] This game makes for a good reference point as it is globally known, easy to pick up and play, and has a vast amount of openly available resources such as level image and texture tile sets. This accessibility allows for an easier collection of experimental data during the testing phase. This chapter presents our methodological approach to the implementation for this purpose.

For the sake of providing a proof of concept, two simplifications are made to the implementation. The first simplification is that *Mario* has a collision size of a single pixel instead of the actual 16 by 16 pixels. This is done to allow collision checks to be made by coordinate comparison instead of a 2-dimensional method. The second simplification is that we assume that all collision pixels are reachable. This allows us to avoid superfluous calculations. Remark that one significant drawback of this assumption is that a level impossible to complete due to two platforms being too far apart will not return a difficulty $d(L) = 1$.

Our prototype is written in C++ and uses the *openCV*[2] library for all image manipulations. The input is comprised of a still image of the studied level, along with a number of textures for collision objects and the set of pawn properties A. The danger zone requires no input

[1] https://mario.nintendo.com/history/
[2] https://opencv.org/

Figure 4.1: Partial example of level 1-1 of *Super Mario Bros.* as a three colour channel result.

as we define it to be the entire lower bound of the level. This approximates the game-over states reached by falling into gap obstacles. Remark that the implementation uses an image recognition library only as an alternative form of input, as we do not have access to core game information that would normally be available when working in a game engine. The result of the solution is a triple colour channel image having the same size as the original level image, coloured according to collisions (B: blue), reachable area (R_{p0}: green) from the start point of the level p_0 and danger area (D: red). Remark that any pixel of the resulting image can be multicoloured as shown in Figure 4.1.

The first step in the analysis process is to identify the collision zones from the input level image. This is done by employing the *matchTemplate*[3] function with the studied level and a collision texture as parameter. A normalized coefficient is obtained through the function call representing how well the game level matches the specified texture at a given position. A value of 0 means that there is no similarity and a value of 1 means that the texture matches perfectly. We then proceed to accept positions as part of the collision only if the returned coefficient is above a predefined threshold, in our case 0.85. This sequence of operations is repeated for each distinct collision texture.

As for the calculation of the reachable zone, each collision pixel with a free pixel directly above it is considered a platform, that is a pixel on top of which the pawn can be positioned. Figure 4.2 depicts pixels that are considered a platform in an example level. Since we assume all platforms are reachable, we conclude that the reachable space of such pixels is exactly

[3]https://docs.opencv.org/4.x/df/dfb/group__imgproc__object.html#ga586ebfb0a7fb604b35a23d85391329be

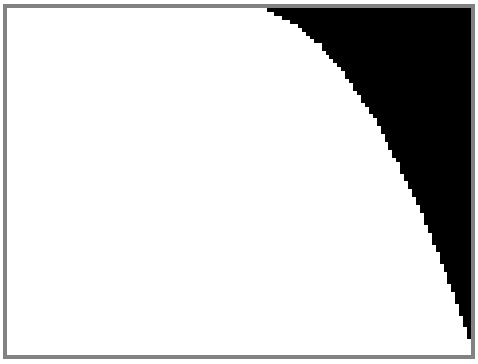
Figure 4.2: Example level, platform pixels are coloured in red.

R_{p0}, that is $R_{p0} = \cup R_p$. This allows us to traverse the level without considering the order of traversal.

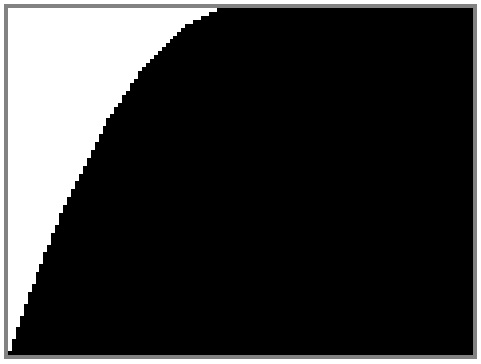
The second step in the process is to take each collision pixel and calculate its reachability. In other words, where can the pawn go starting from a given pixel? Since a pixel can interfere with the accessible zone of another pixel, it becomes impossible to treat each pixel individually. One way to remedy that problem is to simulate all possible movement from all black pixels. We use the following approximation in order to simplify the process.

The jump is separated into two parts as depicted in Figure 4.3. The first part is the ascending motion of the jump. From the player's avatar's properties A, the minimal and maximal paths from the origin of the jump are used as bounds. Any collision pixels that might cause interference with this ascending jump are considered if they are only between those bounds and within the maximum height of the jump. In our case, the minimal path occurs when the

initial horizontal velocity of the jump is 0, and the maximal path is when the initial horizontal velocity is at it's maximum. All collision pixels are assessed with the ascending part of the jump before moving on the second part of the jump.



(a) Ascending part of the jump.



(b) Descending part of the jump.

Figure 4.3: Collision prone areas associated with a left facing jump. Jump origin is the lower right corner.

The second part of the jump is the descending motion following the ascending motion. This motion is different than the ascending motion in two ways other than direction: the motion originates from multiple points, and each point allows a single path from it. The origin point for the descending motion are based on the results of the ascending motions. Any point situated at the top of the ascending motion, whether it be at the maximum jump height or right beneath a collision pixel, is considered a root of descending motion. The velocity at the roots of descending motion are calculated from the velocity changes applied during the ascending motion. Similarly to the ascending motion, the descending motion can be bound with the minimal and maximal paths from the origin of the jump to deal with collision pixels.

Both part of the jump are combined to provide the resulting reach of a given origin pixel. The process is then repeated for every platform pixel of the level. The danger area is then constructed using the following algorithm: the entire bottom row of pixels is considered part of the danger zone. The overall danger zone is constructed by propagating this row upwards. Each danger pixel can generate up to another three pixels. For example, given a propagation angle of 45°, each danger pixel propagates to the pixel above and to the left, directly above

and above and to the right. This example is illustrated in Figure 4.4. If a pixel is a collision pixel, it cannot become a danger pixel.

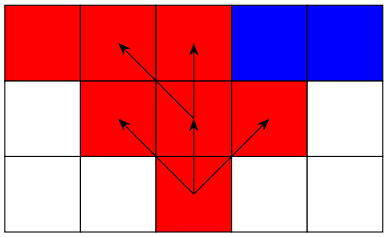


Figure 4.4: Example 45° **danger propagation. Danger pixels are in red and collision pixels are in blue.**

The final result is an image combining all collision, reachable and danger pixels as colour channels. Both metrics are then calculated to the resulting image.

CHAPTER 5

EXPERIMENTATION

Since we do not have access to the source code and assets of *Super Mario Bros.*, the collision and pawn properties could not be directly retrieved. Instead, the collision data was extrapolated from level images with an image processing library, and the pawn's properties were either taken from related works on the game, or estimated. This first step of the experimentation process is not directly linked to our proposed framework, but is, in this case, necessary as the initial conditions of our model could not be met. The collision data and pawn's properties used from here on out may not be a perfect replica of the original game.

The process of identification of collision data relies on finding a set of textures that are associated with collisions. These textures were taken directly from the game images. Figure 5.1 is a complete list of textures used to identify collisions. A program written in C++ with the OpenCV2 library takes in a level image and a set of collision textures and outputs a text file (*.txt*) with the collision data. Each line of the output file contains two points in 2D space describing a collision rectangle. The collision textures were located within the level image using the *matchTemplate* function.

The collision data contained in the output file is further processed in two ways. First, redundant collisions are created as a byproduct of the *Super Mario Bros.* tile-set as show in Figure 5.2. Those redundant collision are parsed out by removing any collision box that does not adhere

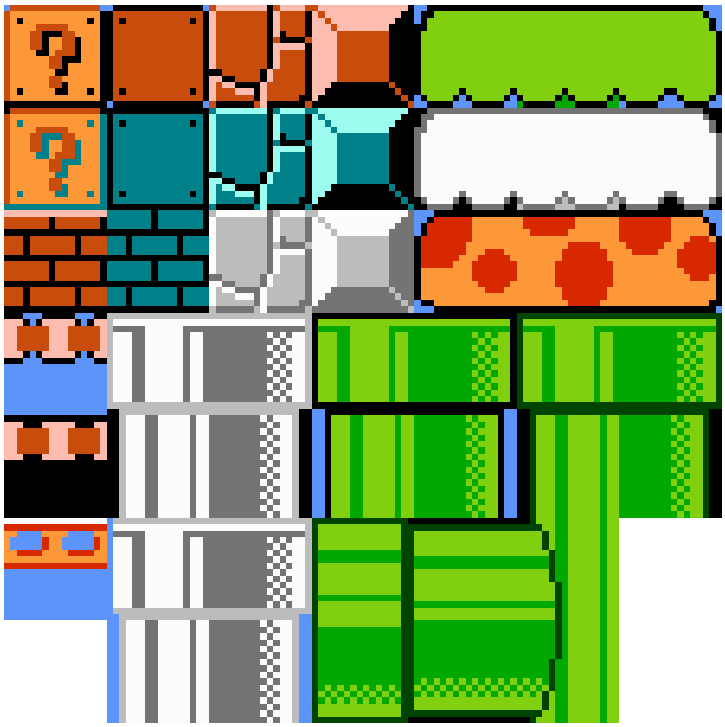


Figure 5.1: Complete set of collision textures, taken directly from test levels.

to the game's grid of 16×16. For example, a collision box situated at position $(160, 8)$ would be removed since 8 is not a multiple of 16. Second, if two collision boxes are horizontally adjacent, they are merged together to create a larger collision box. This effectively reduces the amount of collisions to track during the next steps. Figure 5.3 shows an example of connected collisions.

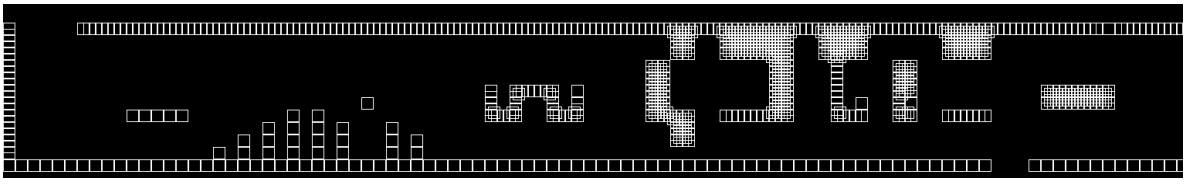


Figure 5.2: First half of level 1-2 of *Super Mario Bros.*, raw wire-frame collisions in white.

The first iteration of the test program was done in a way to assign either 0 or 1 to the reachability value of any pixel. In this test, a pixel is reachable or it is not. No further distinction is made. Similarly, the danger value of any pixel is either 0 or 1, denoting the presence or absence of danger respectively. This first iteration has the purpose of providing a relatively quick test bed for our theoretical model. In exchange for the rapid development of the iteration,

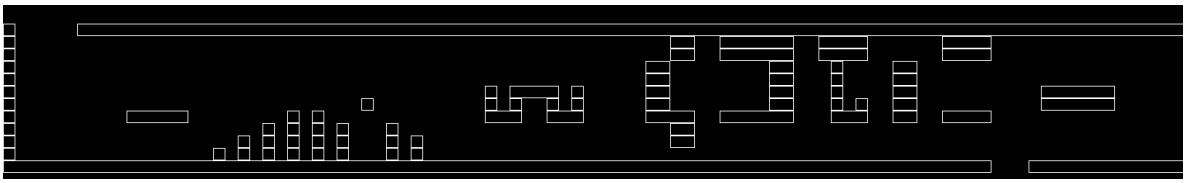


Figure 5.3: First half of level 1-2 or *Super Mario Bros.*, merged wire-frame collisions.

simplifications are made that reduces the precision of the model.

The second iteration of the test program rectifies a good deal of the inaccuracies introduced by the first iteration by evaluating the reachability on a gradient. A reachability value of 0 still signifies the pixel is not reachable, but a reachability value of 1 signifies the pixel is reachable from every platform pixel. Recall that a platform pixel is defined as a collision pixel that has no collision pixel directly above it. For the set of platform pixels P, the reachability function for a given pixel $p \in L$ as described in chapter 3 is the sum of platform pixels from which p can be reached over the total amount of platform pixels. The danger function is the same as the first iteration. A gradient-based danger following similar principles of propagation was briefly tested, but added considerable computation time and was ultimately removed form the tests.

Super Mario Bros. contains eight worlds, and each world has four levels. However, difficulty is not evaluated for all levels. First, the fourth level of each world (X-4) is a boss level where the majority of the difficulty is concentrated in the fight at the end of the level. Since we do not consider dynamic elements such as enemies, the difficulty evaluation of those levels is not representative of their actual difficulty and are thus removed form our analysis. Second, level 2 of world 2 (2-2) and level 2 of world 7 (7-2) are underwater levels. As such, they do not share the same pawn's properties of the rest of the levels. Out of the original thirty-two levels, twenty-two are evaluated.

Table 5.1, Figure 5.4, and Figure 5.5 summarize the results of the experimentation. Columns labelled M_A contain the area-based metric while the columns labelled M_P contain the perimeter-

based metrics. Metrics with subscript 1 are without gradient reachability and metrics with subscript 2 are with gradient reachability. The execution times presented covers the entire process of evaluation excluding the extraction of collision data.

Level	M_{A1}	M_{A2}	M_{P1}	M_{P2}
1-1	0.21	0.17	630	80.31
1-2	0.23	0.28	751	39.92
1-3	0.81	0.76	1093	275.70
2-1	0.26	0.20	649	89.58
2-3	0.51	0.52	1100	235.21
3-1	0.29	0.29	895	117.18
3-2	0.14	0.12	282	56.11
3-3	0.79	0.78	1225	246.53
4-1	0.26	0.22	831	120.84
4-2	0.31	0.44	1097	64.70
4-3	0.86	0.85	1141	248.41
5-1	0.18	0.16	625	74.77
5-2	0.42	0.36	1752	140.50
5-3	0.81	0.76	1093	275.70
6-1	0.54	0.44	1555	224.21
6-2	0.17	0.18	628	81.38
6-3	0.88	0.87	1035	237.96
7-1	0.16	0.14	544	76.01
7-3	0.54	0.50	1014	211.87
8-1	0.33	0.29	1425	122.56
8-2	0.49	0.43	1665	188.14
8-3	0.25	0.20	585	61.52

Table 5.1: Test results of the area-based and perimeter-based metric over levels of *Super Mario Bros*. M_{A1} and M_{B1} are the area-based and perimeter-base metric respectively, without gradient reachability. M_{A2} and M_{B2} are the area-based and perimeter-based metric respectively, with gradient reachability.

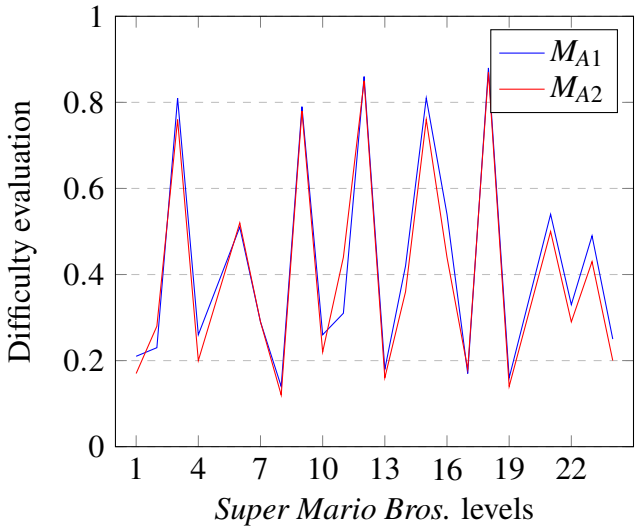


Figure 5.4: Difficulty evaluations of *Super Mario Bros.* levels using area-based metrics.

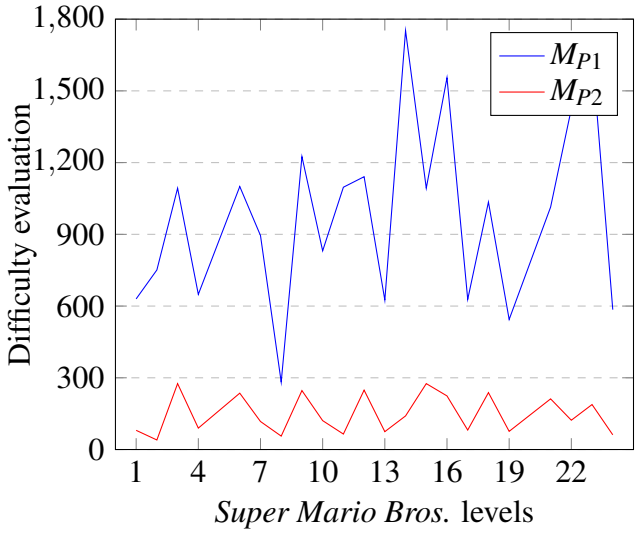


Figure 5.5: Difficulty evaluations of *Super Mario Bros.* levels using perimeter-based metrics.

CHAPTER 6

DISCUSSION

As previously mentioned in chapter 3, our current approach does not take into consideration any dynamic element of a level such as enemies and projectiles. The calculated difficulty is thus exclusively linked to the level topology. In the best case scenario, a level would have no dynamic elements at all and the difficulty evaluation would be theoretically accurate to the overall motor difficulty. In the worst case scenario, a level heavily populated by dynamic elements would have its difficulty evaluation be severely underestimated. In either case, our model establishes a lower bound on the actual difficulty of a level, including dynamic elements.

For example, both area- and perimeter-based metrics put the level 7-1 (Figure 6.2) at a lower difficulty than the level 1-1 (Figure 6.1). From a topological standpoint, the level 7-1 has one fewer gap to traverse than level 1-1, explaining it's lower evaluated difficulty. That being said, level 7-1 has over 25 enemies while the level 1-1 only has about 15. In other words, some levels like the level 1-1 will only be slightly underestimated while other levels like level 7-1 will be disproportionately underestimated.

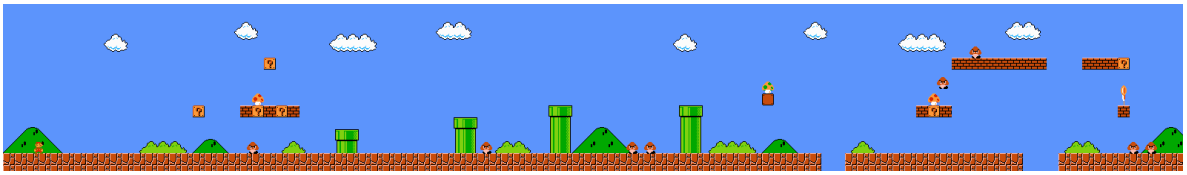


Figure 6.1: First half of level 1-1 of *Super Mario Bro.*

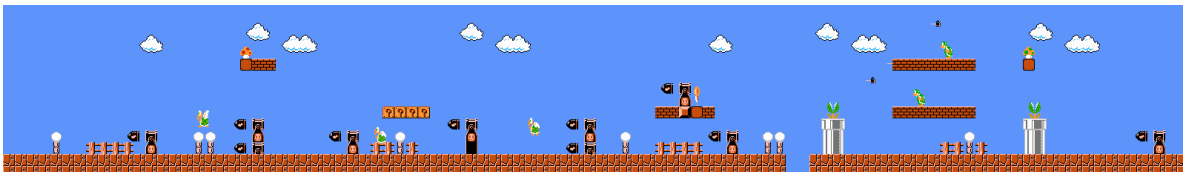


Figure 6.2: First half of level 7-1 of *Super Mario Bro.*

Moreover, it should be noted that our model does not consider elements like game bonuses. Indeed, these elements can have an impact on the difficulty of a level. For example, in a game like *Super Mario Bros.*, transformations (*Fire Mario*, *Invincible Mario*, etc.) have an impact on the number of moves the character can take before dying. This has the effect of reducing the difficulty of some passages.

It is interesting to note that the levels' difficulty seem to increase within worlds most of the time, as in the first, fourth, fifth and seventh worlds. Some worlds, on the other hand, have a difficulty dip within their second levels, but the world still ends with more difficulty than it starts. These worlds are the third and sixth worlds. An outlier to this trend is the eighth world, where the difficulty peaks in its second level, and its third level's difficulty falls back down and ends below its first level.

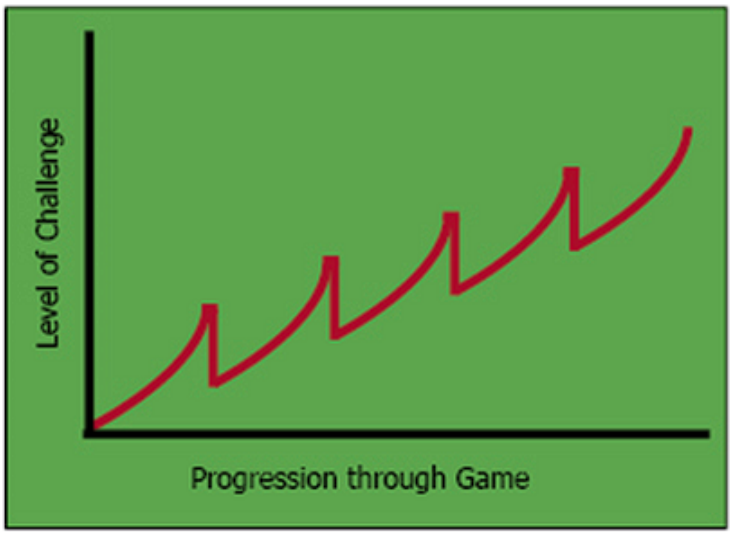


Figure 6.3: Nishikado Difficulty, Holleman (2018)

When compared to the Nishikado difficulty Holleman (2018), the concept of increasing

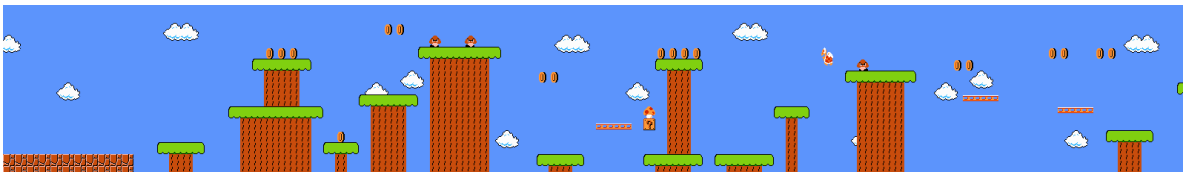


Figure 6.4: First half of level 1-3 of *Super Mario Bro.*

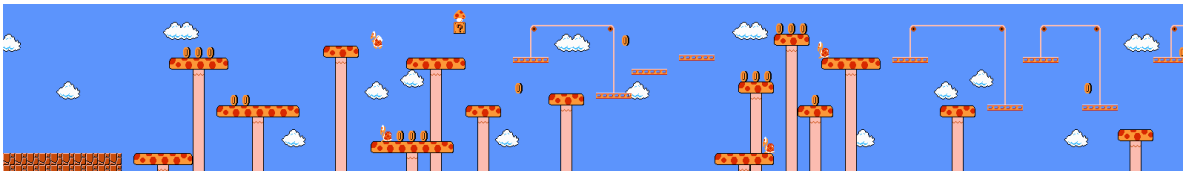


Figure 6.5: First half of level 4-3 of *Super Mario Bro.*

difficulty between worlds such that a world starts slightly more difficult than the previous world started is not present in our results as illustrated by Figure 6.3. This realization leads to a hypothesis on the nuances of the Nishikado difficulty: the difficulty increase within a given world is primarily motivated by topology, while the increases across worlds in primarily motivated by dynamic elements. This hypothesis can further be studied with a more completed version of our model that considers dynamic elements as well as topology, and where levels can be evaluated through a purely static or dynamic lens.

6.1 AREA-BASED METRIC REVIEW

In light of our results, we notice that our two metrics seem to have different sensitivities. The area-based metric M_A seems to be affected greatly by the size of the obstacles. For examples, the levels with the highest $M_A(L)$ difficulty are levels 1-3 (Figure 6.4), 3-3, 4-3 (Figure 6.5), and 6-3. All of these levels are "aerial" levels; that is, the levels are built as a high number of small platforms over a mostly empty space. This effectively generates a single sizable gap spanning the entirety of the level.

It should also be noted that the area-based metric seems sensitive to the overall size of the level. For example, a very small level with only one obstacle will get a higher difficulty score

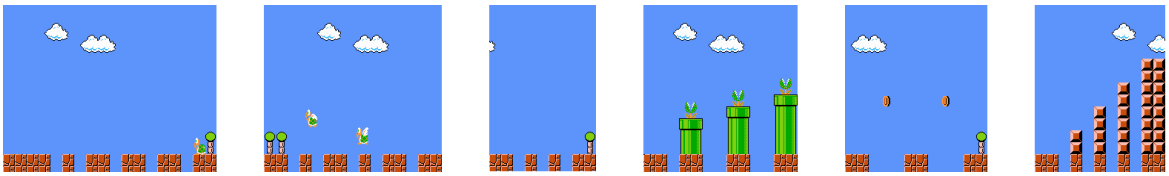


Figure 6.6: Sections of concentrated gaps of level 8-1 of *Super Mario Bro.*

Figure 6.7: First half of level 8-2 of *Super Mario Bro.*

than the same level with additional portions containing no obstacles. This is a side effect of expressing the difficulty as the ratio of danger over reachability.

6.2 PERIMETER-BASED METRIC REVIEW

The perimeter-base metric on the other hand, seems to be most sensitive to the quantity of obstacles. The level with the highest perimeter-based difficulty evaluations are level 5-2, 6-1, 8-1 and 8-2. These levels have between nine and 25 gaps, while the next level with the most gaps is level 4-1 with only six gaps. Furthermore, the distance between gaps also seems to have an impact of the perimeter-based metric. For example, level 8-1 (Figure 6.6) has an evaluation of 122.56 compared to level 8-2 (Figure 6.7) which has an evaluation of 188.14 despite having seven more gaps. A notable difference between those two levels is that level 8-1 has its gaps tightly packed together while level 8-2 has its gaps more evenly spread across the whole level.

6.3 LIMITATIONS

Our model offers a topological approach to difficulty evaluation, but fails to consider a variety of elements in order to be effective on the overall motor difficulty of a game, and be usable in

an industry environment. The current limitations of our model rest in the lack of consideration of three main types of game elements and mechanics. This categorization is made with technical differences in mind that might indicate a need for a separate approach to solving each type individually. The three types are: irresponsive dynamic elements, responsive dynamic elements, and element sequencing.

6.3.1 IRRESPONSIVE DYNAMIC ELEMENTS

Irresponsive dynamic elements refer to game elements that move in space, specifically those that are on a fixed schedule independent from the player. Some examples of this type of element in *Super Mario Bros.* are as follows: *Goombas* walk in a specified direction until they either fall off the level or hit an obstacle and turn around, *Piranha Plants* exiting pipes at fixed intervals and moving platforms following a predetermined path in a loop. A depiction of the behaviour of the *Goomba* and *Piranha Plant* is made in Figure 6.8. Most forms of projectiles also fall into this category. This category revolves primarily around the introduction of the notion of time, breaking our current "snapshot" view of levels.

In order to simplify this problem, irresponsive dynamic elements can be subdivided into either cyclical and non-cyclical elements. A cyclical element refers to elements that repeat in time such as piranha plants and moving platforms and non-cyclical element refers to elements that does not repeat in time, such as *Goombas* and projectiles.

The introduction of time could potentially be addressed by decaying pheromones similar to an ant colony system. Every dynamic element would deposit an amount of pheromone as it moves about the level which evaporates as time goes. For cyclical elements, the pheromone trail would regularly overwrite itself allowing the elements to be evaluated over a single cycle instead of the entirety of the elements' life time. For non-cyclical elements, the elements would have to be simulated until the elements disappear, either by falling outside the level space or by becoming cyclical elements. The later can happen if a *Goomba* gets stuck between

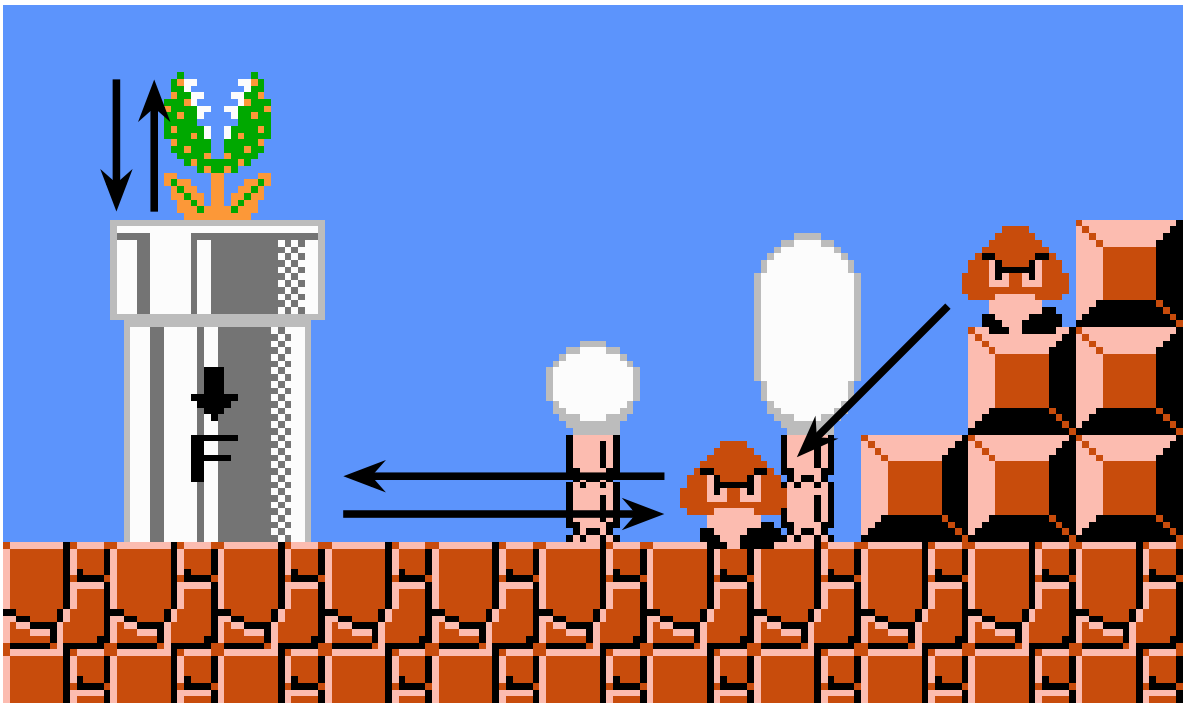


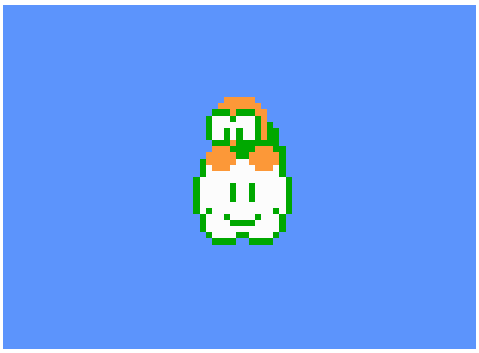
Figure 6.8: Illustration of cyclical (*Goomba*) and non-cyclical (*Piranha Plant*) obstacles from *Super Mario Bros*.

two pipes for example.

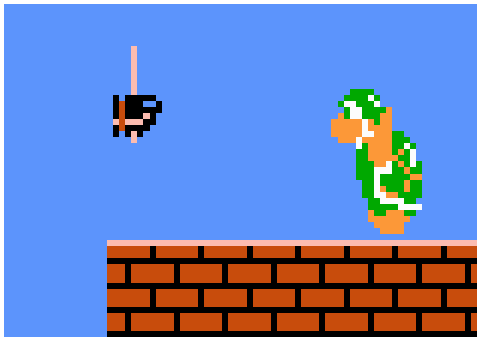
This approach to solving the issue of irresponsive dynamic elements requires elements to have a computable route through the level and to either be repeating at set intervals or have a definite and known life span.

6.3.2 RESPONSIVE DYNAMIC ELEMENTS

Responsive dynamic elements refer to game elements that move in space, specifically those that are dependent on the player. A responsive dynamic element will modify its behaviour in relation to the player's actions. Some examples of this type of element in *Super Mario Bros.* are *Lakitus* and *Hammer Bros* (shown in Figure 6.9). Both enemies throw projectiles aimed at the player. Any game element using player tracking thus falls in this category. Similarl to the irresponsive dynamic elements, this type of element requires the notion of time, but with the added concept of player awareness.



(a) Lakitu.



(b) Hammer Bro.

Figure 6.9: Illustration of *Lakitu* and *Hammer Bro* from *Super Mario Bros*.

The main difference with the irresponsive counterpart of this type, is that the movement of elements can not be pre-calculated. Since the player has the power to change the way these elements operate, the actions of the player must be taken into consideration when handling these elements.

This issue is troublesome as it is somewhat contradictory to one of our objectives in this research: abstracting the difficulty away form the player. Of course, in our case, this refers to human participants and not in-game pawns. It is however undeniable that the behaviour of the in-game pawn is the result of the human participant's actions. In other words, to consider this type of element, the model must propose a clear distinction between the two so that the in-game pawn is accepted as part of the model, while the human participant is rejected.

6.3.3 ELEMENT SEQUENCING

This limitation does not revolve around a specific element, but the sequence of elements. The issue arises in situations where an element imposes a set of restriction onto the player for future elements. In *Super Mario Bros.* for example, a jump's reach is proportional to the velocity of the player when the jump occurs. In a scenario where two gaps are close enough together, the first gap might force the player to slow down as to not overshoot the jump. This causes the player to reduce its velocity and the reach of the second jump. An example of this

is depicted in Figure 6.10.

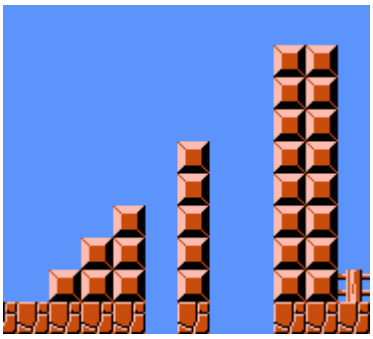


Figure 6.10: Illustration of element sequencing from level 8-2 in *Super Mario Bros*. The middle pillar limits the momentum going into the second jump.

A possible solution to this issue is to generate the history of all possible relevant information for every pixel. In the case of the jump in *Super Mario Bros.*, every pixel would hold the set of all possible velocities at that point. This set could be weighted by probability of occurrence, although it is unclear at this point if added nuances would generate a noticeable difference in results as demonstrated by our gradient versus non-gradient metrics.

CONCLUSION AND PERSPECTIVES

In this work, we were interested in the study and evaluation of the difficulty in video games. The novelty of our approach is to propose an automatic method of evaluation independent of player subjectivity.

We have presented our results based on our proposed approach to model 2-dimensional platformer video game levels in order to calculate its degree of motor difficulty without player involvement. We presented a static theoretical model based on the representation of a level through reachable and danger pixels, and provided two possible metrics. The first metric revolves around the ratio of reachable danger area to the entire reachable area. The second metric evaluates the difficulty as the length of the border between the reachable danger area and the rest of the reachable area.

As an experimentation, we implemented a program in C++ that analyses level images and extracts collision data. It then used that data to evaluated the degree of motor difficulty for various *Super Mario Bros.* levels. With this program, we calculated the difficulty induced by both our metrics on 22 levels of the game. The excluded levels were so as they relied on a different rule set than the included levels.

The first emergent characteristic we noticed is the sensitivity of the metrics. The area-based metric seems to vary more importantly as the size of the obstacles vary, while the perimeter-

based metric varies greatly when the quantity of obstacles vary. We also found the results produced by the area-based metric to more easily understandable and intuitive as it depicts a ratio. This property allows a level to be evaluated in isolation, while the perimeter-based metric requires a reference point.

Currently, this tool does not consider mobile elements such as enemies, projectiles and mobile platforms. This has the effect of accurately describing the difficulty in levels devoid of mobile elements in the best case, and underestimate the difficulty of levels with multiple mobile elements in the worst case. The levels closer to the end of the game *Super Mario Bros.* are the most affected by this because they contain the most dynamic elements. Our current approach can thus be considered a lower bound to the actual motor difficulty of any level.

The next perspectives of this research is to revise our current model in order to consider the three limitations mentioned in chapter 6: non-responsive dynamic elements, responsive dynamic elements and element sequencing. Each limitation can be tackled independently based on their underlying concepts, such as the notion of time and player awareness. Some basic solutions were described in the aforementioned chapter. Once all of the presented limitations are treated, the model should then evaluate with further accuracy the motor difficulty of any level from the game *Super Mario Bros*. It could also be interesting to research the other difficulty types mentioned in chapter 1: sensory, logical and emotional difficulty. Connections could be made with the motor difficulty allowing further development of our model to be applicable to all difficulty types.

From there, the goal could be to expand the model to all games and reach the point of global use. This could be achieved in three parts. The first undertaking would be to transform the model into one that operates on a continuous domain, and away from the pixel-based approach. The second task would be to generalize the model to any type of game. The last modification would be to introduce the third dimension into our analysis. With these changes, our model should be general enough to allow itself to be tailored to any game. This final product could

ideally be made into a plugin or package for popular game engine such as *Unity* and *Unreal Engine*.

BIBLIOGRAPHY

Aponte, M.-V., G. Levieux, et S. Natkin. 2011a. Difficulty in videogames. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology - ACE '11*. ACM Press.

Aponte, M.-V., G. Levieux, et S. Natkin. 2011b. Measuring the level of difficulty in single player video games, *Entertainment Computing*, vol. 2, no. 4, p. 205–213.

Bopp, J. A., K. Opwis, et E. D. Mekler. 2018. "an odd kind of pleasure" differentiating emotional challenge in digital games. In *Proceedings of the 2018 chi conference on human factors in computing systems*, p. 1–12.

Caillois, R. 1958. *Les jeux et les Hommes*. Gallimard.

Chauvier, S. 2007. *Qu'est-ce qu'un jeu?* Vrin.

Crawford, C. 2003. *Chris Crawford on game design*. New Riders.

Denisova, A., P. Cairns, C. Guckelsberger, et D. Zendle. 2020. Measuring perceived challenge in digital games: Development & validation of the challenge originating from recent gameplay interaction scale (corgis), *International Journal of Human-Computer Studies*, vol. 137, p. 102383.

Desurvire, H., M. Caplan, et J. A. Toth. 2004. Using heuristics to evaluate the playability of games. In *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04*. ACM Press.

Djaouti, D., J. Alvarez, J.-P. Jessel, G. Methel, et P. Molinier. 2008. A gameplay definition through videogame classification, *International journal of computer games technology*, vol. 2008.

Ericson, C. 2005. *Real-Time Collision Detection*. CRC Press.

Fraser, J., M. Katchabaw, et R. E. Mercer. 2013. *An Experimental Approach to Identifying Prominent Factors in Video Game Difficulty*. Coll. Lecture Notes in Computer Science, p. 270–283. Springer International Publishing.

———. 2014. A methodological approach to identifying and quantifying video game difficulty factors, *Entertainment Computing*, vol. 5, no. 4, p. 441–449.

Holleman, P. 2018. *Reverse Design: Super Mario World*. CRC Press.

Levieux, G. 2011. Mesure de la difficulte des jeux video. Thèse de Doctorat, Conservatoire national des arts et metiers-CNAM.

Mourato, F., F. Birra, et M. P. dos Santos. 2012. *Enhancing Level Difficulty and Additional Content in Platform Videogames through Graph Analysis*. Coll. Lecture Notes in Computer Science, p. 70–84. Springer Berlin Heidelberg.

———. 2013. *The Challenge of Automatic Level Generation for Platform Videogames Based on Stories and Quests*. Coll. Lecture Notes in Computer Science, p. 332–343. Springer International Publishing.

———. 2014. Difficulty in action based challenges. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. ACM.

Mourato, F. et M. Santos. 2010. Measuring difficulty in platform videogames. p. 173–180.

Nakamura, J. et M. Csikszentmihalyi. 2014. *The concept of flow*. Coll. Flow and the foundations of positive psychology, p. 239–263. Springer.

Van Rozen, R. et J. Dormans. 2014. Adapting game mechanics with micro-machinations. In *Foundations of Digital Games*. Society for the Advancement of the Science of Digital Games.

Volkovas, R., M. Fairbank, J. R. Woodward, et S. Lucas. 2020. Practical game design tool: State explorer. In *2020 IEEE Conference on Games (CoG)*, p. 439–446. IEEE.