



We Move People

Module 2

Brief

Our second task was a group project, in that we would all fully ideate, iterate, and create a 2D prototype game in Unity. In 2 months, this is a daunting task, however one I am willing to take on to push my boundaries, improve my teambuilding, and garner a greater understanding of player's needs out of game mechanisms. By the end, I should not only have built a game and learnt to code in Unity, but also what game mechanics bring to the table, how I can subvert them, and how I can make a mechanic intriguing to build a premise off it.

Subvert the norms by reimagining a classic video game mechanic, to explore the communicative power of movement and interaction.

What is subversion? What is considered the “norm” for an industry like gaming?

What are some examples of video game mechanics? How have they evolved over the years with technology?

How does movement and interaction give information to the player? How do I communicate effectively without making the setup too complex?

My group – group 1 – consists of myself, Esi Amoah, Guy Porat, Joe Roberts, and Olivia Briginshaw. Before anything in a videogame – before story, before aesthetic, before characters – the way you play it has to be considered. The mechanics of a game is what set games apart from, say, a visual novel or a movie, and as such I believe they are the most important part. After all, if players aren't engaged, is it a game at all?

Deliverables:

- One game prototype, built in Unity.
- One portfolio of work, detailing ideation to creation.

Roles in the Game Industry

As my group have to find where we're going to fit in to make our prototype, I'm going to look at industry roles, not only to help with the project, but to broaden my knowledge of the industry itself. After all, if it's something I am going to be a part of one day, I ought to know how it runs.

Pre-Production

Pre-production is the process of deciding what the game is going to be about and how it's going to be made. It contains jobs such as concept artists, script writers, financial advisors, game runners, and game devs/designers. These first few people will help make a prototype for the game, and decide if there's a place in the market for it.

Production

Production is the part where the game gets made. It is the largest and most time-consuming part, where the initial ideas of the game are either dropped or carried forwards, depending on factors such as time, money, and resources, such as programmers, level designers, 3D modellers and riggers, animators, FX artists, sound designers, and testers. By the end of this stage the game should run, and be ready for the public.

Post-Production

Post-production happens after the game comes out. Despite being finished, there will still be bugs to fix and patches to be made, especially in larger games. Game devs are kept on to fix these problems, as well as meetings held to decide what worked and what didn't. All design documents, assets, and code will be stored, in case they are needed again.

Production Department

Will organise their production and ensure goals are kept within resources and time constraints. Roles include: assistant game producer, community manager, games producer, games publisher, marketing executive, etc.

Art Department

In charge of creating the 2D and 3D assets, characters, and environments. Roles include: 3D modelling artist, concept artist, environment artist, etc.

Programming Department

Creates the game engine, game AI, and code. Roles include: AI programmer, engine programmer, gameplay programmer, generalist programmer, physics programmer, network programmer, etc.

Animation Department

Responsible for the movement of any kinetic asset. Roles include: animator, technical animator, rigger, etc.

Design Department

Separate but related to the art department, the design team dictates the aesthetic of the game, and determines how it will play. Roles include: gameplay designer, lead game designer, level designer, user experience designer, writer, etc.

Audio Department

Manages background music, voice-acting, and sound effects for the game. Roles include: audio programmer, music composer, sound designer, etc.

Technical Art Department

Bridges the gap between programmers and artists. Roles include: graphics programmer, technical artist, video effects artist, etc.

Quality Assurance

Manages the final quality of the game to ensure there are no glitches or bugs. Roles include: games tester, quality assurance build engineer, etc.

What is a Mechanic?

In videogames, mechanics are the way in which you control the player. They are an action or set of actions for the player to perform. In this sense I think mechanics are the most important aspect of a game, because they serve the most vital role: how you play. If you're not really playing, what's to differentiate it between a visual novel, or a film? Creating engaging mechanics should be step 1 of making a videogame, always, and has been a rule since the arcade days of gaming.

Verbs: what the player is doing. Eg: Explore, destroy, build, survive, steal

Rules: why the player is doing it. Eg: rock beats scissors.

Mechanics: how the player does it. Eg: walk, dialogue, shoot.

Standard videogame mechanics have been so long unchanged that there's almost a specific language around them now. Most seasoned players don't need to be told to press A to jump; they know it instinctively. These skills are often transferable by genre. After all, there isn't too much difference from one shooter game to another, so if you're good at Overwatch for example, then chances are you'll be able to pick up Valorant, or Apex Legends, and have mastered it in a short amount of time. This "language" is entirely based around mechanics, and is rooted in how they've evolved over the years.

Sicart, M. (2008). Defining Game Mechanics. *Game Studies*, [online] 8(2). Available at: <https://gamestudies.org/0802/articles/sicart>.

Evolution of Game Mechanics

Mechanics have been often held back by limitations in soft/hardware, which is why the earliest videogames such as Pong and Tetris had such simple mechanics. The franchise of Super Mario games all stemmed from the 1983 platformer Mario Bros, where its scope was limited by what arcade machines could do. As technology evolved, mechanics were allowed to become more elaborate, along with the graphics. This is best demonstrated in fighting games. Street Fighter (1987) is simple, with a short range of buttons for your moveset (both fighting and dodging) and a healthbar. Compare that to Super Smash Bros: Brawl (2008), which has a wider range of movesets, the "shield" button, a special attack to obliterate enemies, and a whole lineup of objects to use against each other such as clubs and power-ups.

The genre had evolved – and so did the mechanics.

Hierarchy of Game Mechanics

Not all game mechanics are created equally; indeed there is a hierarchy of importance based on sustaining a player's interest and motivation.

1. Core Gameplay and Core Game Mechanics
 - a. The set of activities the player will undertake most frequently, and is critical to getting to the win condition.
2. Core Meta-Gameplay
 - a. New activities that share with the core gameplay mechanics; or core game mechanics with new semantics. This may not go towards the win state of the game, but will enrich the payer experience.
3. Satellite Game Mechanics
 - a. Mechanics that enhance already-existing activities, either by adding new features to an existing mechanic, or by modifying an existing features. This also enhances player experience.

Fabricatore, C. (2007). *GAMEPLAY AND GAME MECHANICS DESIGN GAMEPLAY AND GAME MECHANICS DESIGN: A KEY TO QUALITY IN VIDEOGAMES GAMEPLAY AND GAME MECHANICS DESIGN CONTENTS*. [online] Available at: <https://www.oecd.org/education/ceeri/39414829.pdf>.

In videogames, mechanics are the way in which you control the player. They are an action or set of actions for the player to perform. I think mechanics are the most important aspect of a game, because they serve the most vital role: how you play. If you're not really playing, what's to differentiate it between a visual novel, or a film? Creating engaging mechanics should be step 1 of making a videogame.

Different Mechanics in videogames

- Shoot
- Jump
- Run
- Survive
- Craft

What is Subversion?

Subversion is the act of taking one's expectations and giving an outcome that wasn't anticipated. So to understand what would be considered "subversive" to the world of videogames, we have to first establish what is "normal". For such a massive industry as games design, you'd think there would be an abundance of creativity – and while there are many subversive and imaginative games out there – most videogames are made by companies, who run for profit, not art. What sells well for one game will be copy-pasted to the next dozen copycats until the market is oversaturated, and very few bother to change much but aesthetic.

AAA Games

AAA games are from the biggest companies in gaming – they are huge teams of people, but they don't often like to take risks. There is little innovation between minor tweaks in these games, especially mechanic-wise. Think of most shooter games released by AAA studios. They all contain roughly the same goal and rules: shoot others, don't get shot, heal, and reload. Perhaps different characters will have different abilities, but no shooter game has done anything truly unique and subversive for years, not even so much in style (realism reliably sells, so realistic games are what's made). And since AAA studios often set the standard for video games, their riskless approach to innovation sets the norms for video games.

Mobile Games

Mobile games could have been a huge forward-step in the accessibility of videogames. A phone is necessary to modern life, and in turn that means people could have access to thousands and thousands of games in the app store. However, while there were some hugely influential games back in the late 2000s to early 2010's (Jelly Car (2007), Temple Run (2011), and Candy Crush (2012) call to mind), mobile games have largely fallen out of fashion due to the complete oversaturation of the most popular games. There are an overabundance of lazy match-3 games trying to profit off the popularity of Candy Crush; Temple Run similarly spawned a wave of endless runners that have a similar look and style. And if it didn't create a lot of copycat games, then the franchise itself discontinued, or became unplayable. Jelly Car and its sequels were discontinued and removed from all stores in 2014; Candy Crush has got so many new features from competitive ranking to "tickets" to the everyday spin of the wheel you have to click through five popups to play the main part of the game; and one of my favourite games as a child, Angry Birds Space (2012), has been completely discontinued and removed too – and as insult to injury, Angry Birds 2 only allows 5 lives, and is similarly riddled with ads and payments. Mobile games are now associated with ads and cheap graphics and profit. Most people I know only have one game on their phone, that they've either hung onto from nostalgia or has somehow not been tempted to go down the route of quick profits and adspace.

Historical Evolution

There are also a lot of "norms" in videogames that go back a few decades. Platformers, for example, came from Donkey Kong (1981) during the Golden Age of videogaming when they were run in arcade machines; as such the technology was limited. As time has gone on there have been new innovations to the genre of platforming, but the essence will always go back to its arcade roots. There are only so many ways to innovate someone running and jumping. Similarly, hardware for videogames has also reached an expected standard: two joysticks; two triggers on the top; a D-Pad; a quartet of buttons – all of these elements feature in the Playstation, Xbox, and Nintendo Switch controllers, which are the most popular modern consoles. One of my university tutors, Dr. Adam Proctor, recently took us on a trip to an art exhibit in the Nutshell in Winchester, and showed us a 3D printed console similar to a Gameboy – except with a tiny crank on the side. A crank, for a videogame! The idea delighted me. It felt new. It felt subversive. It felt exciting to interact with.

Videogame Subversion

In my opinion, to be doing something truly new with the videogame medium, you have to subvert the mechanics. They are what you control the lens of games with; they are how you interact, play, explore – everything. You can write an incredibly subversive story, but if it's in a simple choice-based videogame format, then only the story will be subversive – not the ludic element, the element from which the game is born. This can be done either through the software, doing something new with the way players interact with the screen, or hardware, doing something new with the controls themselves.

Mechanics in Storytelling

In my opinion, a good mechanic makes or breaks storytelling, and not just in videogames. My favourite novels have had an interesting plot device – or mechanic – to serve the narrative and tie into thematics. These mechanics make books interesting, unique, and above all, engaging. What I hate more than anything is mechanic or a system that's been done before – because then even if it's well-written, it's forgettable. As someone who doesn't play a lot of videogames, I can still recognise the importance of a good mechanic, and have tried to break down some of my favourites below.

Babel, or the Necessity of Violence: An Arcane History of the Oxford Translator's Revolution by R.F. Kuang

This is R.F. Kuang's best book to date, highlighting the impact and the exploitation of immigrants and colonies in the British Empire, with the most unique magic system I've ever read about that perfectly encapsulates the themes. This system – or this *mechanic* – manages to not only engage the reader, but also break down and explain etymology in a palatable way, make debates on the importance of translation types, and all of it ties back to the theme of communication as a form of connection, or as a form of violence. It was not the characters, or the world, or the plotline that engaged me the most, but the silver-working; the unique mechanics of the magic system underpinning the entire book that set it apart from the rest.

S. by Doug Dorst and J.J. Abrams

Mechanics in novels do not always have to be through the narrative itself. S. by Doug Dorst and J.J. Abrams is a book that's unique through the way you read it. Initially disguised to look like a library book, when you open it there are actually two stories: one through the main body of text, the other through the meticulous annotations by two scholars trying to solve the mystery of the author. You have to distinguish what annotations came first and from who via colour and handwriting types, as well as sift through the handwritten notes they slot in the pages of the book. This in itself is a type of mechanic, and made the book a lot more unique and interesting to read!

House of Leaves by Mark Z. Danielewski

Initially starting as a regular book, you read about the protagonist moving into a new apartment, finding a book left by the deceased man who lived there previously. The book is about a documentary featuring people who lived in a house that find doors appear suddenly, or that their house has dimensions that doesn't quite make sense. Then he begins to notice his own apartment doing the same. And slowly, slowly, the text in House of Leaves begins to warp and wrap on itself, mimicking the nonsensical layout of the apartment. At first you brush it off as a printing error, until they words themselves are spiralling, overlapping – a very unique mechanic that ties itself to the theme well.

The Music Box Paradox

The Music Box Paradox is a novel series of my own creation, built around the mechanic of time-travel, that's intended to be read two different ways to come to two different conclusions depending on reading order. The protagonists of the two books, prequel and sequel, are cursed with prophecy: an affliction where people can “jump” a family generation into the next person that bears the curse. Throughout the novels they are constantly swapping with each other, two separate sides of the same story passed down through history. This mechanic is unique, because it not only transcends the internal time of the book, but also the physical novels. It also gives into the theme of generational trauma, cycles, and transgenderism – and in my opinion more engaging than the tired idea of prophetic dreams.

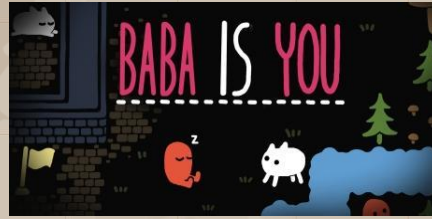
Generally I have always made my mechanics first, because engaging the player/reader is the best way to make them care about your story, and thereby make your product successful. However while a mechanic is easy to slot into a book, I've found it's a lot harder to apply it to games. Not only is there a layer of physical activity you have to account for, but subverting a specific action in an industry as enormous as videogame production is difficult in itself. This project is going to make a very unique challenge!

Subversive Mechanics in Videogames

Admittedly, videogames don't really hold my attention unless I think there's something new in it. You could have made the most wonderful, soul-shattering piece of art, with stunning visuals and a narrative that won the Nibbies despite being a videogame, but if it didn't contain something new mechanic-wise, then I probably got bored and moved on very quickly. Most modern Mario games feel exactly the same to me, and I only really keep up with most new Pokemon installments out of nostalgia-influenced obligation. I've come to learn not to expect too much from AAA studios whose first priority is profit. But below are three beautiful games with such interesting mechanics I just couldn't put them down.



VVVVVV is a puzzle game with a subversive take on the word “jump”. Instead of hopping into the air and coming back down, jumping will completely change your gravity. Being a character who must cross a lot of dangerous spike pits, this makes the game understandably hard, and often frustrating because it is easy to forget at first about the jump mechanic. Jumping has become so ingrained into most casual players as something easy, something default, that changing this default throws them through a loop. You can't just leap over a pit – you have to find a spot on the ceiling to land on, to then come back down.



Baba is You is an indie puzzle game officially released in 2019, however was originally a demo for a videogame jam in 2017. This is why the game is so sweet, simple, and fun, with an emphasis on mechanics over story or graphics. Baba is You gives the player the freedom to completely change the rules of the game while also giving a clear “win” state – a difficult balance to achieve. You, as Baba, move blocks with words around that will define the mechanics. If the “Baba is You” blocks are lined up, you will be Baba: a tiny, bunny-esque creature. However, if you change “Baba” to “Rock”, then “Rock is You”, and you will no longer control Baba; you will control the rock. You can change the win state to something you think will be easier (eg: Flag is Win to Baba is Win) – and you can even reach a game over if you accidentally push a block, and make it so you control nothing. The game and puzzle mechanic are totally unique and refreshing to play.



Portal is definitely the most famous subversive puzzle game, pioneering a mechanic that had never before been seen and twisting it into a slow buildup of increasingly more difficult puzzles. Even its game slogan – “thinking with portals” – alludes to its core mechanic. The player is first given one gun, to beam portals into specific surfaces of a research lab. When they enter the blue portal, they will come out the orange portal, and vice-versa. When they are given a second gun to completely control the portals, that's when the puzzles get very difficult. It was completely original and took the world by storm with an additional layer of genuinely amazing storytelling and good graphics. Each puzzle you solved made you, the player, feel smart for solving it – and that's the mark of a very good videogame.

Mechanics as Metaphors



As both a game and a poem (as stated by its creator, Cecile Richard), Novena has a lot of artistic metaphor hidden within it, reinforced by its simple mechanic. All you have to do is return to the shore. Each time you go there is a slight change in weather, but nothing other than that. This could echo stanzas with small changes to the line while the general rhythm is treated the same. Within the narrative, going back to the shore is treated the same as your presence to a long-suffering friend. You don't have to do anything; just being there is enough.

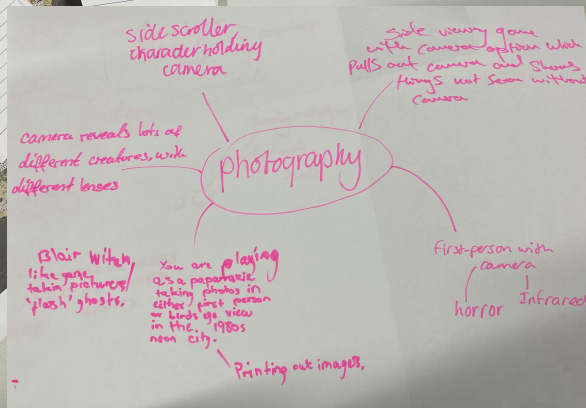
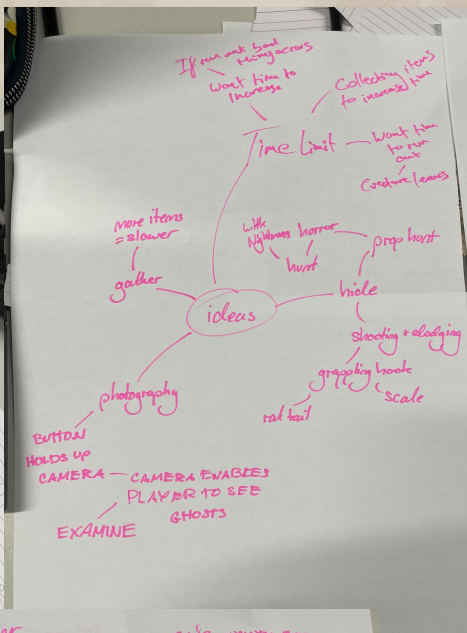
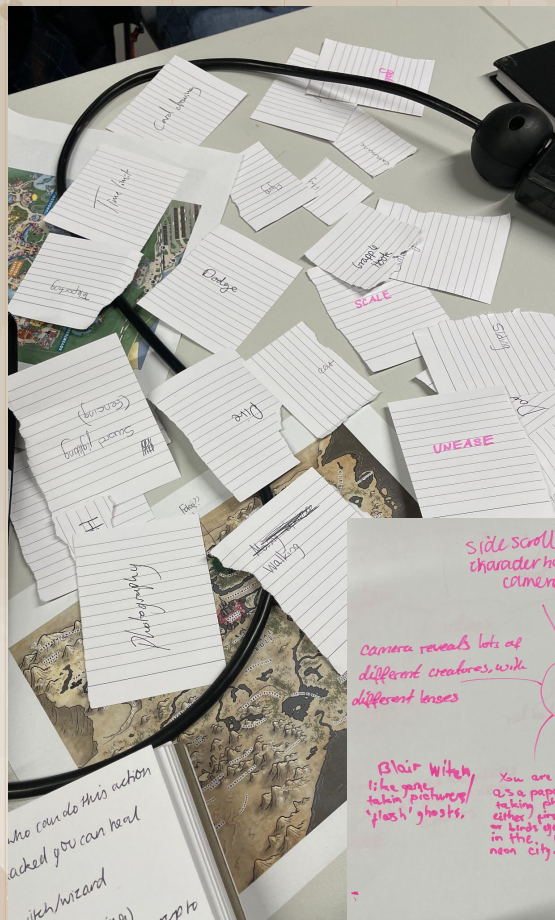


Layers of Fear is a sensational horror game that captured people with its realism and off-beat approach to tension. The metaphor here is in the game's mechanic of guiding you. Instead of unlocking areas of the house to explore as a more traditional horror would, the player character stumbles through rooms that are always changing, flipping the layout of the enormous mansion constantly, serving to confuse and frighten. The metaphor here is that your player character is suffering through a changing state of events. You were an acclaimed artist, with a beautiful, loving wife who had a talent for musical instruments, and an adorable baby daughter. Then, as if in an instant, your gallery burnt down; your wife, caught in the fire, was so badly damaged by it she can no longer play her instruments, and nor is she beautiful; and without your wife to take care of the child it falls to you to take care of a kid who cries constantly and needs looking after. It also reflects your deteriorating mental state. Your drunkenness and psychosis are reflected with rooms changing in an instant, hallucinated rats, and hallways stretching to infinity and back.



Depression Quest is a deceptively simple choice-based Twine game, created by Zoe Quinn, Patrick Lindsey, and Isaac Schankler. As a metaphor for depression, and how it sucks out all the energy and goodness from you, it presents you with a scenario about your life with a couple of options – only, the best option (usually the one that means you have to open up fully to people, or that take up the most energy) is always struck out. You have to scramble to pick what you think the second-best decision is, only some drain you more than others. You find the more you isolate, the more decisions get struck off, until only the worst, most self-destructive habits remain. However, not all is necessarily lost. Choose better options and better decisions become available. There is always a way to go back and better yourself. The ending you get reflects how well you were able to cope with your depression – however notably, the disability always robs you of that first, best option.

Concept Ideation



My group used a mindmap to come up with an array of ideas for mechanics we could use, combining mechanics, emotions, and genres to conceptualise potential new games with an invented “card game” done in class. From this we narrowed it down to three main ideas we liked: hide, gather, and photograph. The former two because they seemed simple, and slightly more unconventional than something like “jump” or “run” – and the last because it was completely unconventional, totally unique.

We came down to a vote, and when the vote ended in a tie, a coin flip, which decided photography. However, after we struggled to think of something simple and subversive for photography, quickly changed it over to “hide”. We were able to conceptualise and agree on some ideas at this stage. The next slide is a Miro board we used for a virtual mindmap, that we were able to update independently over the week to settle on an idea by next meeting.

Untitled form

Questions

Responses

5

Settings

5 responses

Accepting responses

Summary

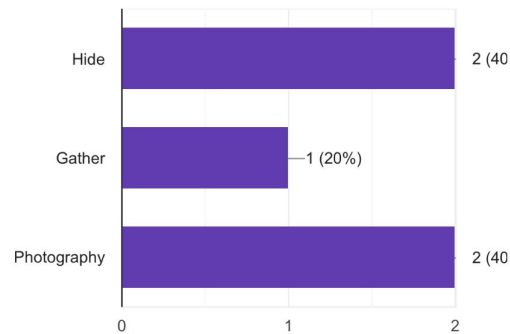
Question

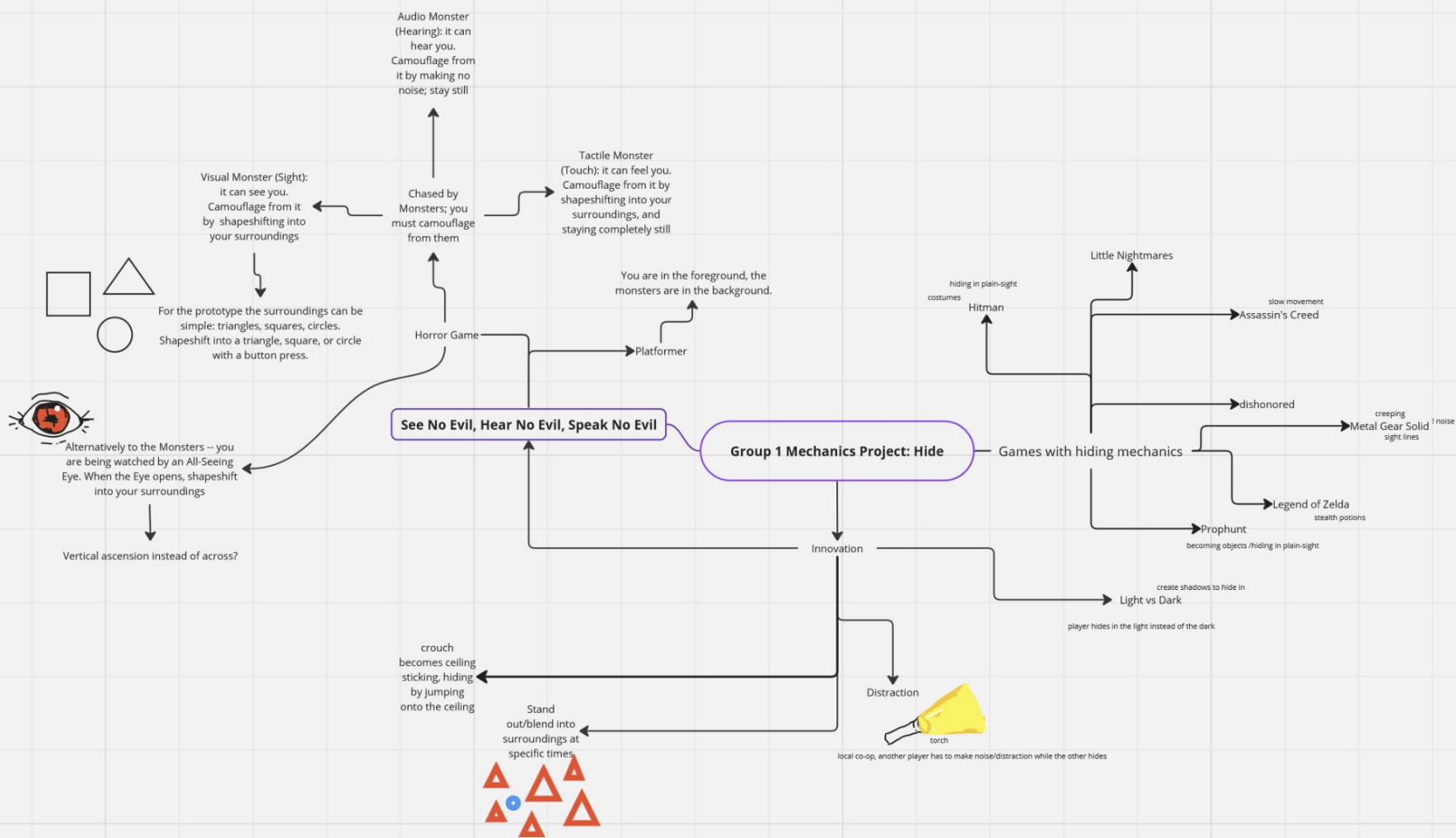
Individual

Mechanic

Copy

5 responses





The “Hide” Mechanic

There are many different games that have utilised hiding and sneakery, so in order to know how to subvert it, I looked to other games to examine what they did. Usually hiding is part of the core-meta gameplay, a secondary mechanic, especially with wide-open adventure games. For example in Skyrim (2011), the main controls revolve around fighting and exploration. While stealth is a very popular aspect of the controls, it is not the main control. However, the games below are either heavily characterised by, or the full intention is, sneaking around.



Little Nightmares is a 2017 horror game where you play as a tiny girl in a yellow raincoat, and your whole objective is getting to your destination, and staying out of sight while doing so. As a tiny person in the midst of enormous, horrifying people, you have to find creative ways of keeping yourself hidden in the shadows – in cabinets, under tables, in vents. While there is no specific mechanic to hide, it all revolves around staying in shadowy places out of eyeline of the antagonists, and if you are spotted, it's game over. This is the most traditional way of the hide mechanic.



Assassins Creed utilises a progressive alert system, in which a metre fills up on your screen, and if it gets high enough, combat begins. What fills the metre are things that will make you more conspicuous. Loud noises, dead bodies, and staying in the light will fill this metre. Additionally, if a guard sees you, you only have roughly one second to hide yourself before combat begins. It's a very elaborate, unique system that allows the player to know exactly how much trouble they're in, instead of a one-hit kill if you're seen. You can even use it to your advantage, purposely using the metre to attract guards you know you're stronger than.



Metal Gear Solid (1998) was one of the first games to use the hide mechanic in an interesting way – with tactical espionage. Before this game, that aspect hadn't really been explored. With new mechanics like “creeping” and “sightlines”, its popularity exploded, and has been added to most subsequent games that include a sneak option.

Types of Hiding

However, there are different many types of hiding other than just staying out of sight, and as we began to brainstorm on Miro, I realised that most I hadn't ever seen utilised outside of a one-off gimmick or a minigame – never as a core mechanic.

HIDING IN PLAIN SIGHT:

The term “hiding in plain sight” refers to a way of hiding by not really hiding at all; by blending in over crouching conspicuously behind structures. The best example I can give is from the game Inside (2016) where you play as a little boy, being mysteriously hunted by seemingly the entire world, which has fallen into some strange dystopia. During a couple parts of the game, you're required to stand in a long line of zombie people being monitored by a robot, and act just like them so as not to be spotted. There are multiple exercises of jumping in time with the others; spinning in a circle; and even walking had to be strictly timed and regulated during this portion. You're “hiding” by blending in with the others. We came up with an idea about being a shape, able to blend in with the same shapes.

CAMOUFLAGE:

This idea was mine, and mostly inspired by a Minecraft minigame. There are the “hunters” vs the “hunted”, where the hunters are gifted with swords and armour, and the hunted are gifted with camouflage. During the game you can disguise yourself as any block, and affix yourself somewhere inconspicuous: a haybale in a pile of haybales, a furnace next to a crafting table. It requires having to know the map (or hitting random blocks until you catch someone), and was tremendously tense when someone a hunter was investigating around your area – but tremendously fun, too. So what if the player character could camouflage too?

LIGHT AND DARK:

Most videogames require you hide somewhere shadowy. When you want to hide in real life, you'd usually choose somewhere obfuscated by darkness. However, what if you were being searched by something nocturnal? Then you'd be required to keep to the light. Additionally, it would make the dark moments more tense if the player character did not possess nightvision, and only had a hazy idea of when they were being seen. It could even have a “Grandma's Footsteps” aspect with the light turning off and on.

DISTRACTION:

Another way of hiding is to use a distraction, which is a concept we briefly toyed with. You'd have to complete a puzzle within a time limit to set off a noise or a false prey, which would in turn distract a monster. Failure to set it up properly would result in the distraction coming too late, or the noise going off too early, or even not at all.

CEILING CONCEALING:

Most videogames have the player hide by crouching on the floor or hiding under objects. There are less videogames about using the ceiling to your advantage. After all, not a lot of people look up when they enter a room.



The New Concept

In the end, a lot of the ideas we had about subversions in got combined into one. This would come back to bite us, but for now we really liked the concept.

The Mechanics

The mechanic is in shapeshifting – our camouflage element. You play as a star, and different button inputs can turn you into either a circle, square, or triangle. Inside the map are colonies of circles, squares, and triangles all fighting for dominance in a rock-papers-scissors style battle (triangle beats circle, circle beats square, square beats triangle), where a square touching a triangle turns the triangle into a square. You must navigate through without being caught by the enemy, or letting anyone know you are, in fact, a star (since there is a time limit for the length you can shift as any one thing). Complete the game and you make it to the end, in a room full of star shapes, and you won't have to shapeshift again.

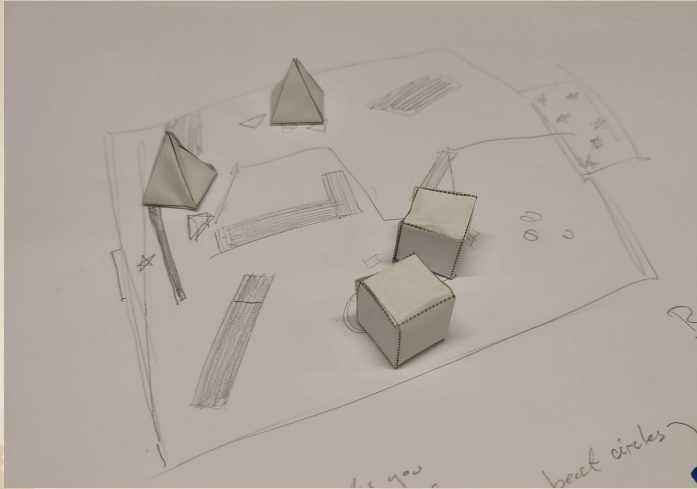
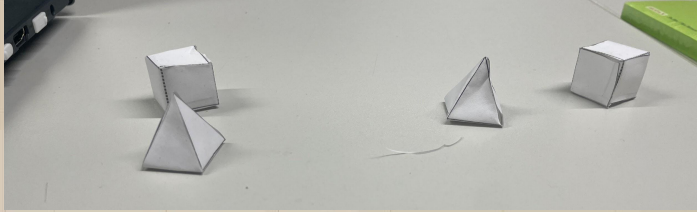
The Metaphor

The metaphor is about adapting to conform in society. Most people alter their behaviour based around the different people they meet, and sometimes people are hostile if you don't conduct yourself in the appropriate manner, so this is a message most people I think can understand. Additionally the star carries connotations of hope, specialness, and positivity, a much-needed mindset if you are constantly trying to change yourself: you're beautiful as you are.



Paper Prototype

The concept ended up being very messy to put on paper, and we realised it would be equally as messy to put into practice, so we began to steer away from the “fighting for territory” idea and more on the puzzle element of camouflaging. Our tutor, Emma Reay, additionally asked how we were going to indicate to the player a puzzle mechanic. From this our concept began to change. Again.*

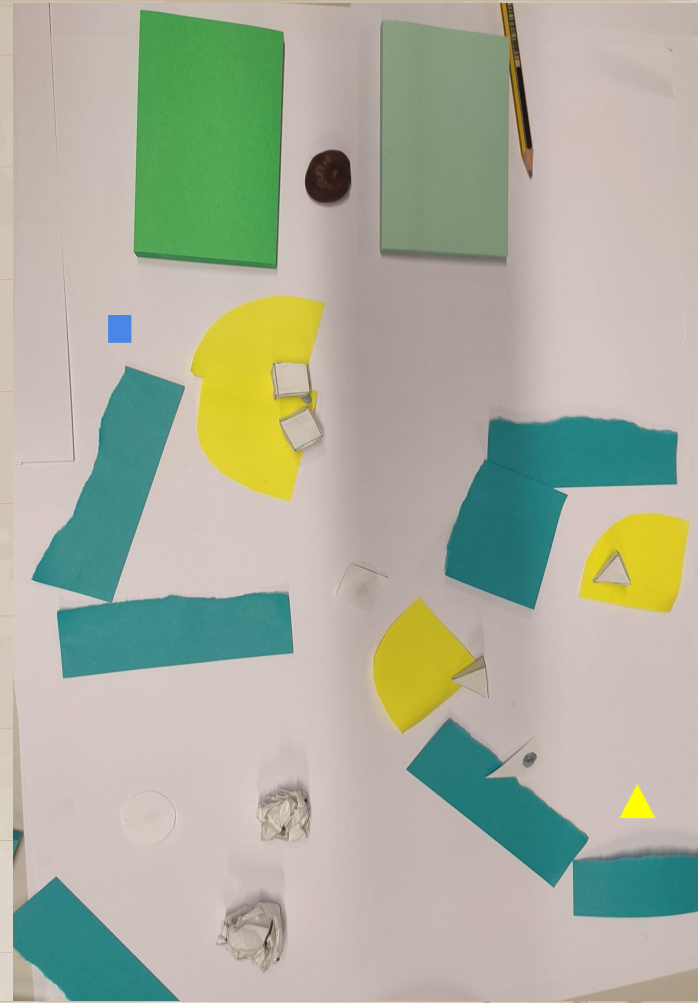


Starting from the top, you are first shielded by two walls on either side to stop the patrolling squares from immediately seeing you. Squares, we decided, would move in stiff, gridded lines. The yellow zones represent the radius in which the shapes can see you. From here you are protected, and it should be fairly intuitive to stay out of the light zones.

From there you see a smaller shape on the ground (the blue square). Pick it up, and you gain the ability to shapeshift into a square. This means you can safely get past them into the triangle zone.

Triangles move more randomly than squares, but you're still roughly able to predict what direction they will go in based on what angle they hit a wall. However since squares are predatory to triangles, they will ignore you, and you are not required to go into their zone – however the next zone, circles, ARE predatory to you. Circles move completely randomly and have 360° vision. You will have to retrieve a triangle from the triangle segment before you go forward, without revealing you're a star. It's a fairly simple first puzzle that should get the player to grips with all the mechanics necessary to proceed.

*Joe was absent on this day.



C#

During this phase of production, I also realised (after a week of relative inactivity) that we were all pretty passive people, and nobody was going to do much without being “assigned” a job, and we had nothing in terms of a prototype in a proper game engine, or roles. We hadn’t elected a leader; most jobs until now had been collaborative or by vote. Despite not being... overly confident, I took conditional initiative and gave people some jobs. I had Esi code a player-tracking movement, Joe (since he seemed the most experienced with code, and told us he had friends who knew C# well) to do some of the shape behaviours, Guy on level design (since he came up with the paper prototype level), and Olivia making the poster for the Arcade display (since she had been the one drawing nice diagrams and pictures for us in our notes/moodboards). Meanwhile I attempted a camera tracking movement and a player movement. I wasn’t confident, since I don’t comprehend maths and hadn’t ever coded before, but both of them ended up working. I was going to attempt the shapeshift mechanic too, when we hit complications.

Camera Tracking Code

```
C# PlayerMovement.cs X C# CameraMovement.cs X
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 No asset usages
6
7 public class CameraMovement : MonoBehaviour
8 {
9     public float FollowSpeed = 2f; // Unchanged
10    public float yOffset = 1f; // Unchanged
11
12    public Transform target; // Unchanged
13    // Start is called before the first frame update
14    // Event function
15    void Start()
16    {
17    }
18
19    // Update is called once per frame
20    // Event function
21    void Update()
22    {
23        Vector3 newPos = new Vector3(target.position.x, target.position.y + yOffset, z - 10);
24        transform.position = Vector3.Slerp(a: transform.position, b: newPos, t: FollowSpeed * Time.deltaTime);
25    }
26
27 }
```

Player Movement Code

```
C# PlayerMovement.cs X C# CameraMovement.cs X
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 No asset usages
6
7 public class PlayerMovement : MonoBehaviour
8 {
9     public float moveSpeed = 5f; // Unchanged
10    public Rigidbody2D rb; // Unchanged
11
12    private Vector2 movement;
13
14    // Update is called once per frame
15    // Event function
16    void Update()
17    {
18        movement.x = Input.GetAxisRaw("Horizontal");
19        movement.y = Input.GetAxisRaw("Vertical");
20    }
21
22    // Event function
23    private void FixedUpdate()
24    {
25        rb.MovePosition(rb.position + movement * moveSpeed * Time.fixedDeltaTime);
26    }
27
28 }
```

Complications

The game was too complicated. Much too complicated. While Joe was coding shape behaviours, he realised how complicated it was all going to be, and laid it all out for us. Having “sightlines” was going to be more difficult than we’d intended, and coding got very complicated when there were walls for shapes to bounce off and block views. Most of the way through the timeline, we’d have to start again. Luckily he already had an idea that was both simple, subversive, and, mostly importantly – he claimed it was codeable in a week.

The New Concept (...again.)

The new concept went back to one of the original three mechanics: gather. The player would spawn as a simple circle and collect as many X’s as possible within a time limit. Once your time limit was over, the next player would start. This could either be competition focused, to see who would collect the most X’s, or done as a collective. Joe, Esi, and I unanimously voted for the second option, since it engaged more people to work to one goal, which is a little rarer than being competition-focused. Guy and Olivia abstained from the vote.

Rules:

- when not transformed, everyone attacks you
- if they see you transform, you’ll be attacked
- circles beat squares, squares beat triangles, triangles beat circles



Roles:

- level designer
- movement coder
- enemy movement coder
- transform coder

Metaphor

- Conform or you’ll be targeted
- The world demands constant change

Players Goal

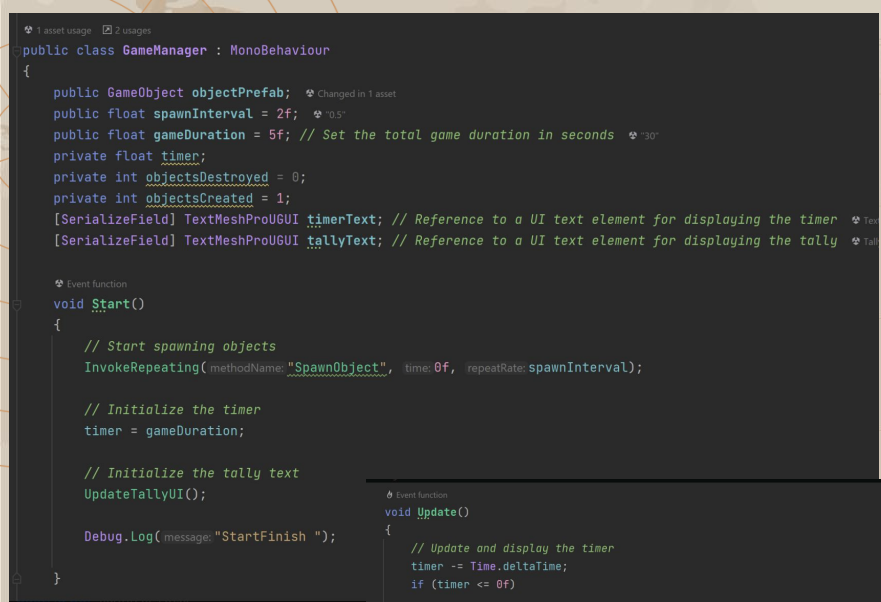
- Arrive at the “safe point”

Obstacles

- ~~at~~ limited no. of transforms
- Cooldown between transformations
- If seen, you have a limited window before being caught
 - ↳ can try to just run it without hiding

Fail State

- Game over



```

1 asset usage 2 usages
public class GameManager : MonoBehaviour
{
    public GameObject objectPrefab; // Changed in 1 asset
    public float spawnInterval = 2f; // 0.5
    public float gameDuration = 5f; // Set the total game duration in seconds // 30
    private float timer;
    private int objectsDestroyed = 0;
    private int objectsCreated = 1;
    [SerializeField] TextMeshProUGUI timerText; // Reference to a UI text element for displaying the timer
    [SerializeField] TextMeshProUGUI tallyText; // Reference to a UI text element for displaying the tally

    // Event Function
    void Start()
    {
        // Start spawning objects
        InvokeRepeating(nameof(SpawnObject), time: 0f, repeatRate: spawnInterval);

        // Initialize the timer
        timer = gameDuration;

        // Initialize the tally text
        UpdateTallyUI();

        Debug.Log(message: "StartFinish ");
    }
}

```

```

// Event Function
void Update()
{
    // Update and display the timer
    timer -= Time.deltaTime;
    if (timer <= 0f)
    {
        timer = 0f;
        // Handle game over or other logic when the timer reaches zero
        objectsDestroyed = 0;
        timer = gameDuration;
        // CancelInvoke("SpawnObject");
    }

    UpdateTimerUI();
}

```

```

// Frequently called 1 usage
void UpdateTimerUI()
{
    // Update the UI text element with the current timer value
    timerText.text = "Time: " + Mathf.Round(timer);
}

```

```

2 usages
void UpdateTallyUI()
{
    // Update the UI text element with the current tally
    tallyText.text = "Objects Destroyed: " + objectsDestroyed;
}

```

Joe's Code

I was a little disappointed that my own code was now void after working quite hard on it (it did not come naturally to me, hard as I tried), (also our research on hiding was now void which was a little irritating), I am very much grateful to him for taking on the job. After 2 weeks he handed over the prototype, and there was only one alteration where, instead of having a player character, it was now point and click.

We had a week left now, which wasn't a lot, and still had to design it. Currently the clickable object to gather was a pixel .jpeg of a pot. We held a meeting that Saturday to decide on design, however only Joe and Esi entered the call. I suggested a simple design of sheep in a field to complete the metaphor, as counting sheep is often what people do to go to sleep.

```

// Frequently called 1 usage
void SpawnObject()
{
    if (objectsCreated < 1000){
        Debug.Log(message: "SpawnObjectRan: ");

        Vector3 spawnPosition = new Vector3(x: Random.Range(-150f, 150f), y: Random.Range(-90f, 30f));
        GameObject newObject = Instantiate(objectPrefab, spawnPosition, Quaternion.identity);
        float randomScale = Random.Range(10f, 50f);
        newObject.transform.localScale = new Vector3(x: randomScale, y: randomScale);
        objectsCreated++;

        CollectibleObject collectibleObject = newObject.AddComponent<CollectibleObject>();
        collectibleObject.SetGameManager(this); // Pass a reference to the GameManager to the CollectibleObject
    }
}

// 1 usage
public void ObjectDestroyed()
{
    // Called when an object is destroyed
    objectsDestroyed++;
    UpdateTallyUI();
}

```

Sheep Complications



We decided that Esi would design the field background, and I would make the sheep. I wanted this task to see if I could code in any variation to the sheep. Initially my idea was just to make an identical one in red and blue, and get them to spawn in at random for some variety. This task did not end up being something I could integrate, although I did very much try.

I had an uncomprehending look at the code and made a space underneath `objectsCreated++`; since that was in the functions that dictated the object behaviour. The best way to go about it was to make a folder, and have the code randomise what asset was selected from that folder to spawn. That way I could make more unique sheep sprites, maybe even a whole farm. However, I couldn't make it work. I tried over the next 2 days and failed. My wording was never precise enough on Youtube tutorials to convey what I meant, so I tried making some of it up. Unsurprisingly this didn't work.

My (Attempted) Code

```
string[] assetPaths = Directory.GetFiles(folderPath, searchPattern: ".png");
foreach (string assetPath in assetPaths)
{
    Sprite assetSprite = LoadAssetSprite(assetPath);

    GameObject newAsset = Instantiate(assetPrefab, GetRandomPosition(), Quaternion.identity);
    newAsset.GetComponent<SpriteRenderer>().sprite = assetSprite;
}
```



Design Wrap-Up

To finalise the game and tie it all together, Esi designed the background, chose the font and tweaked the words around for a proper objective, and even added in her own background music she'd created!



Evaluation

After the first module, I went into the second with cautious confidence. I have worked in teams before to make games on just a week's schedule. Plus it would provide an opportunity to get to know the class better, as people tend to form and stay in their own cliques. And while I really enjoyed getting to know and working with my teammates, in the end what hindered the project the most was exactly what I thought it was going to be: over-ambition and lack of communication.

We spent far too much time coming up with an idea we could use, and even longer chasing an unattainable goal, not self-assigning roles (or assigning others roles) – and that all dragged the project down a lot. I consider myself a passive person. I like having control over my own tasks, but generally dislike to dictate others because I don't want to be viewed as pushy, or rude. Equally I think my other teammates felt the same. It was difficult getting us all on the same page, as inevitably one of us wouldn't be in class that day, or wouldn't messages on the discord server, or turn up for an online call, or didn't use Miro or Jira. Initially I thought having 5 to a group instead of 4 was better, since we had more people to do the tasks. In the end it kind of ended up being a curse.

However, as all these failures bit us from behind, it was actually the most cohesive our little group got; the last few weeks were by far the best in terms of both productivity and communication. We had a sense of camaraderie about our shared mistakes and the pressure caused us to finally stop worrying about each other and divy up the roles to put us at our strengths. We had a sense of relief at having a prototype to turn in, and additionally something that was designed, with music. It was not the best-case scenario. It was definitely a bit rushed. However I intend to learn from these mistakes going forwards. I really, really wouldn't like a repeat of this.





Thank you! :)

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**