PROJECT DOCUMENTATION

This section of my deliverable will explain the Game Design and systems in Chroma Mancer, as well as how I designed and developed the game, via its Game Design Document and Development Documentation

The game can be downloaded from the Chroma Mancer website at:

https://chroma-mancer.com/

Overall Table of Contents

CHROMA MANCER

A game by Sebastiao "Sam" Casaleiro

Part 1: Game Design Document

TABLE OF CONTENTS

## Brief Overview

ELEVATOR PITCH

A Bullet-Hell survival game with RPG elements where you draw your own character and unlock new colours as you level up. Each colour is an attribute, and the way you use the colours affects your gameplay stats.

GENERAL INFORMATION

Genre: Bullet-Hell, Bullet-Heaven, Survival, Top-down shooter, Rougelite

Platform: PC

Players: Single-player

Language: English

Similar games: Binding of Isaac, Vampire Survivors, Scribblenauts, Nuclear Throne

Target audience: fans of similar games; fans of rougelites; artists; people who enjoy: customisation and avatar creation, power fantasies, overcoming challenges, expressing themselves and having a unique personal avatar.

UNIQUE SELLING POINTS

- players can draw anything they want whatsoever within the given canvas
- highly replayable due to endless combinations of colours and drawings
- multiple unique stats encourage users to replay the game with different builds
- game is highly customisable and personisable - all sprites can be modified, as well as the colour palette and background
- heavy emphasis on movement mechanics (backflip!)

BASIC CONCEPT

A pixel art top-down bullet-hell survival game with RPG elements set in a big open level with procedurally generated obstacles. Players draw their own character using a limited colour palette and fixed canvas, and customise/change it every time they level up. Their character drawing directly corresponds to their gameplay stats, as each Colour represents a gameplay stat, and the number of pixels in that colour corresponds to the number of points you have in that stat.

## Story

The Player character was innocently exploring the forest surrounding the village where they live. After getting deep into the woods, they come across a Colour Palette and Paint Brush. They start having fun and drawing creatures with these tools in their notebook. Unbeknownst to them however, these are ancient powerful magic artifacts that were thought long lost. The creature drawings come to life and become hostile. They spread, multiply, roam across the countryside and start terrorising the village and its people. Having unwittingly unleashed monsters into the world, the player feels remorse and the need to fix their mistake.

Thankfully a mysterious Owl - the protector of these artifacts - shows up and explains the situation to the Player: the monsters stole the colours when they were unleashed, so the Player must now defeat them and regain the colours - put them all back in the Palette and return the magical artifacts to their rightful place. Each enemy has a bit of Colour essence (in the form of ink drops) inside them that the player collects by killing them. Absorb their power and collect enough of them to regain a Colour. Using the Paint Brush, which the Player thankfully still has, they can manipulate the Colours to enhance themselves and become more powerful. The Mysterious Owl also grants the Player the power to shoot magical projectiles to help them in this quest.

The Player must now destroy all these monsters, collect all the missing colours, and return the artifacts to the Protector Owl!

## Gameplay Loop

Players start at Level 0 and max out Level 6. Their Level represents how many colours the Player has unlocked (there are 6 colours/stats available in total). At the start of the game, they only have the colours black and white available to them. These are "dummy colours" that do not impact stats, and players must create a drawing using both, either, or none of them at the start of the game within the given canvas.

After the player has finalised and saved their character drawing, the game begins. Every few seconds, a wave of enemies will spawn. The type of enemy and its number are pre-determined to assure that the game is balanced and has a steady difficulty curve, but their spawn position in the game world is randomised. The player will now attempt to kill these enemies by shooting them with the 4-directional projectiles, while dodging their attacks and the obstacles present in the map. They can make use of the extra movement mechanics such as the Dodge and Backflip to get past obstacles and outmanoeuvre the chasing horde of enemies. Also, the terrain/obstacles can be used to your advantage by leading enemies into running into a bush or tree and getting stuck. Upon killing an enemy, they become a corpse on the battlefield and drop an Ink Drop (XP Item). Additionally, if the Player has taken damage, there is a 1/6 chance the enemy will also drop a Heart (healing item). The player should attempt to collect these Ink Drops quickly so that they may level up as soon as possible. Regardless of whether the player kills all or any of the enemies, the next wave will always spawn after a fixed number of seconds.

Once the player has collected enough Ink Drops, they level. The player can now pick a new colour/stat, between 3 different randomly chosen colours, to unlock and add to their palette. This allows them to increase the value of that respective stat proportionally to the number of pixels that they paint using that colour. After that, the Colouring Notebook UI opens again and Players are free to draw again. They can choose to expand upon their previous character drawing, or re-draw it completely. Not all colours need to be used, and no changes need to be made if players are already satisfied with their current build. After levelling up, the player is fully healed to their maximum HP.

This loop continues until all enemies in the pre-determined spawn waves have spawned and the player reaches the maximum level. Once the player reaches the maximum level (Level 6), they cannot level up anymore as there are no more new colours to unlock, and their character drawing cannot be changed. However, the game is only over once all enemies have been killed! In the current prototype, reaching the maximum level is not required to achieve the win condition, so you could technically beat the game at levels 0-5. However, nobody has done that!

## Mechanics

COMBAT

Players have one tool for dealing damage to enemies: their projectiles. They can be shot in four directions (up, down, left, right) using the arrow keys. However, projectiles can also travel diagonally. An offset is applied to give shooting more momentum and nuance when the shooting direction and movement direction are perpendicular. For example, if the character is moving Up (🡡) and shooting Right (🡢), the projectile travel diagonally up-right ↗ (i.e. north-east).

How much offset is applied depends on long the character has been moving for: the longer they move in that direction, the bigger the diagonal offset. Additionally, if players are moving and shooting in the same direction (ex: moving right and shooting right), the projectiles will receive a speed boost.

MOVEMENT MECHANICS

Besides simply moving in 8 directions, there are extra options available to the Player to navigate the game world and outmanoeuvre enemies. Both options cost 1 Stamina point to perform. They are:

1. a dodge roll
2. a backflip

1. Dodge Roll

The dodge roll behaves similarly to games like Dark Souls and Enter the Gungeon. It grants the player character a speed boost during which they are invulnerable for a fraction of a second (default value is 1/4 second). However, their collider is still enabled so they can pick up items like Ink Drops and Hearts, but cannot phase through enemies or obstacles, while dodge rolling.

2. Backflip

The Backflip is a very versatile and powerful movement mechanic. It was inspired by the Super Mario backflip, in particular from the game *Donkey Kong* (1994). Backflipping makes the player character vault backwards into the air in a Sine Wave motion. Like the dodge roll, it gives the character a speed boost and makes them invulnerable to damage for the duration of the movement. However, it also enables the player to jump over any enemy and obstacle.

If mastered, these grant the Player a lot of options for avoiding enemies, positioning themselves in favourable situations, and truly dominating the battlefield.

## Game Systems

DRAWING SYSTEM

The main Unique Selling Point of Chroma Mancer is this system wherein players freely draw their own character in a given canvas. This drawing becomes their character sprite in the game world. Their gameplay stats are extracted from this drawing by analysing every pixel, counting the occurrence of each colour, and assigning 1 point to that colour's respective stat for each pixel painted with it.

The drawing system - or as I called it, the Colouring Notebook - is a simple pixel art drawing program inside the game. It has three main elements:

1. *the Canvas* (a spiral notebook with squared paper)
2. *the Colour Palette* (a wooden painter's palette)
3. *the Sidebar* (a piece of scrap paper)

All these elements have been designed after real objects to feel diegetic and like they belong to the game world. There is also a paint brush cursor whose colour changes to match the currently selected colour.

1. THE NOTEBOOK

This is element houses both the canvas for the drawing, as well as a preview of the stats you would get from the current drawing.

The canvas is a 16x16 grid that fits seamlessly within the squared design of the notebook page. A thick dashed outline has been drawn around the edges to clearly delimit the size of the canvas, and each pixel is clearly separated with the checkered design. Players can Left-Click any square in the canvas to paint it using the currently selected colour, or Right-Click to erase it (making the pixel transparent). It's simple but effective, and took a lot of effort to make :)

On the left side margin of the notebook page is the controls/instructions, explaining to the player how to draw and erase.

On the right side of the page is the Stats Preview, displaying how many pixels the player has coloured using each colour, and how many points they would get into each stat when they save the drawing. It only displays stats for the unlocked colours to prevent needless UI clutter and player confusion.

2. THE COLOUR PALETTE

This wooden palette overlayed on top of the notebook displays all colours available to the player as paint splashes on the palette. Each paint splash is different to make it look organic and diegetic. The default colours are black and white, with more colours being dynamically added as the player levels up and chooses new colours to unlock. Clicking on any of these paint splashes determines the currently active colour - the colour that the player will paint with when clicking squares in the canvas. The Cursor (in the form of a Paint Brush) is re-coloured to match that selection in order to make it clear to the player which colour they are currently painting with.

3. THE SIDEBAR

A small piece of scrap paper on the side. It contains a real-time to-scale preview of the player's drawing (so that they may get a feel for what their character will look like in the game world) as well as some UI buttons. These buttons are:

- Save
- Reset (displayed as circular arrows icon)
- Clear (displayed as a trash icon)

The Save Button finalises the drawing process, setting the new drawing as the player's character sprite and deriving the new stats from it. It also saves the drawing to disk inside the game folder. This closes the Colouring Notebook and resumes the game.

The Reset Button resets the canvas to its original state. In case the player made some changes to their character that they don't like, they can revert back to the way the character was before any changes were made (i.e. when the Colouring Notebook first opened). It's sort of like a big undo button.

The Clear Button erases the entire canvas, setting every pixel to transparent. In case the player wants to delete everything and make a new drawing from scratch.

STATS SYSTEM!

There are six stats/attributes in the game with corresponding colours. They are:

1.	Health:	Magenta
2.	Attack:	Purple
3.	Movement Speed:	Cyan
4.	Fire Rate:	Yellow
5.	Shot Speed:	Bright Green
6.	Stamina:	Dark Green

1. HEALTH

> Num of Hearts you can have

Health in Chroma Mancer uses a Hearts system similar to Binding of Isaac or Zelda. Players' health is represented by:

- Full Hearts: 2 HP
- Half Hearts: 1 HP
- Empty Hearts: 0 HP

The maximum number of hearts is always displayed. Hearts become smaller as the player's HP is depleted. For example:

- if the player has 6 maximum HP and 6 current HP, their health is shown as 3x Full Hearts (2+2+2=6)
- if the player has 6 maximum HP and 3 current HP, their health is shown as 1x Full Heart, 1x Half Heart, and 1x Empty heart (2+1+0=3)

The base value for Health is 2 HP, so Players always start the game with 1 Full Heart. If they wish to increase their maximum HP, every 20 points in the Health stat (i.e. every 20 Magenta pixels in the character) will add 1 new Full heart. Maximum HP is only increased in increments of 2 (full hearts).

2. ATTACK

> Damage dealt by projectiles

Attack in Chroma Mancer is an integer value (i.e. 1, 2, 3...). It determines how much damage your projectiles inflict on enemies. The base value is 1. Each 30 points into the Attack stat (i.e. every 30 Purple pixels in the character) increase the Attack value by 1. The HP values of enemies range from 5 to 15, so every extra damage point is a big noticeable upgrade. The Chaser enemies go from taking 6 shots to kill to only three shots after increasing the damage value by 1.

Additionally, every extra attack point increases the projectile size by 25%.

3. MOVEMENT SPEED

> How fast you move around

Movement Speed in Chroma Mancer determines how fast your character moves around in the game world. It also affects the distance of your Dodge Roll and Backflip. The base value is 2.0, and every 30 points in Speed increases your movement by 1. It's a decimal value, so every pixel in Speed does increase your movement very slightly. Players should be carefully, because they might find it hard to move accurately and hit their targets if they put too many points into speed.

4. FIRE RATE

> Shots fired per second

Fire Rate in Chroma Mancer determines how many projectiles you can shoot per second. It's a decimal value and the base value is 2.0. Every 20 points into this stat increases the fire rate by 1 extra shot per second.

5. SHOT SPEED

> Speed your shots travel at

Shot Speed in Chroma Mancer determines the velocity at which your projectiles travel, as well as how far they travel. It's a decimal value and the base value is 4.0. Every 16 points into this stat will increase your projectile speed and range by 1.

All projectiles experience gravity fall-off after *x* seconds, so increasing the speed they travel at also effectively increases its range. Therefore, you can use this stat to hit enemies that are farther away and turn your character into a sniper.

6. STAMINA

> How often you can dodge and jump

Stamina in Chroma Mancer is a resource used for Dodging and Backflipping, which recharges automatically 1 point every 2 seconds.

It's represented like Health (see above), using Full Charge (2 points), Half Charge (1 point), and Empty Charge (0 points) icons. The base value is 3 Full Charges (6 Stamina points), and is increased in increments of a Full Charge (just like the Health System). Every 25 points into Stamina grants the player an extra full stamina charge. Additionally, Stamina also recharges faster as you put more points into it, with a maximum value of recharging 1 Stamina point every 0.75 seconds.

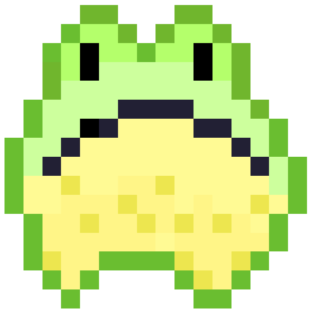
## Enemies

Enemies are a core part of Chroma Mancer, as they provide the Player with a challenge to overcome and a need to level up their characters. Enemies spawn in pre-determined waves (designed and balanced by me) at specific time intervals. There are a total of four unique enemy types that will all challenge the player in different ways, at increasing levels of difficulty. They have several aspects in common, though, such as:

- dealing 1 damage to the player
- dropping 1 Ink Drop (XP) on death
- leaving behind a corpse on the battlefield after death

The four different enemy types are the following:

[continues on the next page]



1. CHASER

♡6 HP

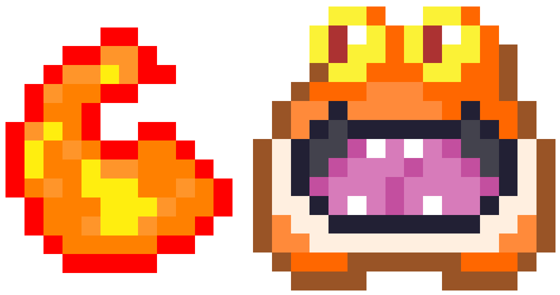
The most basic enemy is the Chaser - a sulky green frog. As the name implies, they chase the player directly and deal damage by coming in contact with the player. However, they are anything but basic. They move with momentum, so there is some delay when adjusting to the player's position to give them a feeling of natural movement. This also makes them vulnerable to sudden direction changes via the backflip, as they will take some time to adjust to the player's new position. Their speed varies according to their distance to the player:

- they are **fast** when **far away** so that the player never completely outruns them
- they are **slower** at **medium range**
- they **greatly speed up** at **close quarters range**

This incentivises players to be strategic and keep them at medium range, since if they get too close, they go mad with bloodlust and become extremely fast. It also encourages players to use the backflip, as that is their biggest weakness.

On top of this, careful but effective randomisation was added to their movement delay and target offset. This randomisation decreases the closer they get to the player, giving a sense that the Chasers are narrowing down on their prey as they get closer. This was deliberately done so that when in a large horde, each individual Chaser moves noticeably uniquely and distinctly from the other Chasers, giving them the impression of being natural organic creatures with their own brain instead of a mindless hivemind horde of drones.

They are best kept at distance, as they become very deadly when they close the gap!



2. SHOOTER

♡3 HP

The next enemy is the Shooter - a vibrant orange frog with a big smile. Behind that mocking smile are warm intentions: he shoots a fireball projectile at the player! As the name implies, this is a ranged enemy that deals damage by shooting fireballs at the player. It is the only ranged enemy in *Chroma Mancer*.

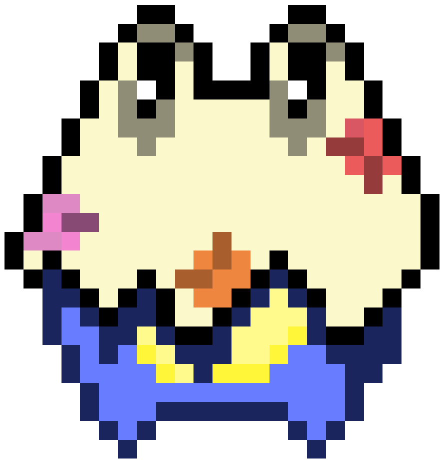
In contrast to the Chaser, the Shooter excels at medium range, where it prefers to stay. Their movement AI is totally different and has the following rules:

- If the player runs away, the Shooter will chase after them but always stay at a safe distance
- If players try to get near them, the Shooter will run away.
- Every few seconds will tactically reposition to a new medium-range position.

Their Shooting AI obeys the following rules:

- Cannot shoot while running away
- Must be stationary to shoot
- Will shoot once ever 5 seconds, if able. If not able, wait 5 seconds and try again.

They are extremely deadly if left unchecked and should be a priority. They can be dealt with by quickly closing the gap by dodging through their fireballs, and quickly dispatching them while they're not able to shoot back. Better yet, invest points into Shot Speed to kill them at long range and beat them at their own game.



3. GHOST

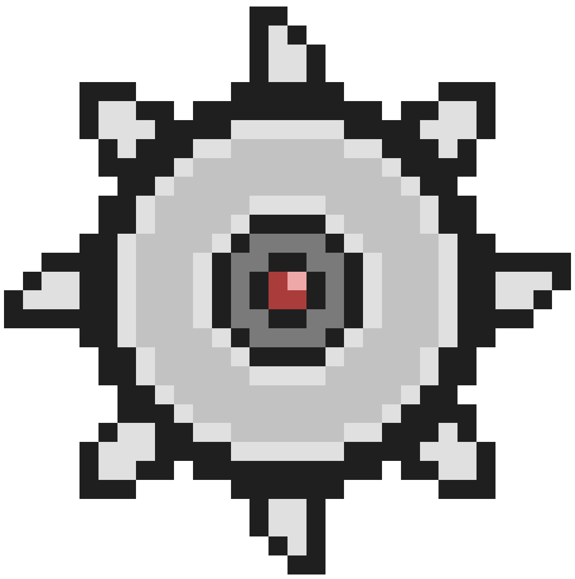
♡10 HP

The Ghost is a Blue Frog with a patched white-ish sheet over their head that floats up and down in the air. It's twice the size of the previous enemies in the game world. It introduces a wholly new movement system that will challenge players:

- instead of moving in the game world, it moves across screen space!

No matter where the player is or goes, the Ghost will linearly move from edge of the screen to another, while gloating up and down in a smooth and spooky sine wave motion. For example, it will move from the left-side of the screen to the right-side of the screen, until it has fully left the screen. After 5 seconds, it will respawn and once again move across the screen, except in a different direction every time. This continues until the Ghost is killed.

This idea came to me because I wanted something totally different to haunt the players, something impossible to outrun or escape, that can only be dodged. Players will need to learn the movement mechanics to successfully avoid and outmanoeuvre this enemy.



4. BOUNCER

♡15 HP

This truly menacing beast is the biggest, hardest, and strongest enemy in the game. It's a giant spinning sawblade of death.

It builds upon the Ghost's movement by also moving across Screen Space instead of World Space, however it never leaves the screen at all until it is killed! The Bouncer slowly fades in and spawns at the center of the screen and then moves in a random direction. Every time it hits a screen edge, it bounces/reflects off of it (kind of like that bouncing DVD logo animation). Like the Ghost, it is impossible to outrun or escape, but due to its relentlessness and huge size, it poses a massive threat to the player and must be dealt with ASAP!

Since they spawn in the middle of the screen and are so dangerous, they are faded out and their collision is disabled when they spawn. They must bounce on the screen 3 times before they fade in and collisions are enabled. This is so that players have time to see what's coming, predict their movement, and prepare for the worst.

If players can reliably deal with this threat, even in multiple numbers, then they will have mastered *Chroma Mancer*!

Part 2:

DEVELOPMENT DOCUMENTATION

My journey though the development of my project ***Chroma Mancer***:

from initial ideas and concepts to their practical application and implementation: covering Game Design, Programming, and UI Design.

TABLE OF CONTENTS

Introduction

It took a long time for me to figure out what I wanted to do for my Bachelor Thesis. We spent most of the 7th semester learning about scientific approaches to writing a thesis and how to formulate a good question, that my head was completely focused on the theoretical part and I could not nail down a topic I wanted to investigate. It was only when I stopped worrying about the research question and starting thinking about *"what project do I want to make? What new kind of game am I interested in creating?"* that I finally made up my mind: **creating a completely moddable game**. The initial idea I locked onto was simply to make some kind of game (genre yet undecided) where players would be able to modify every single sprite in the game: from their character, to the enemies, to the level environment, and even the UI. My long-standing passion for modding was the key driver behind this idea.

However, before I did any practical work, I realised that this simply would not be interesting enough. If it's just a normal game with no need for modding, most players would ignore it or not even notice it was a feature. That would not do: I needed the modding to take center stage and therefore to create an intrinsic need to mod the game, that would be inextricably linked to the core gameplay loop. Therefore, the new goal was: **creating a game that was designed to be modded**. A game you would have to mod in order to progress through the level(s).

I was getting closer to my *Eureka!* moment, but now I had to figure out how to link modding into the core gameplay loop. The first thought that came to mind was creating a puzzle game where modifying elements of the puzzle or the rules of the level would be required in order to solve it - this was obviously inspired by *Baba Is You* (2019), my favourite puzzle game in recent years. However, I couldn't really come up with any concrete examples for puzzles or levels without pretty much making a copy of *Baba Is You*, and I had never designed a puzzle game before, so there would be too many challenges and new ground to cover. I scrapped the puzzle idea. But this got me closer: what *is* a type of game that I have experience in making? **2D fast-paced challenging arcade games**.

Now I set to brainstorming ways in which this genre and my modding gameplay concept could be linked. I quickly set on the main concept of having a survival type shooter set in a big and open level, as this would serve as a perfect playground for players to interact with the modding system and play out their creations. Now all I needed was that modding system... Then it came to me. A modding system where players can not only re-draw every object in the game, but where that object's very attributes are determined by this drawing: **"Inferring Stats From Images"**, as I called it in my brainstorming document. This is word-for-word what I wrote in my original brainstorming session on 30/01/2023:

> Modding Method #1: Inferring Stats From Images

> This could be really cool because it would mix artistic expression with gameplay strategy. Not only will players try to draw an aesthetically pleasing item/character, but they will also have to consider its gameplay implications and try to make wacky colour combinations work. Combining art with strategy.

> How it would work:

> 1. Pixels of a certain colour increase certain attributes. For example, when drawing a weapon/ability/projectile including red pixels would increase the item's damage. More red pixels = more damage. More blue pixels could increase defence, etc. To balance this, sprites will have a fixed maximum image size, so players will have to balance the stats they want within the given limited canvas.

> 2. Even within the fixed canvas size, some pixels could be left transparent, so the engine would count how many coloured pixels there actually are. More pixels = bigger heavier item/character = slower attacking/moving speed. This allows players to opt for slow heavy hitting weapons/characters with huge stats, or simpler but nimbler weapons with less stats but which are more mobile and attack faster.

Thus, the concept for what would become *Chroma Mancer* was born and set in stone. I knew I had finally found it. My *Eureka!* moment had come! I still had to fill in a lot of gaps in the game design of how exactly this system would work, but the core design pillar was here and I was very happy with it. It's a concept I honestly don't think has been tried before: a game where you freely draw your character, whose attributes and stats are directly based on this drawing. Most character creators these days allow great depth in customisation, but almost none of the choices you make while designing your character will impact gameplay significantly, if at all. Therefore, I knew I had something interesting and unique that was worth developing. I started the barebones skeleton of this project in early February, with no idea of what the end result would be.

Scope & Early Design Revisions

You may have noticed in my original idea that I intended for every object to be re-drawable and have moddable stats, however in the final prototype of *Chroma Mancer* only the player character can be drawn and re-drawn and have matching stats. I even wanted to include passive items that players could find and unlock, and then draw to have whatever effect they desired. Thankfully, after an initial mentoring session with my supervisor, Prof. Emmanuel Guardiola, he advised me to focus on *one single vital element*: in this case, the most interesting and prominent element would be the **Player Character**. I early on decided to focus on only allowing the Player Character to be drawn and have its stats changed according to the drawing. Passive items were scrapped entirely and never made it past the concept stage. I had also made a concept for enemies to be drawn by the Player as well. It would follow the same concept of inferring stats from images as the Player Character, but you may ask: *why would anyone make powerful enemies and not just easy-to-defeat weaklings?* Well, my initial idea was that the game would be a highscore chaser, and that enemies would grant you score directly proportional to their strength. So if you drew very weak enemies on purpose, you might have a hard time dying, but you wouldn't be getting very many points, compared to someone who drew challenging enemies. Alas, for the same reason cited above of focusing on a single element of interest, this idea was scrapped and the enemies in *Chroma Mancer* all have specific hard-coded and balanced stats.

LEGACY FEATURES

And yet, my original idea for a "completely moddable game" still lives on in the final version of the Chroma Mancer prototype! Every single sprite in the game, including the enemies, environment details like grass and trees, UI elements like the Hearts and XP icons, and even the colour palette used for Stats **can** be modded and replaced by a drawing of the player's choosing. They just have to make sure that their drawing is the same size (pixel dimensions) and is given the correct game. Then they must place this image in the CUSTOM folder present inside the Game's Data folder. More details on how this works can be found in the upcoming Modding System section.

Programming Vital Systems

1. MODDING SYSTEM

I began my project by creating a simple modding system in which players could replace any sprite that was flagged as moddable. My Mod Manager looks in a specific folder (`[Game Folder]/Data/CUSTOM`) and sees if there are any .PNG files whose name and size matches any of the sprites flagged as moddable. If so, those images are loaded and used in-game in place of the original sprites. This is the traditional way of loading mods, such as the way it is done in Source Engines where I first learn how to mod games, and is the reason why I designed my system in such a way. The system is also able to generate multiple sprites from one spritesheet PNG according to the desired size, which was used for modding the UI icons. This process is explained by a series of "`readme.txt`" files in the `CUSTOM` folder.

Unfortunately, due to this process having to be done via the file system, it's not very user friendly and obvious that it's there, so the majority of players did not make use of it (as far as I know). I had wanted to also make modding accessible from in-game via a UI menu, but due to lack of time and it not being the primary feature of my game or thesis, I decided to scrap that idea.

2. STATS SYSTEM

The next task to take on was creating the stats system that I described above, where the number of pixels of a certain colour equal your stat points in its respective stat. I started by creating an **`Image Analyser`** script that loops over every pixel in the player's avatar and counts the occurrence of each colour. At first, it simply displayed a colour code and its occurrence value. Then I created a **`Stat`** class that would contain the individual information about a single stat - such as its name, colour, and value - and a **`Stats`** component to manage all the stats in one place. With these two parts in place, it was then a matter of seeing whether the colours found in the player's avatar existed in the **`Stats`** list, and if so, set the appropriate stat's value equal to the number of that colour's occurrences. In line with the modding philosophy, the colour representation of each stat is not fixed, and is actually determined by an image file called "`palette.png`" that can be modified by players as mentioned above. Thus, the colour representation of each stat - and thus the colours that the player can unlock and draw with - are determined at runtime and fully moddable.

Initial Design Work

1. OVERALL CONCEPT

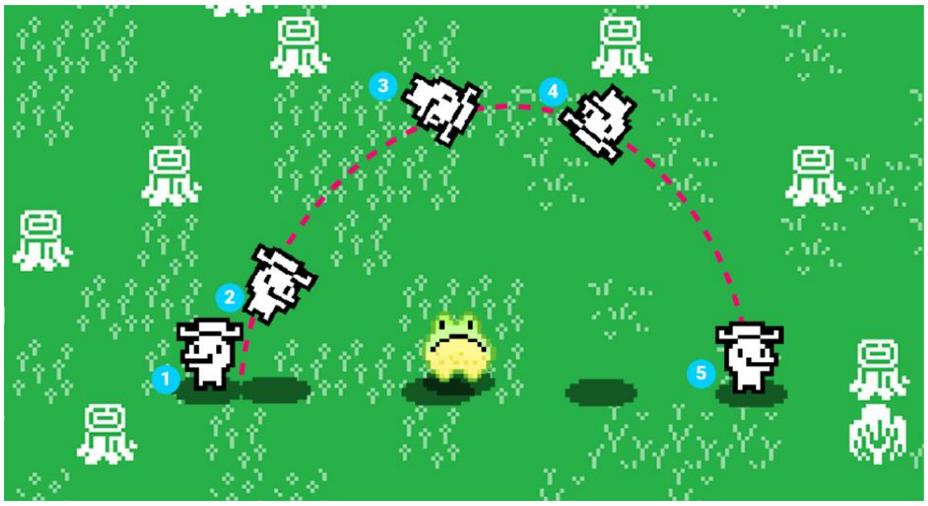
I now had the two most basic fundamental systems working for my concept, but before I could proceed, I needed to define what I wanted the actual game to be. As mentioned before, I wanted to create a *2D fast-paced challenging arcade game.* Since my concept was about **players creating their own character**, I felt it would be a good idea to set the game in a **single large open level** to act as a sort of playground for players' characters and builds. The ideal game to fill this open environment would be a **top-down bullet hell shooter** in my view, as you could fill large areas with enemies and their projectiles, and you'd need a lot of space to run around dodging the projectiles and kiting enemies. However, I didn't want the gameplay to be too hardcore and require precise aiming, so I decided to:

1. use four-directional shooting via the arrow keys, instead of 360º aiming with the mouse;
2. never have a completely overwhelming number of enemies on screen, instead spawning them gradually in small waves. However, it can still get overwhelming if players don't engage and defeat enemies quickly enough, so it motivates players to be aggressive and not simply run away forever.

2. MOVEMENT

Before I designed any enemies or went into more nuance with the shooting system, **I wanted to focus on the movement**. This was unrelated to the concept of character creation, but I always felt like similar games in the genre - like *Binding of Isaac* and *Vampire Survivors* - had very rigid and limited movement. Therefore, to make my game stand out even more and feel more enjoyable, especially since it was to be set in a large level with fast moving enemies, I sought to give players really satisfying movement options to use. The first and obvious candidate is the **dodge roll**, which is becoming ubiquitous thanks to *FromSoftware*'s games, and was also present to some extent in similar games such as *Nuclear Throne* and *Enter the Gungeon*. However, I made my dodge roll longer and faster to feel more responsive. Additionally, the length of the roll *can be increased* if players level up the `Movement Speed` stat.

But a simple dodge would not be enough to be memorable these days, so I needed something more interesting and unusual. Thankfully, I had been playing *Donkey Kong (1994)* for the *GameBoy* so I knew exactly what I had to do: **add a backflip!** For me, it was an extremely satisfying movement mechanic to use that I have not seen anywhere outside of Super Mario games, especially in 2D games. In my game, the backflip would allow players to vault backwards a great distance and thus **jump over any obstacle and enemy**. Since my game is top-down, the backflip can be performed horizontally or vertically, and thus I would need to add some kind of fake-3D to make the movement convincing. By giving the player a shadow that is permanently "grounded" and moving the player's sprite in a **sine wave motion above the shadow**, as well as giving the sprite a rolling animation, I was able to achieve this effect fairly well I believe. Even after the project's conclusion, I still think that the backflip is one of the biggest "*wow factors*" in my game that playtesters loved. Thank you, Donkey Kong for the GameBoy!



Example of the backflip movement

3. THE ENEMIES

Only after perfecting the movement mechanics did I move on to create the meat and bones of the game: the cast of enemies that will face off against the player. I wanted to focus on relatively few enemies on-screen (at least compared to the usual Bullet Hell game that has hundreds) and make each enemy more of a formidable presence and bigger challenge to deal with. Additionally, I wanted to motivate players to use the backflip and reward those that did so, therefore I sought to create fast enemies that were vulnerable to quick direction changes. Again because of this heavy movement focus, I wanted most enemies to be melee, so that players would be mostly dodging enemy creatures instead of a barrage of projectiles. I only created one ranged enemy and my main aims when designing enemies were:

- **fast** movement that is vulnerable to quick direction changes
- interesting and **unique movement** (not just blindly rushing the player)
- **threatening presence** but not overwhelming quantity

With these three pillars in mind, I came up with four unique enemy types that I think work really well together and bring a lot of life and character into my game. I first created the four unique movement types and tested them on a placeholder circle enemy. Only when the movement was completed were the enemy sprites created and their stats balanced accordingly. The Chaser Frog, Shooter Frog, and Ghost Frog sprites were created by my girlfriend Maria Myronenko. Each enemy was balanced to only deal 1 damage to the player per hit, as I believed their inherent threat should come from their complex movement and not from arbitrary abstract damage points. They do however have a varying number of Health Points, between 3 and 15. The health values are all integers so that the number of shots required to defeat an enemy is predictable and consistent.

For more information on each enemy type, consult the **Enemies** section of my **Game Design Document** (printed here before my Development Documentation) where I go into detail on each enemy. I will not do that here again to avoid repeating myself needlessly.

4. SHOOTING

I wanted the game to have four-directional shooting instead of 360º, so implementing the shooting system was rather quick. That is, the barebones version of it, which I wasn't satisfied with. Simply moving the projectiles in a straight line felt wrong when playtesting, as if they weren't affected by gravity or momentum at all. It felt like they lagged behind and didn't go where they were supposed to. I looked to similar games like the Binding of Isaac to see how they addressed this issue in their shooting mechanics.

The solution: to add a **diagonal offset in the direction of movement** that is affected by the duration of the movement. That is to say that when moving **left ←** and shooting **upwards ↑,** these two directions would be combined into a vector going diagonally up-left ↖ (i.e. north-west).

In addition, the amount of diagonal offset is determined by **how long you've been moving** in that direction. For example, when moving up ↑ `(0,1)` and shooting right → `(1,0)`: if you've only been moving for a short amount of time, it will be a minimal offset `(0.2,1)`, but if you've been moving for several seconds then the offset vector will be fully diagonal `(1,1)`. I found that this system really gave the shooting a feeling of weight and momentum, and was an exponential improvement.

Beyond that, I also extended the fake-3D theme to the game's projectiles by giving them a shadow and having them be impacted by **gravity**. After x seconds, the projectile will have a decreasing Y-position and move closer to its shadow, giving the effect of falling towards the ground. I also made a **poof smoke cloud** type effect that is spawned when a projectile falls or hits a target to give them more feedback and **game feel**.

5 ENVIRONMENT AND OBSTACLES

After I had finalised the movement, enemies, and shooting, and playtested everything in the level, I noticed it **felt too open and too empty**. You could dodge away forever in any direction, so there was no incentive to be strategic with your movement and no downside to running away eternally. I had to improve this.

As such, I first created **fixed borders** around the level using **trees**, which allowed the player to travel 50 units in all directions. This was a good compromise between feeling like a large level but still contained in a limited space. Secondly, I added **obstacles**. This are objects you would normally find in this type of forest environment, such as **rocks, tree stumps, and bushes**. Neither the player nor the enemies could move through these obstacles (except for the Ghost and Bouncer), so player now had to **move more strategically** to avoid running into a wall and cornering themselves. However, this also granted them the ability to **lure enemies into a corner** and blast away at them while they are unable to move away. As mentioned in the movement section, the backflip allows players to gracefully vault over any obstacle (except for the border trees), so they always have options to escape. I coloured the obstacles in **full white** so that they would stick out against the background and be immediately noticeable.

Using the same tile-based system I created to randomly spawn different obstacles in different positions every time, I added **environment tiles** such as grass and flowers that are also procedural generated into the world. They are purely visual and anyone can walk over them. I coloured them in a transparent white, so that they **blend in with the background** while still being visible. I decided to make them procedurally generated because it does add more **visual variety** and stops things from always looking identical. Moreover, since the level was still quite vast, it was much easier to populate these environment tiles into the game world via a script than to manually place them one by one in the Unity Editor. *Worker smarter, not harder!*

Story / Tutorial

I intended on having a short intro level, where the game's narrative would be introduced and players would learn the movement mechanics - particularly the backflip. Players were supposed to have wandered through a village and surrounding forest until they found two items: the **Colour Palette** and the **Paintbrush**. This would create **in-world** explanations for why the player has these items when they draw their characters. Then players would be prompted to draw anything they wanted four times: unknowingly to them, they would be **designing the sprites of the enemies** that would later face off in the game. This would put them directly in the shoes of the game's protagonist, as the story that I envisioned revolved around an innocent kid finding Magic Colours and a Paintbrush that could **bring any drawing to life**. Thus, the drawings they had created in their notebook would escape into the real world and turn hostile, stealing the Magic Colours for themselves and attacking the local villagers. This would give the players the **motivation and objective** for **defeating all the enemies**, **recovering the stolen colours**, and returning them to safety. Moreover, a Mysterious Wise Animal (in the form of an Owl because I enjoy *Twin Peaks*) was going to be an NPC that would explain to the player the power of these Magic items, the quest that they must now pursue, as well as granting them the magic ability to summon projectiles in order to defeat the enemies. It might be a simple story, but I quite like because it would give a perfect **explanation for everything** that then takes place in the main gameplay loop. As it stands, the cast of enemies and game objective can feel totally arbitrary, but given this context and story, I think players would clearly understand the reason behind everything and therefore feel more immersed and engaged with the gameplay.

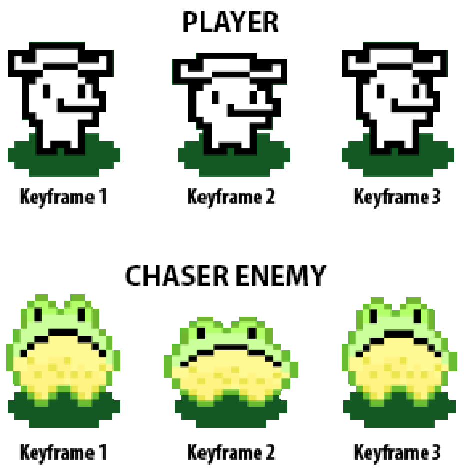
Alas, I very regrettably did not have time to implement this Intro level before sending the game out to playtesters. I had a very basic skeleton sketched out, including the creation of tiles and even houses, but I found myself **spending too much time** on this intro towards the end of the timeline while the game still **needed a lot of polish of finishing touches**, so I had to cancel it for the time being. The houses and environment tiles that were created found their way into the game, however, serving as the starting spawn point for the player and as obstacles to avoid respectively. If I had just a few more weeks to work on my prototype, **this would be the feature that I would go back to and finalise** above all others.

Game Feel

A topic that I'm very interested in is ***"game feel"***, which relates to making games feel juicier and adding more feedback in order to increase player engagement and immersion. In this regard, I am very much inspired by the games and talks of Dutch game designer Jan Willem Nijman, formerly half of *Vlambeer*.

CHARACTER ANIMATION

The first thing I did was creating a basic animation for the player character and the frog enemies. Since players would manually draw their own characters, it would be **impossible** to give them pre-determined animations - unless I forced players to draw multiple frames to create their own animation, but that would certainly be too much work and onus to place on the player. Therefore, I created a dynamic animation that simply **manipulates the existing sprite** based on the principles of animation: **a squash and stretch**. The player character and frog enemies with squash and stretch as though they're bopping or breathing. Playtesters agreed that this added a lot of **life and charm** into the characters.



Main keyframes of the bopping animation

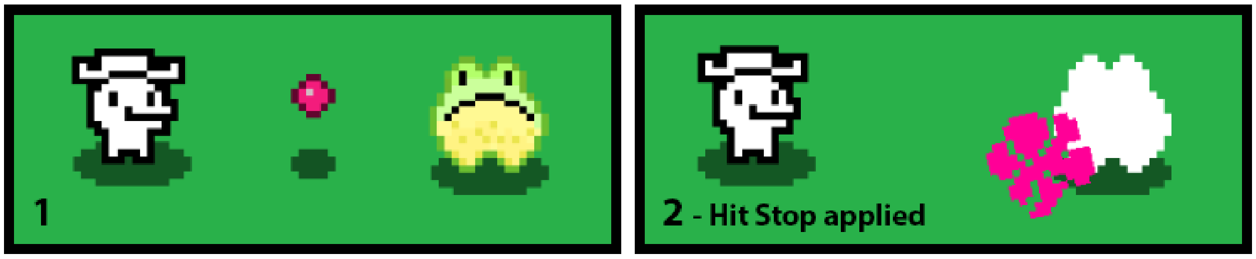
(watch gameplay video or play game for clearer understanding)

SMOOTH CAMERA

The next thing I did was creating a Smooth Camera. This camera follows the player with some slight delay and momentum, to make it feel like it's a **physical camera** doing its best to follow the player's actions, and to give the player an increased sense of speed. On the gameplay side, making the camera lag behind the player allows them to get a better **view of the horde of enemies chasing them**. Another important task for the camera, inspired by one of Jan Willem's talks, is focusing on what the player is aiming/attacking at. To this end, when the player is shooting in a direction, the camera is offset and moves ahead in that direction, so that the player can get a **clear view of that they're aiming at**, at the cost of a lesser view of what's behind them.

HIT STOP

Next important element was a "hit stop" animation. TV Tropes aptly defines hit stop as "freezing or slowing time right at the moment of an impact to create the impression that something **hits harder**". In my game, when either the player or an enemy take damage, their sprites turn full white and the game freezes for 100 milliseconds. This might seem like a tiny detail but it enhances the gameplay feedback a great deal, in my opinion.



Projectile Hit Feedback

KNOCKBACK

In the case of the player, a knockback force is also applied when they take damage, pushing the player away from the damage source. This is for two reasons:

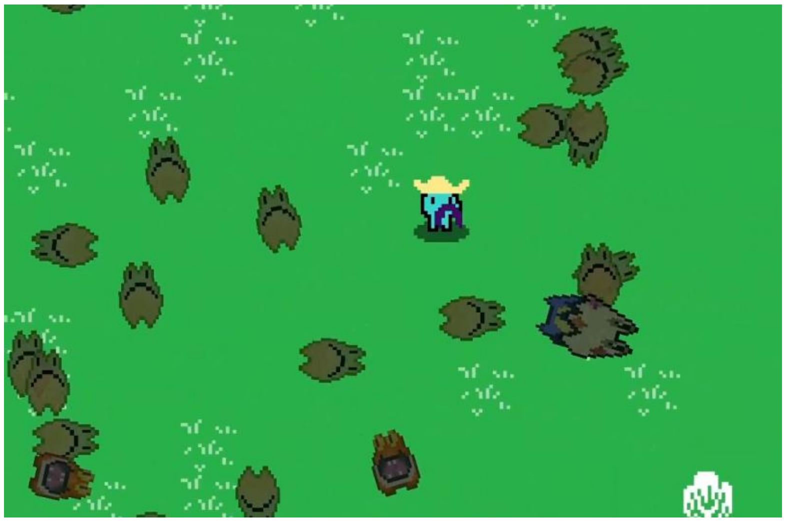
- to give **more impact and feedback** to the act of taking damage
- to push the player **away from danger** and give them an opportunity to get away.

Additionally, the player has a period of **invulnerability** for 2 seconds after taking damage, during which their sprite blinks on and off, and they are immune to damage but can keep moving and attacking.

I had also planned to extend this damage knockback to the enemies, both to make the player's projectile feel more powerful and to allow the player a way of physically pushing back enemies. However, due to every enemy having a unique movement system with their own script, I was not able to quickly implement a knockback that would work for every case. I would similarly have to create a specific knockback method for each movement, so I scrapped that feature for the time being and sadly didn't have time to revisit it.

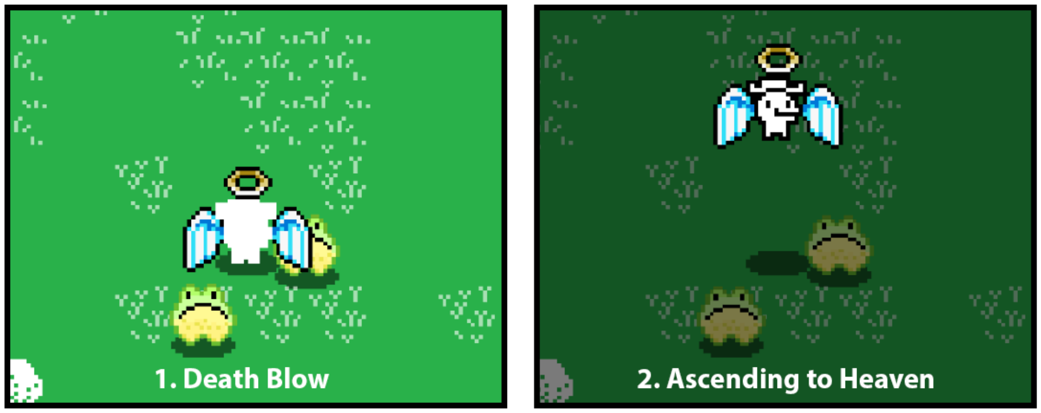
CORPSES

What I instead did for the enemies was making it so that they all **leave a distorted corpse on the battlefield** permanently after being killed. Being a 2D game it doesn't take any processing power to add a sprite to the game world, and it makes the player feel more **powerful and badass** by being able to look back at all the enemies they've defeated and stand atop of a mountain of corpses. Besides being distorted, corpses are also rotated and darkened to clearly differentiate them from still-living enemies. Additionally, for comedic effect, corpses also emanate a "stink cloud" effect when they spawn.


The player standing in a battlefield littered with corpses

PLAYER DEATH ANIMATION

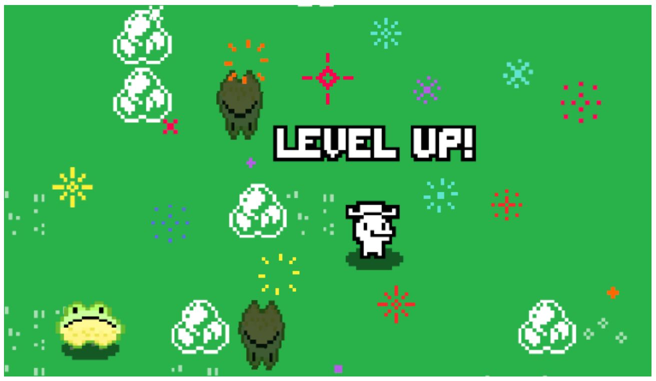
Similarly, I also wanted an **impactful** death animation for the player instead of a plain and boring "game over" screen. I decided to create a dramatic and comedic animation where the player's character **becomes an angel**, gaining angel wings and a halo over their head, and slowly **ascends to heaven** as the game world fades to black behind them. I think it's a pretty neat way to get a good last look at the character you spent so long creating and say your goodbyes. Rest in peace.



Player dying and ascending to heaven animation

LEVEL UP FIREWORKS

Lastly, I created a Level Up animation made up of the text "Level Up!" gradually growing in size above the player's head, and tons of fireworks popping off randomly around the player as they moved around. I created four different fast firework animations instead of just one, so that it wouldn't get samey and boring, and give it a more organic and natural aesthetic. I wanted to add a celebration feeling to levelling up and congratulate the player on their performance, and I thought that this would be a fun and fitting way to do so.



Fireworks celebration (exaggerated for effect)

PIXEL PERFECT RESOLUTION

I was making a pixel art game and I wanted everything to be pixel perfect to get an authentic retro feel and tie everything together. Thus, I had to choose a base resolution to target, that would define the size of the camera/game world. Since I was working with units of 16 pixels and I wanted the resolution to be 16:9 compatible, I arrived at 384x216. A true 16:9 resolution that would scale perfectly on modern monitors, and allowed the game to be clearly visible without being too zoomed in or too zoomed out, and giving everything an authentic retro pixelated feel. In addition to this, I added a Pixel Perfect Camera that players can optionally turn on to actually render the game at 384x216 pixels and then upscale it to their monitor's native resolution. Unfortunately, one side-effect of rendering the game in pixel perfect mode was that the Camera movement is no longer smooth when moving diagonally, it jitters and shakes. I tried my best to fix that, for example by rounding the camera's position to the nearest pixel unit, but the jitters never went away. I wish I had been able to solve it, since I really love the pixelated aesthetic and it really makes everything look more cohesive and gel together, especially when the characters are animated.

Drawing System & UI

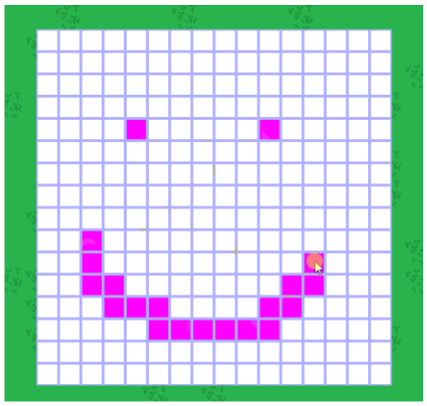
My original idea would be that players would draw their character outside the game, and use the modding system to put it in the game. I even added a feature to allow changing and updating the modded sprites at runtime to accommodate for this. So with this system, every time players levelled up, they would have to edit their drawing outside the game with third-party tools, and the game would automatically update it. In hindsight, just typing it sounds convoluted and ludicrous to say the least. In my defence, I originally thought of this because I didn't think that I would be able to create a sufficiently decent in-game drawing tool; however, I am proud to say that **I did indeed create such a system!**

After creating the movement system, enemies, and shooting, I decided to tackle the challenge of creating **my own pixel art drawing system instead of Unity**. I couldn't find any tutorials online, so it was up to me to design it and build it from scratch.

1. INITIAL PROTOTYPE - THE CANVAS

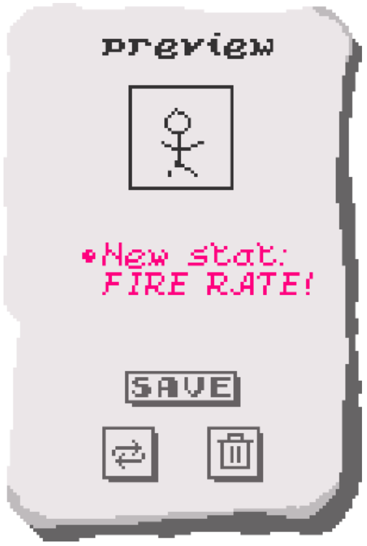
Luckily, my first intuition proved successful. Using the UI Builder, I set up a container that would hold 16x16 buttons. This container would be the canvas and each button would represent a pixel. The buttons would have no styling and be a totally transparent square, until they were clicked: on click, their colour would change to match the currently selected colour. Players would naturally also want to erase things, so a right-click would set a "pixel's" colour back to transparent. To be able to extract this drawing and create a Texture out of it, each button was mapped to a two-dimensional array. When saving the drawing, a texture would be created by iterating over this 2D array and getting the colour value at every pixel coordinate.

Surprisingly, after setting everything up, **it worked** flawlessly: **except it was not enough**. Pixels only reacted to a click, not to a click and hold as you would expect from a drawing program. So if players clicked a pixel and held left-clicked as they drew a long stroke, only the first pixel would be coloured. Therefore, I had to make pixels also react to a "mouse over" event and then check if a mouse button was being pressed - both left and right mouse buttons. After this improvement, **the basic drawing system itself was finalised!** No fancy extras like shape tools or selections were implemented, because I would have no idea how to do so and because they wouldn't be necessary given the small canvas.


The first working prototype drawing canvas

2. THE COLOUR PALETTE

At this moment, there was only one hard-coded colour available for testing, so then I set to making a colour palette. This was again made with buttons which took the shape of ink splashes on a wooden painter's palette. These would be populated at runtime based on the colours that the player had unlocked in the game, and each colour splash would be tinted to reflect its colour. On a click, the currently active colour would be set to that of the splash, and any new pixels painted would be set to that colour.



3. THE PREVIEW SIDEBAR

Great, all the drawing functions were now officially in place. However, players still needed to be able to save the drawing, so I created a paper sidebar with a "SAVE" button - actually styled as a button this time. I also wanted to add a real-world real-time preview of the avatar, so that they could see it exactly as it would like in-game. The canvas was much bigger so the character was overblown, therefore I thought having a to-scale preview without gridlines on the sidebar would be helpful. This was also inspired by the preview window present in Aseprite, the pixel art software that I used to draw all the game's elements.

While the drawing system could be considered finalised, I noticed two things that would be extremely helpful when testing it: a way to undo all changes and a way to clear the entire canvas. Therefore, I set to work and created two buttons that would do just that. The Reset button would reset the drawing to its original state when the Drawing Notebook first opened, and the Clear button would set every pixel to transparent. Both buttons also had to trigger an update on the preview to reflect this change.

4. STATS AND TUTORIAL

Another vital piece of information to show the player is the Stat points that they would get based on their current drawing - i.e. the number of pixels coloured in each Stat's colour. To that end, I created an icon to represent each Stat and display a number next to it representing the attributes points they would get in that stat - both of which are coloured in the Stat's colour. I experimented with various designs, such as whether to include it in Sidebar or on the Notebook, and whether to write the stat's name or use icons. Ultimately, icons were needed because since I was working in a pixel perfect environment, I quite simply did not have enough space to write the full stat names - only vague abbreviations that playtesters said they couldn't decipher. Since I still had some space on the margins of the Notebook, I settled with displaying the Stats there using icons. I briefly experimented with 8x8 icons but they were too vague and small, so I settled with 16x16 icons and a 3-digit number to display all Stats and their respective values.

The final piece of information to display was the controls to teach players how to use the drawing canvas. There were only two actions - draw and erase - so I wrote that alongside an icon highlighting which mouse button to press on the left margin of the Notebook. The other actions, such as selecting a colour and saving the drawing, were perfectly intuitive and playtesters ever had any questions regarding them, so I didn't need to explain that.

5. THE CURSOR

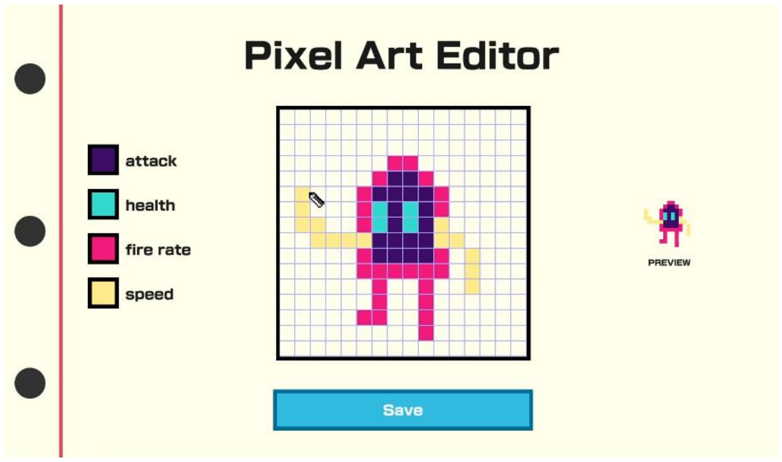
You might think that finally everything is finished, but unfortunately there was something bugging me. I had set the mouse cursor to be a paintbrush when in the Drawing Notebook, but it was always black and white. As such, there was no indication of what the currently selected colour was until the player painted something on the canvas. My UI idea for this was to tint the Paintbrush cursor in the current colour, which would make it clear as the cursor is fairly big. Alas, there's no way in Unity to tint a mouse cursor like you can do with a sprite. I had to get creative...

I knew what to do, but I didn't know if I had the strength to do it. Since a cursor is treated as a Texture in Unity, the only way to achieve this would be to create an entirely new cursor with the desired colour. And so I created a Cursors static class that would manage the cursor and do just that. I had already written code that would analyse images and create textures, so I could quickly repurpose my earlier work to create a method that would take an existing texture, copy it to a new texture, and set every white pixel to a specific desired colour. And that is how the Paintbrush works in my Drawing Notebook. Every time you click a new colour, a new paintbrush is born!

UI Design

Once it became apparent that I was capable of creating a pixel art drawing program, my creative mind instantly become focused on "**what should it look like?** what should the interface be?". Obviously, the initial interface was extremely barebones and ugly, as I just wanted to test if my concept would work at all. I started working on the Drawing UI after implementing everything up to the Save button and real-scale preview of the avatar mentioned above in Section 4.3. The drawing system's features were only finalised after all the UI was in place.

My mind immediately thought of a **diegetic UI**. I didn't want an abstract look with computer menus, but instead a UI that would be entirely based on **realistic real-world counterparts**. I felt that it would make the drawing system **more intuitive and easier to understand**, as well as add some more charm to the game: everybody knows what a colour palette is and what a notebook is, I wouldn't have to explain that. Additionally, it's something that the player character would be carrying themselves in the game world and would **tie perfectly into my game's theme and story**, adding to the immersion. In this regard, I have to say I was inspired by *Metro Exodus*, which features amazing diegetic UI and every "menu" is an object being carried and operated by the player character.



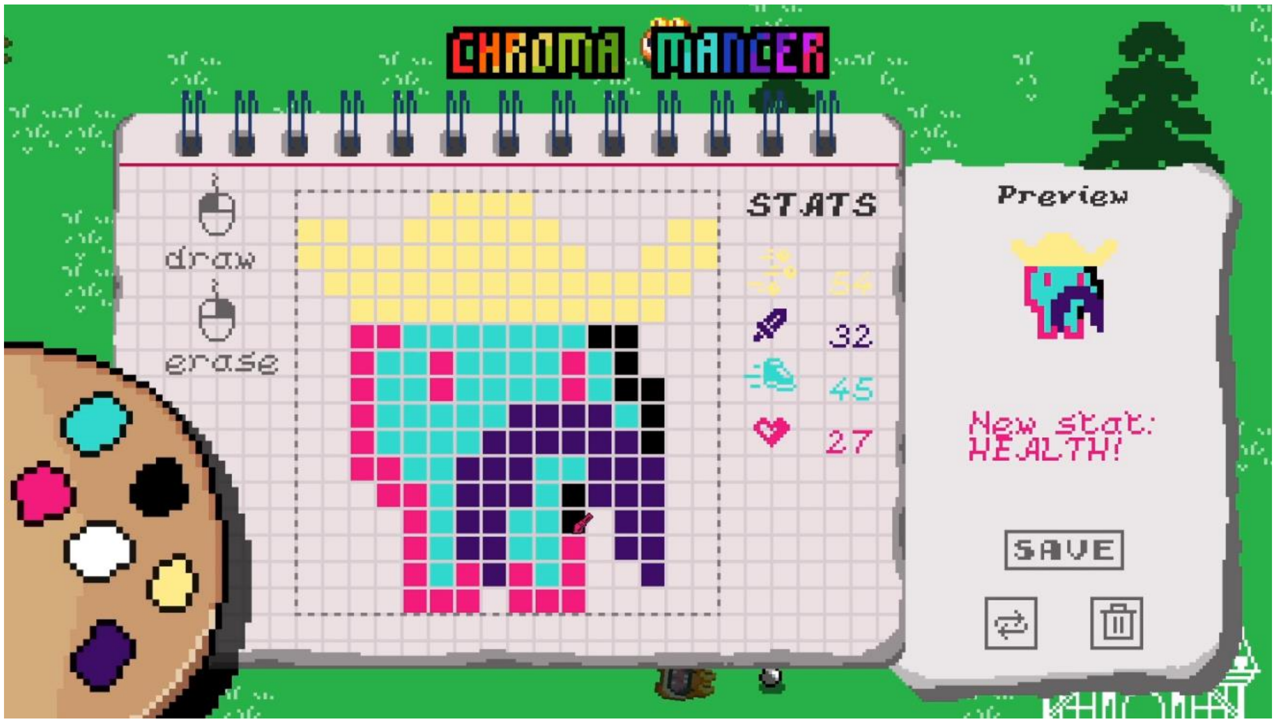
Second iteration of the Drawing UI design

My concept from the get-go therefore was to have the drawing take place on a **squared notebook** - the squares would serve as just perfect ways to integrate pixel art into a piece of paper as well as gridlines to separate each pixel clearly: *two birds in one stone, and it looks authentic!*

For the colour selection, a **wooden painter's palette** was the clear candidate: I have one in my bedroom that I have used extensively, so I didn't need to think twice here. Lastly, I needed some kind of sidebar to house the UI buttons (save, reset, clear) as well as the game world scale preview of the avatar. I thought of a simple **piece of scrap of paper**, as it wouldn't need be very big to contain all those elements, and again it's something that the player character could believably be carrying to quickly jot down notes.

The overall concept was now defined, so it was time to execute it in practice. I drew all the UI myself in **Aseprite**, an amazing pixel art program, and everything was drawn pixel-perfect according to the retro resolution of the game mentioned before `(384x216)`. I have a great pet peeve for "pixel art" indie games that use high resolution UIs and thus break the entire design core and immersion of the game, so I vowed to be true to my chosen resolution. Also, if players used the Pixel Perfect Camera option, the game will actually render at 384x216, so anything bigger than that would be downscaled and look horrible.

It took a few tries and plenty of drawing to arrive at the final version, but by trying to be **as authentic as possible to the real-world inspirations**, I managed to arrive at a UI that I'm very proud of. Everything looks clear but rugged as though it's experienced plenty of use, and by overlaying all these different elements, it adds a sense of physicality and 3-dimensions into my 2D pixel art game!



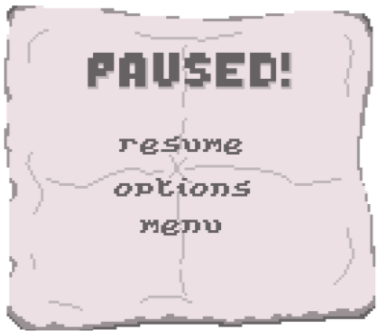
Final iteration of the Drawing UI design!

UI Menus & Polishing

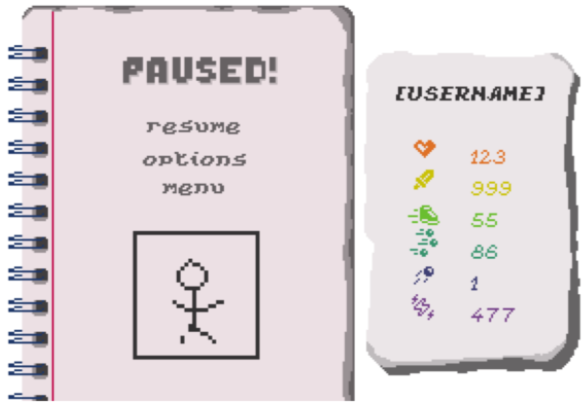
All my time spent iterating and working on the Drawing Notebook UI paid dividends because I also unknowingly **created a UI identity** that the rest of the game would follow. Most other menus are based around the same Notebook created for the drawing system with some minimal changes, such as making it portrait and removing the squared paper, and every menu is based around the diegetic scrap paper design philosophy. I was therefore able to relatively quickly create multiple menus that all have a cohesive design and fit seamlessly together.

PAUSE MENU

To have the game accessible to playtesters, I would need to create a series of menus. The most obvious being a Main Menu and Pause Menu. I had already done the functionality for the Pause Menu, so I started there. My first concept was just a compact scrap of paper with a few text options, but when I did an alternate version using the Notebook, it felt much better and more immersive. The bigger size allowed me to display extra flavour such as an up-scaled version of the player's avatar. I also made it display the player's name and stats on a paper sidebar similar to the Drawing UI.

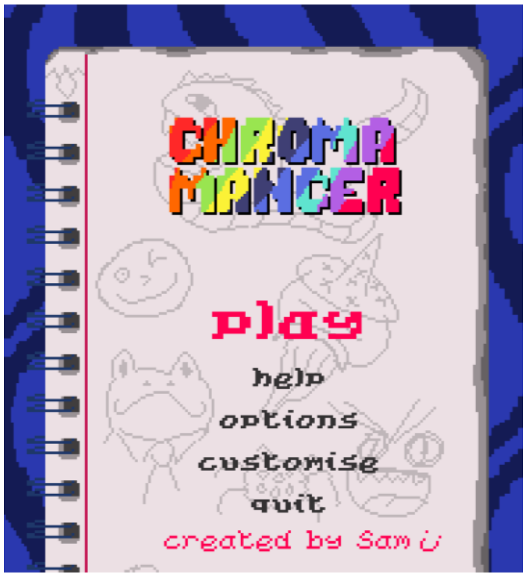


First Pause Menu design

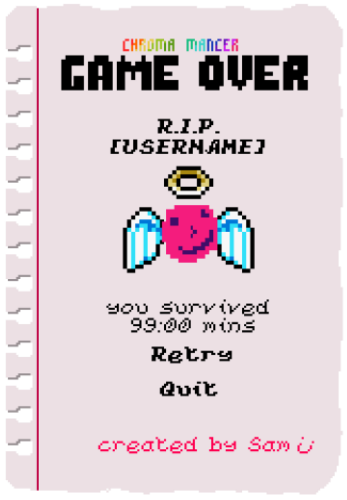


Final Pause Menu design

MAIN MENU



Next up, the **Main Menu**. It contains such options: play, help, options, customise, quit. It features the Notebook again but this time it's filled with doodles to **immediately show the player that this game is about drawing**. I also created a curtain-like background whose colour slowly transitions across the entire hue spectrum, to similarly **show that colours are an important part of Chroma Mancer**. I decided to finally create a logo for the game in order to showcase it here instead of just a bland title. It's the title CHROMA MANCER in big bold letters overlaid with a rainbow-like pattern, to once again show the importance of colour in my game.

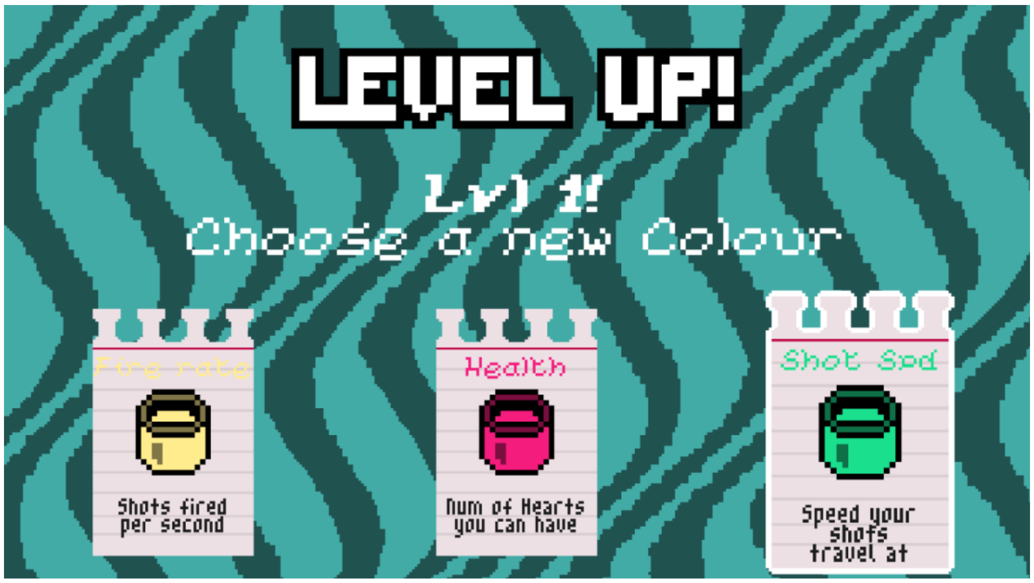


GAME OVER

Then I finally tackled a menu that was glaringly missing from the game: a Game Over menu. Before this, playtesters had to use the pause menu and restart after dying, so it was time to fix that. It's still based on the Notebook theme but slightly differently as I wanted to give it an obituary feel. It's a page torn out of the notebook with a big portrait of the player's character, complete with their newly-given angel wings and halo to emphasise the death aspect and give it a more light-hearted tone, with the heading "R.I.P. [player username]", and text showing how long they survived (in minutes and seconds). I wanted to give it a rough crumpled up paper look, but I couldn't get it right without distracting from the text displayed on the menu, so it's unfortunately a pretty clean piece of paper besides being torn from the notebook.

LEVEL UP

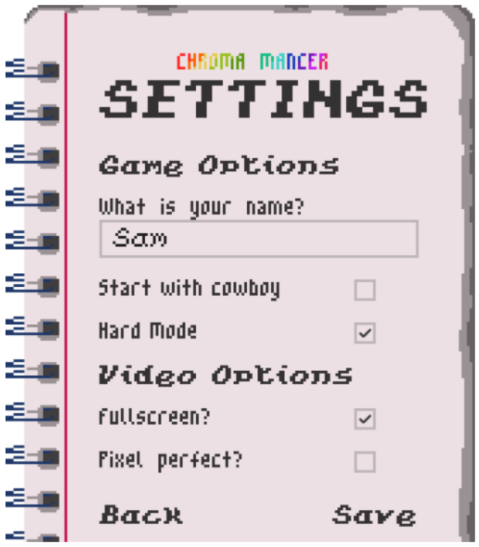
Another menu that was glaringly missing was a Level Up. Until this point, I had used a hardcoded progression of colours when levelling up to make testing faster, but it was time to finally make a menu when players could choose a new Colour/Stat to unlock. This is the most different of all the menus since it departs from the Notebook aesthetic and is a bit more abstract and flashier. I wanted it to be “in your face” and not so subtle as the other menus. However, it still fits the theme by using little notepad pages to display each Colour option, as well as representing each Colour with an authentic bucket of paint. It also includes the same colourful curtain as the Main Menu. Due to the pixel size constraints, it was a challenge to include all the relevant information. I had to use the small pixel font I could possibly find in order to fit even the summarised stat descriptions you see in each choice.



The Level Up menu

OPTIONS MENU

Following that was the Options Menu. Being a long-time PC gamer, having an options menu is vital to me even in a mere prototype. To speed things up, and since I had established a consistent UI theme, I created this menu entirely in Unity without drawing it in Aseprite first.


CHROMA MANCER
SETTINGS
Game Options
What is your name?
Sam
Start with cowboy ☐
Hard Mode ☑
Video Options
Fullscreen? ☑
Pixel perfect? ☐
Back Save

Due to deadlines I wasn't able to include as many options as I wanted, but I created some basic Gameplay and Video options. For gameplay, I allowed players to change their username, start with a default character if they don't want to draw something from scratch every time, and enable hard mode. Hard mode was done to try to cope with balancing. See the Balancing section for more information. As for Video options, I added a toggle for Fullscreen or Windowed mode, as well as a toggle to enable/disable Pixel Perfect Camera mode. I would have liked to add the option of choosing a specific resolution rather than forcing native monitor resolution, but I simply did not have the time to implement that.

HELP

The final thing left to do was to inform the players of the game's controls and objectives. So I created a Help menu, which features a 2-page open-book layout. On the left page, it explains the controls for moving, shooting, dodging and backflipping; on the right page, it displays the objectives and gameplay loop. This menu can be accessed from the Main Menu and Pause Menu. Additionally, I also put the infographic explaining the controls directly in the game world below where players spawn, so that they immediately understand how to play without having to navigate menus. The key combination for performing a backflip is a bit nuanced, making it hard to explain it clearly in a minimalist style, so I noticed that some playtesters did not understand how it worked until I explained them myself.

That was the last menu I created. I also added a "Customise" option to the Main Menu because I wanted to create a customisation menu, where players could re-draw and modify the Enemy sprites and UI Icons in-game, as well as change the colour palette and level background colour to their liking. This would be a much more accessible way to use the modding feature than by interacting with the file system. However, this would be the most complex menu of all, so I sadly had to cancel it. Even just the option to change colour palette I could not implement sufficiently quickly, because Unity's UI Builder does not have a default colour picker UI component. I would have to create my own from scratch, and at this late stage in the project timeline, there simply was no time left. Regardless, I still left that option in the Main Menu to remind myself to implement it when I get back to working on Chroma Mancer!

Player Data

For my thesis, I was interested in analysing the character drawings that people would create in my game, so I needed a convenient way to receive them. For this, I created two methods.

The first method is extremely simple. Every time you save a character in-game, it gets saved as a PNG image inside the gamer folder. The name of this image contains the player's username, a unique identifier, the character's level, as well as a timestamp of when it was created. This gives me all the information I need to organise the drawings efficiently and match them to their respective author in the feedback survey - this is why I ask for people's username in the survey. Then, I kindly asked playtesters to send these drawings to me manually, via Discord or email for example. Most of them thankfully did so; however, as I had predicted, a decent number of playtesters did not (for reasons unknown to me).

This is where my second method comes in. I needed a way to have the game automatically send these drawings to me. I unfortunately have no experience in game analytics, but thankfully have a lot of experience in Web Development. Using my knowledge in this field, I was able to create a self-hosted online database (using PocketBase) as well as a Web API (using SvelteKit) capable of submitting data to this database, also self-hosted. I got the basic prototype working fairly quickly, and then I just had to learn how to call this Web API inside of a Unity game, which took only a couple of days. Now I had two ways of collecting player drawings working simultaneously, and to my surprise, the Web API worked flawlessly and I received every drawing that every player made on there. I even received drawings of players that did not answer the survey (which were ignored for my thesis analysis). There was one single case of a playtester whose drawings I do not have, because they played the game before the Web API was online and they deleted the game before they were able to send the drawings to me.

All in all, I would say that the combination of both methods was extremely effective and I am very pleased with the results.

Conclusion / Post-Mortem

I realise that this has become an extensive document, but in all honesty, I took on an extensive number of tasks when creating this game all by myself that I felt deserved to be addressed and discussed. With all that being said, I am extremely happy with how my game turned out and very delighted and humbled by the overwhelmingly positive response from players. From the people that I watched play, all of them were engaged in creating their character and became very attached to it, even feeling extra sad when they lost them. One thing that that stood out to me was watching my stoic friend Joey who has not much of an inkling for the arts play and draw animals. He is the person I would have most expected to just make random scribbles for stats, and yet he was actually taking the time to create figurative creatures and getting attached to them: "I actually liked the bird, goddamn", he said genuinely, after dying with his bird character.

To summarise, I am very happy and proud of the game I created. If I had to do it all over again, I would not do anything differently – except perhaps have a greater focus on balancing and playtesting early on with various different playtesters. All in all, this game concept has a lot of merit and potential, and therefore I fully intend to continue working on it and turn it into a fully-fledged game after I graduate!

Statement of Originality

This is to certify that the content of this project, documentation and thesis is my own work. It has not been submitted for any other degree or other purposes. I certify that the intellectual content of my submission is the product of my own work and that all the assistance received in preparing it as well as all sources used have been properly acknowledged.