Table of Contents

[1 Analysis 1](#_Toc3904577)

[1.1 Background and identification of the problem 1](#_Toc3904578)

[1.2 Market research and Benchmarks 1](#_Toc3904579)

[1.2.1 The Legend of Zelda (1986) 2](#_Toc3904580)

[1.2.2 Shovel Knight Treasure trove (2014) 4](#_Toc3904581)

[1.2.3 Legend of Zelda: Breath of the wild 2D prototype 4](#_Toc3904582)

[1.2.4 Metroid (1986) 4](#_Toc3904583)

[1.2.5 Pokémon D/P/Pt (2006) 4](#_Toc3904584)

[1.3 End users of the investigation 5](#_Toc3904585)

[1.4 Target market 5](#_Toc3904586)

[1.5 Specification: 8](#_Toc3904587)

[1.6 Specification justification 9](#_Toc3904588)

# Analysis

## Background and identification of the problem

The video game industry is a major player in the entertainment sector, reaching a global net worth of $135bn in 2018, and 2.3bn active gamers worldwide. (Wijman 2018). It is estimated to reach $180bn in 2021 after having experienced exponential growth in the past 6 years. As a comparison the TV and video industry is currently worth $136bn as of 2018. (Wikipedia 2019).

A specific attribute of the video game industry is that it is a mix of large international corporations (Microsoft, Nintendo, Ubisoft among others) and small Independent studios (Chucklefish, Yacht club games, Team Cherry, etc). For the latter, optimising time and investment is paramount, and therefore it would be interesting to see if there are some key factors that play into the success of the best-selling and most popular video games of all time.

The sector on which I focussed my research was 2D Action-Adventure games, specifically the movement mechanics of the objects inside the game. My aim for this project is to determine how big a part the mechanics of a game play into its success by looking at past games and testing the appeal of certain recurring and inventive mechanics by coding them into a simple 2D game, and testing it on a restricted sample.

## Market research and Benchmarks

For my investigation, I decided to look at simple 2D platformers and overhead perspective games such as the original Legend of Zelda and Metroid games, that are known classics that defined their genres and the video game industry.

I looked what made them work so well and how they integrated their well-polished mechanics with their gameplay. However, since the game industry has progressed much since then, I also needed to look at some more recent titles to see if the kind of gameplay offered by these games was still successful.

To this end, I decided to also take a look at more recent games and prototypes like the indie hit game Shovel knight, and the 2 dimensional prototype for breath of the wild (a prototype version of the game that was used to test out possible game mechanics), where the whole point was making a fluid system where everything would be able to seamlessly interact and react to the environment around it.

Another thing I wanted to look at was the way the overworld was modelled in main series rpg titles such as Pokémon, since despite the main intrigue being the combat system, the puzzles and level design are quite interesting, being rooted heavily in exploration, and there have been innovative Physics ideas in some titles, incorporating Inertia and momentum with bikes in generation 4 (Pokémon Diamond, Pearl and Platinum), and having collisions and Friction being a large part of some puzzles, as well as exaggerated forces due to winds and dealing with sinking into holes and climbing out of them, as well as breakable objects.

### The Legend of Zelda (1986)

The original Legend of Zelda game is one of the most famous games to have ever been created, and revolutionised the gaming industry, selling over 9.03 million copies (Fandom 2019). It involved non-linear gameplay and forced players to think about what to do next, as well as encouraging sharing and communication between players to allow them to unlock the different secrets stashed within the game, as opposed to the multitude of linear games that existed at the time. It was also modelled as a coming of age story and featured a progression system whereby the player and Link would grow stronger together.

The game simulated a sort of forced fake three dimensional perspective on the player to make it seem as if they were looking diagonally down onto a field, instead of the world seeming completely flat, a method still widely used in video games today.

The character is only able to move in the cardinal directions and with a highly accurate collision detection system. Link's (the name of the primary protagonist) movements and sprite animations are fluid and smooth giving the illusion of fully analogous movement, giving a sense of fluidity to the controls. Seeing as how this level of fluid control has become a staple of mainline Action adventure games and future two dimensional overhead view Zelda games, it is safe to say that this is a key factor in the success of these games.

While it seems that some obstacles are themselves pixel accurate on their hitboxes or the bounds where entities such as the player can touch them, the majority of the various obstacles of the game such as rocks and trees are not. These obstacles occupy a space which is around the same size as the player, and have their sprites reused throughout the game, the same can be said of the basic enemies inside the game and their spawn animations, who themselves are all no larger than the player, and seem to appear in the same square positioning grid as the obstacles, possibly indicating that these spawn positions and block positions are stored in a file which indicates their co-ordinates. Something which could definitively be a possibility, enhanced by the fact that the game was built to be able to run on the NES's limited hardware restrictions, whilst trying to make sure its assets took up the least amount of memory in the cartridge, to increase the amount of things that could be fit inside the game. This could mean that to save space and/or to reduce the complication of placement calculations, Nintendo used a different system than representing each pixel as a single x/y unit. This is also visible from the paths and gaps in the walls and the size of large objects such as lakes and rivers having their dimensions being exact multiples of the player's size. This is made even more apparent from the fact that the walls of the early stages of the game are formed from blocks of the same sprite, which are again the same size as the player. Conversely, this could have been done to more easily build the world. However, since the entities in the game seem to also be able to move to some extent inside a small area (as large as the player indicating that the possibility of movement of the and collision with solid objects of the entities is not calculated from the same hitbox as the one used to judge projectile and attack collisions. The level of accuracy of these collisions must also mean that the size of the hitbox and its current bounds as well as the projectile or attack's current location and hitbox are used in calculations to find if an object hits. Having files to store levels as images, or as maps and creating levels out of smaller pieces seems to still be a mechanic used today in some game creations, with heavy use in 2D platformers and top down Roguelike RPGs where the maps can be generated randomly from modular rooms.

The game also has the ability to manage multiple enemy projectiles at the same time. The different projectiles of the game also have different properties, with some being able to go over obstacles, whilst others would not, prime examples of the former being Link's beam sword and the Moblin's arrows and an example latter being the Octoroc's rocks. This could mean that for possible collisions, the game looks up the projectile's values to see if it is stopped or it goes through the object it is currently touching. One thing to note about the projectiles is that most are generated behind the sprite of the current object emitting it, to make it seem as though they are being shot by that entity. An exception to this is Link's beam sword which is generated under the sword of Link's attack animation. The projectiles also seem to have a bigger z-display position than the obstacles in the game, since they appear to pass over them. An interesting mechanic that is not obvious at first is Link's ability to block projectiles with his shield. If Link is not currently attacking and is facing the projectile coming towards him, it will bounce off diagonally.

Although the game is filled to the brim with enemies, a lot of them are variations and re-colorations of each other. These species of enemies all have the same behaviour, with the differing colorations indicating maximum health. This allows the player to feel a sense of progress as they might struggle at first with weaker enemies, but as they and Link get stronger from overcoming trials and adversity, this allows them to be able to take down stronger enemies.

### Shovel Knight Treasure trove (2014)

### Legend of Zelda: Breath of the wild 2D prototype

### Metroid (1986)

The original Metroid sold 2.73 million copies for the NES (Fandom 2019). It was strongly celebrated for its non-linear exploration and discovery system much like the Legend of Zelda. What set it apart from its counterpart however was its vertical platforming element and darker atmosphere. Along with Castlevania, it helped create its own genre of games (Metroidvania), and is no doubt one of the most influential games of all times.

Much like Zelda there is a strength of progression along the game, as the player must unlock permanent power ups in order to be able to move forward. The game is also heavy in backtracking, as once the player has unlocked a new power up, areas that were previously blocked off might now become available.

Also visible is the way the levels are designed with the rooms of the game being comprised of small blocks which are half the size of Samus, which again displays the methodical approach to designing the world or map to be compact and easier to store, like described above in the original Legend of Zelda section[[1]](#footnote-1).

The game also includes a feature which allows you to save your progress through the use of a password system, that you have the ability to input into the game to access your progress on another machine.

### Pokémon D/P/Pt (2006)

For my research, I decided to look at one of the most best-selling Pokémon games: Diamond and Pearl, having sold over 17.67 million copies worldwide (Nintendo 2018). However, the main selling point was the combat system, although throughout the years, Pokémon has added some staple movement mechanics to its games, with this title being no exception. The game features a complex non-linear overworld, which much like Metroid talked about above requires the player to backtrack before advancing, and gives players a sense of mystical mystery with the atmosphere set by the music and graphics in the sparse secret areas scattered throughout the game. Despite the players having to backtrack through areas they have already visited, the game still manages to push for a sense of exploration through the multitude of various intertwining passages in Mount Coronet's vast cave system, through the use of height differences and platforms, as players will find themselves going back through rooms they have already visited albeit in a different part.

Another interesting factor in the games is the Bike's gearing system. The Bike gives the player the ability to switch gears between third and fourth gear, where fourth gear has some acceleration mechanics and has the ability to travel faster, at the cost of mobility. In some puzzles, the player will have to accelerate the Bike over a distance in fourth gear to be able to climb some muddy slopes to get to new areas, or have to methodically choose which gear to use when having to clear ramps, such that they are able to land in the correct spot to be able to progress. One notable example of such puzzles is Wayward cave, where there are four ramps where the first three jumps have to be performed in fourth gear and the last performed in third. One thing to note is that it is possible to fail to climb the slopes or ramps if the bike does not have enough momentum, with the game having the ability to calculate the height to which the player will climb when they attempt to ride up the slope, and if they do not have enough of it, to make them slide down and away. This acceleration and momentum mechanic for the Bike's fourth gear is also present in the way that when moving fast in fourth gear, the player will experience more difficulty stopping the bike and turning it.

Yet again, looking at the overworld, we see the modular design mentioned so much in Metroid and Zelda, however, in this case, it also uses some of the DS's 3D model capabilities, as visible from the way the buildings are rendered and the design of the walls inside the game's caves. The maps of the D/P/Pt games are broken up into squares of size 32 by 32 tiles. To save memory and processing power, only four of these squares are loaded at once, and in such a way that the world appears uniform and seamless (Bulbapedia 2018). What is also apparent is that the ground textures used in the game for certain areas are also only the size of a tile. Unlike the other games mentioned above however, the processing and data collection is much less obvious due to the fact that the 3D models and sprite designs make it such that the objects of the game blend much better together, as well as the presence of larger objects that take up multiple tiles.

In contrast to the more variable movement of the above titles, the player only has the ability to move by a single tile in the cardinal directions inside the game, albeit the speed at which it happens can vary in some cases[[2]](#footnote-2). Whereas this mechanic would be seen as a hindrance and a downside to the game, it allows the game to be able to generate its random encounters on the different possible tiles of the game, randomly deciding using the encounter probability whether the next tile the player steps on will have a random encounter and if so, what Pokémon it will be. This mechanic remained a staple feature of the franchise itself until Pokémon made the jump to 3D, and even then, in 3D game for the overworld, caves, and most areas, the same type of tile system can be seen.

Some tiles in the game will also have different properties, with slippery ice tiles preventing the player from changing their initial direction of motion, and making the players sprite slide along until they either land on another tile, or collide into an object, allowing for some complex puzzles. Other tiles such as Marsh tiles and Snow tiles can restrict the player's movement by trapping them in a hole temporarily, forcing the player to think about what they do next to make the most efficient path.

## End users of the investigation

The end users being video game developers interested in understanding what mechanics make a game most successful and appeals the most to their consumer base, I reached out and asked a few questions to:\_\_\_

## Target market

Another part of my research to find out what core mechanics would be successful was asking and interview people for what kind of things they looked for in top-down 2D video games. So I interviewed a multitude of people, asking them questions as to what really attracted them towards video game titles, as well as what kinds of attributes they want to see and use to discern what kinds of games they want to buy.

|  |  |
| --- | --- |
| Attribute | Reason |
| Intuitive | The game has to be easy to pick up and easy to control, you have to be able to tell what the controls do, instead of an unnecessarily complicated system that makes the game hard to pick up and play for new players. |
| Not having fake 3D and 2.5D elements | Making the game in a fake 3 dimensional element, or 2.5D when it doesn't need it is unnecessary and distracting, it doesn't add anything interesting to the game in itself. However, detailed dynamic backgrounds to give a sense of a larger world like in games such as: "Shovel Knight or Starbound" work, because they give a sense of dynamic movement otherwise lacking in some 2D platformers. |
| Good music | In general, good music helps set a theme in a game, as well as emphasising the in-game atmosphere to help immerse the player into the world. Multiple people told me that music was a large factor when they were deciding whether to buy a game. |
| Mechanics | When looking at a game, the basic movement and control mechanics have to usually be present, but what really helps to set a game apart are the unique movement mechanics introduced to the player in the game. What's important though, is that these mechanics don't feel shoehorned into the game, but the game and level design is built around these mechanics, forcing you to use and exploit them to your advantage to do well. Making these mechanics satisfying to pull off and use also makes the game more enjoyable, an example I was given here were the Shovel Knight: Treasure trove Shovel Knight, Spectre Knight and Plague Knight Campaigns where the levels of the game all feature the same areas and enemies, but the level design is strongly changed, as are the behaviours of the bosses to make each of the game modes play around the characters and their abilities completely. Another game mentioned was "Celeste", a platformer featuring strong Physics elements, that are used consistently throughout the game. |
| Level Design[[3]](#footnote-3) | Much like described in the Mechanics section, the levels have to be streamlined, designed around the mechanics of the game, to make the gameplay more interesting, a good starting level is one that acts as a sort of tutorial without feeling as one. Most players do not want their hands held through what seems to be the umpteenth platformer they've played, or suffer through multiple hours of expository dialogue on how to perform a single mechanic which will only end up helping them once throughout the game.  Of the 2D games that are most enjoyed and positively viewed for their level design are the ones where no part of the level is obsolete or added just to make the game look good. Every part of the level design has a purpose, to enhance the experience provided by the gameplay.  It is also important that the level give options to the player as to what they are and aren't able to do, give them multiple paths for tackling a problem and reward the player for their creativity and skill.  The games I saw most praised for this were: Shovel Knight Treasure trove and Hollow Knight. |
| Complexity[[4]](#footnote-4) | People like complexity in games where it develops over time, forcing the player to become better, to think more about their actions, without ruining its original intuitiveness. In other words, featuring a difficulty curve to spice up gameplay and push the player to become better. An example for this I was given was the hit game "Super Hot", where the game starts of easy, and with each progressive level becomes harder, taking the player up a learning curve, leading to very difficult and complex gameplay later in the game.  On the other hand, it is important that the game be not swamped in obsolete mechanics such that they overshadow the actual gameplay itself and push the player away from it. |
| Story | Whilst most games have a story, it is important that the story take the backseat, it should itself be based around the gameplay and mechanics, and work in conjunction with the gameplay, not against it. Having the story be completely unrelated to the game serves no purpose whatsoever and often ruins immersion and experience. |

## Specification:

Having taken in the most common and recurring aspects of the above games, I settled on making a 2D top-down game/simulation.

1. The game should have a clear user interface.
   1. The user should be able to have a clear understanding of what is going on where and when.
   2. The user should have a clear idea of the controls and how they work.
   3. The graphics implemented should be smooth and fluid, not slow and choppy.
2. Entities and moveable objects should have fluid multidirectional movement bound by set rules
   1. The players and entities should have the ability to move in both the Cardinal and Ordinal directions.
      1. The movement of the entities should be well defined and the player should feel in control at all times
      2. The sprites of objects should be altered in real time to show their current status, ie direction.
   2. The entities of the game should be subject to set Physical laws dictated by their attributes.
      1. If an entity keeps moving in the same direction for some time, they will accelerate to a maximum threshold value instead of moving at a constant speed.
      2. The entities of the game will be subject to laws of conservation of momentum and Inertia and Impulse, such that they will experience rebound when colliding into something, and have to decelerate over a distance when changing direction. These attributes will be small however, to make sure that the player does not feel their character is out of control.
      3. The collisions of the entities will be dictated by their in-game size, and will be strongly implemented to prevent an appearance of moving into objects.
3. The in-game objects should have multiple properties that dictate how the player can interact with them, defined outside the program and fetched from a database.
   1. These properties should be defined by their type of class.
   2. The objects will be defined in a general database which instructs what kind of object they are, so that the program is able to refer to the database for those types of objects to use parameters specific to that kind of object.
      1. The primary database for the objects should refer to other smaller databases specific to each type of object, through an ID system.
      2. Each different object in the game should have a unique ID to identify it.
   3. Objects should have properties that allow their states to be altered, so that other objects and the player can interact with them.
4. The game should be able to build levels from pre-made maps, that identify the positioning of certain objects.
   1. These maps should refer to the different types of objects in the game and their positioning using their ID, in order to initiate the game.
   2. The maps should constitute of a set size with a set amount of tiles contained within each map.
      1. Each object added to the map should only take up one tile.
   3. The player should eventually be able to move between areas on a larger (in-game) world map.
      1. This movement should be signalled by a screen animation visibly displaying this shift in area while the area is being loaded in.
      2. The game should be made such that it only displays one map at a time and clears the screen from objects whenever the area changes.
5. The game should be able to fetch the data stored in text and database file no matter where the project file is, on any computer, such that file location errors do not occur.

## Specification justification

1. When playing a game, it is of the utmost importance that there is no ambiguity to the player as to what is happening on screen and what they are doing. What is clear in the most enjoyable games is that from the get go, you know what you are and aren't able to affect on screen, so that players don't get lost.
   1. When the player does something, or an object moves on screen, it is important that the game effectively feeds back to the player the output of what they just did.
   2. The controls of the game should be made evident and clear to understand, the game should not be flooded with useless controls and button checks for very specific action, and should have a clear idea of what they can do and how they control the game.
   3. The input and controls of the game are important to how much the player is able to enjoy the game. Looking at the above games researched, none of them were successful because they had bad gameplay, conversely, what sold most of them to audiences so well was because the gameplay was smooth and fluid and responded extraordinarily well to user input.
2. For the game to feel more interactive and structured, adding well defined motion and movement mechanics is important such that the player is able to feel a complete sense of control over their character.
   1. Only allowing movement in the Cardinal and Ordinal directions was chosen due to the simplicity of the design and input requirements, since making it possible to control the character in 360° would be far too complicated to control and implement, not to mention unnecessary, seeing as most games owe their success to making the controls simple and easy to use.
      1. Making sure that the player is happy with the way they are able to control the character is one of the primary aspects of making sure that they want to play the game.
      2. Using sprites to show their current stateeffectively communicates to the player what they are doing, where they are heading, and is a feature that is present in the Legend of Zelda, Pokémon D/P/Pt, Shovel knight.
   2. Making the game bound by Physical laws adds a level of depth and complexity to the game, instead of making it fairly linear and boring, adding some attributes that make the game feel more alive and responsive, as well as allowing some situations that would force the players to use or manipulate these mechanics to their advantage to solve puzzles.
      1. Acceleration is an interesting mechanic because it allows the world inside the game to feel more real and concrete. As described above in the description of Bikes in the Pokémon diamond and pearl section, it allows for some interesting puzzle ideas as well. Although pushing it to too much of an extent reduces the control the player has over a game, and prevents makes manoeuvring more difficult and dissatisfying for the player.
      2. Implementing collisions and impulse such that momentum is conserved, as it forces the user to think about how they are going to pilot their character, without recklessly running into things as this could undermine their progress, for example implementing sliding tiles and a slippery surface at some area with obstacles to manoeuver around carefully.
      3. Dictating collisions by how large the sprites are and how much they overlap instead of hidden values, again increases the communication the game has with the player and lets them know that when they go into an area, they will collide with that object, or how exactly they need to move in order to not get hit by an enemy or a projectile.
3. The addition of in-game properties defined by the class of the user allows control over their mechanics of interaction, as well as a definition to be able to integrate them into the game and have them walk around and interact with the player.
   1. Implementing different classes for the objects allows the formation of derived classes, where a derived class could be defined for a very specific type of object that acts differently, as well as allowing the game to compare the objects to each other more easily, by being able to fetch and set its values. It also allows the creation of functions of the derived classes that are able to take in inputs from a member of the derived class when fetching its values from files. It also means that instead of having one type of uniform structure and table for all of the objects of the game, where a large part of some values might not end up being used for objects resulting in a waste of space and potentially resulting in errors, the game can fetch the specific values from independent tables for each type of object. Using a class to derive classes for all objects also allows data structures containing multiple different kinds of objects to be formed (ie a map of the objects and their positions).
   2. Using a database to define the objects allows their values to be flexible and altered during development to balance and correct the game so that the objects are the game can function coherently with their design, as opposed to hard coding them into the game, where it would be necessary to navigate through the program and needlessly create a different case for what values to generate/attribute depending on the object being created.
      1. Referring to the other databases allows identification and normalisation, of the database to ease manipulating the data and the time taken to search through the document, as well as making the attributes specific to each kind of object easier to manage, change and add.
      2. Referring to an ID instead of a name prevents collisions between objects which might have similar names, and makes it easier when creating maps of the environment, as well as making labelling sprite images easier.
   3. Allowing the objects to have changeable states such as health speed, breakability, movability and so on, allows the game to become more complex, as well as allowing the addition of enemies the player can kill, such factors are present in all the games Researched, and is clearly a core mechanic when it comes to such games.
4. Building levels from pre-made maps and fetching them from files is also a core and instrumental mechanic that is visible in all the above games, as it allows level design, and the linking of separate areas of the game; giving the player the ability to move from one 'room' to another seamlessly.
   1. Referring the different types of objects by their ID allows the program to uniquely identify the objects and create a new instance of them at that position when generating the map.
   2. The tile system, is something we see in Zelda, Metroid and Pokémon, all generating the base map for the world using a set tile system, where large walls or structures are made from the combination of many smaller tiles.
      1. Making it so that each object of the map only takes up one tile prevents overlapping when loading the game, as well as making sure that all the maps have one uniform standard for reading them and generating the objects from their data.
   3. Allowing the player to move between areas of a larger world map adds a sense of openness and largeness to the world making it seem larger and more interesting, making it accessible to move between areas, instead of making the game one giant also makes the world seem bigger, and makes it easier to manage the objects loaded into the memory rather than allowing scrolling.
      1. Visibly displaying a movement from one area to the next using an overlaid animation both gives time to the program to load in the next map, and gives the player time to figure out where they have ended up due to moving to an adjacent map.
      2. Making sure that the game clears the screen from objects and sprites before moving to the next is highly important to make sure that the game does not appear to bug and ruin the gameplay and immersion experience.
5. Making it such that the game is able to fetch the files reliably is necessary to keep the program functioning to prevent any issues or bugs, so that the game can be reliably shared and tested between devices, independent of user.

1. See paragraph 4 in The Legend of Zelda (1986) [↑](#footnote-ref-1)
2. The player can move faster by running or riding on a bike on solid ground [↑](#footnote-ref-2)
3. Tied in with Mechanics [↑](#footnote-ref-3)
4. This section ties in with "Intuitive" [↑](#footnote-ref-4)