C H A R M E D

Contributors:
Rei Ito - s3607050
Morgan Thomas - s3605119
Jamie Lin - s3604742

>Instructions:
The game is located in the 'Build' folder, just click on the executable to play.

>Narrative:
While paddling in the rock pools, a child dropped their precious bracelet, the charms scattering everywhere. Collect the pieces and return them to the child, safe and sound.

>Controls:
- *Use the A and D keys (or left and right arrow keys) to swim forward, alternating between them to build momentum.*
- *W and S to rotate up and down then continue to press forward.*
- *Move mouse to control camera.*

What Changed?

Inspired by 'ABZÛ' by Giant Squid Studios, we had Initially planned to create a game set in an aquatic world where the player is a small fish. One of the main goals with this project was to make the fish interesting and fun to control, creating fluid swimming dynamics and motions. The other main goal of the project was to replicate the game's serene atmosphere but add a chilling twist when the player's fish friends begin being caught by a fisherman. The goal would have then been to interfere with the catching to save the fish's friends.

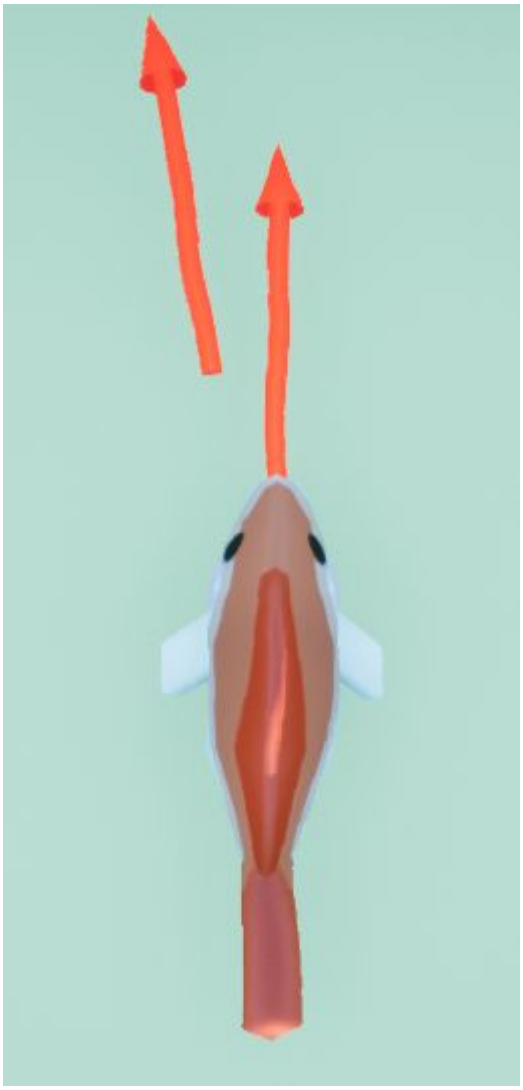
However, we decided to go down a different path of narrative after looking at Studio Ghibli's 'Ponyo', an animated film about a little boy and a fish girl. We wanted to replicate this relationship between the fish and a young child and make the player's goal to help the child look for something they'd lost in the water. We chose this direction instead because it's endearing and complementary to the atmosphere we've created, instead of flipping the calmness of the scene on its head by putting the fish in a life or death situation.

This new concept has been scaled down immensely due to several reasons. We realised the scale of the first concept was too ambitious given our experience in modelling, rigging and gameplay programming because originally, we planned to have too many characters/NPC fish. The artist had to build their 3D modelling skills as the project continued, as they were new to the act. The programmer had to spend most of the time creating a fluid feeling character movement with no point of reference of already established movement systems i.e. fps, platformers, etc. No other NPCs were added since even a passable fish AI required to make fish look OK was still too time consuming. We did way that in the document that we would introduce more high stakes conflict but with our new narrative it made no sense to continue with it.

The Good, Bad and the Ugly

Engine Work

We decided to go with Unreal Engine 4 for this project since Rei was well versed with the engine from his work from his capstone. It also contained elements that were vital to the look and creation for the movement system like: easy to use post-processing and shaders, physics assets and physics-based animations, and fast prototyping with blueprints. It also doesn't hurt to use the same engine that the developers of ABZU used. Rei did most of the movement so here's how it worked and what went wrong.



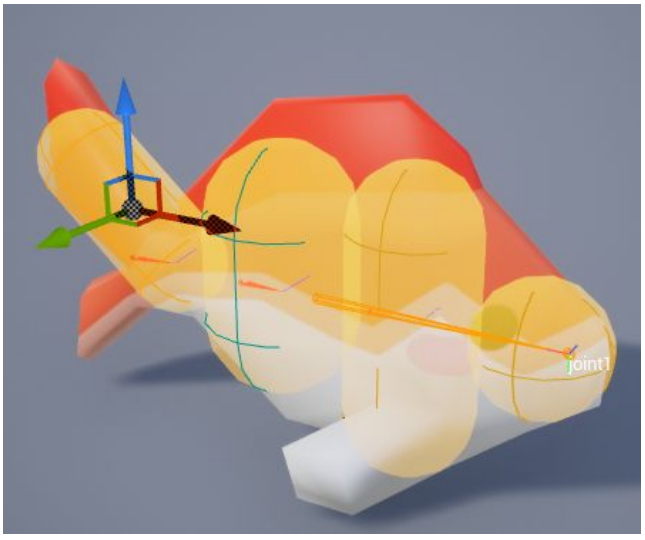
Movement

The movement of the character has been the main focus, and also being the hardest to get right.
The intention was to gamify how a fish moves using physical animation to simulate the movement while the controls had to be something other than 'press a to move forward. So, the movement works with a whole lot of rotational math (yay!). The object has an arrow showing its facing direction, and the mesh has its own for when the player turns the fish, the difference is then calculated to then enter conditionals to decide what to do next. The first one being if the difference is lower than a certain amount either left or right, the object stays facing forward and the check if the arrows have intersected, showing the player is trying to move forward and adds forward force to create movement. The other condition has if values exceeded that max turn area, turn the whole model so that it faces another direction; adding small forward force to simulate the turning motion because this scene is using no gravity so there is no turning momentum. Next was to put in the up and down movement. The same method could have been used but when implemented, the checks were not going through and I'm nowhere near an expert at rotational calculations, so I just made the pitch direction follow the fish's mesh direction. This turned out great in the end since that style of movement would be both dolphin and shark style movement, so the feedback wouldn't feel as good. I did try to have a static camera that followed this movement but no matter what I tried it would always get stuck, destroying the gameplay experience; so, I opted for mouse camera controls. Luckily this also worked out since we changed to a searching game and the player-controlled view makes it less infuriating to look for the charms. It was extremely challenging to get the logic to be correct in both calculations and in game feel, but it fascinating to experience and attempt how these elements intersect to create new, revolutionary movement systems (ala ABZÛ).

Sadly, this isn’t perfect, as there were instances where combining yaw and pitch turning would break the movement system, rendering the player useless. I couldn't figure out what was happening, so I just caught when it happened and reset the objects rotation. Luckily the player can’t see this but that’s both a good and a bad thing; seeing as if there’s a case that I haven’t caught and breaks it, there’s no recovering, breaking immersion and inciting frustration. Would’ve liked to iron this out but trouble with getting calculations right and the need to implement the actual game got in the way.

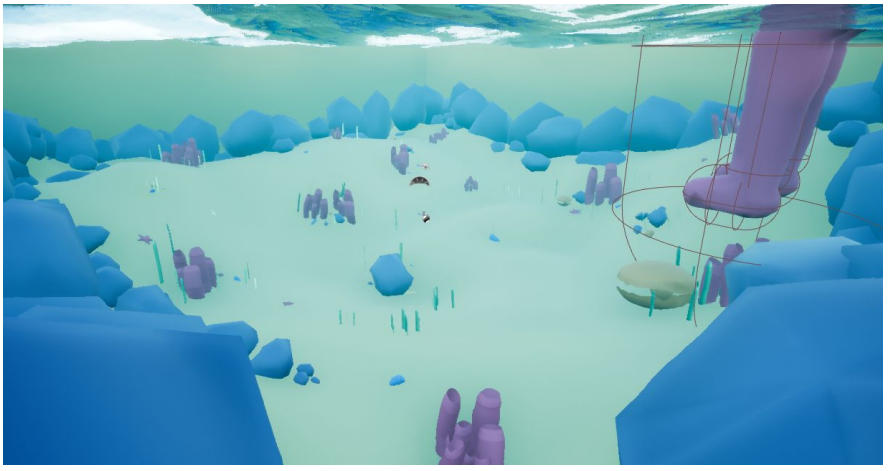
Physics!

Rigging and how it came into the engine was another big part of this assignment since the artists were new to everything 3D, we had to compromise to save on time and experience. We decided to just understand modelling and rigging and use Unreal Engines physics-based assets to simulate physics-based movement. This was another complex yet interesting aspect of the game since this was a mix between physics and visual feedback. Since I needed to get the skeletal mesh to move according to the player’s movement, socket constraints were used between bones to make sure the model didn’t collapse into itself. Values like weight, linear damping and rotational damping were used to simulate an actual fish but fine-tuned to so that the visual feedback matched up with the movement. The same method was used to the bracelet to follow the players tail, using the bone locations to attach picked up charms. It’s not directly obvious but the seaweed is also rigged so they react to the player when touched; not enough time to get them to move as if they were in water and some of the smaller strands glitch out like crazy.



The collisions had to be done manually since the player is set up with one object that’s collide-able (base) and another for the fish direction that has no collision. This was done by checking what the fish collided with, get the collision normal for the direction vector and add force depending on the player’s hitting velocity. In the end this worked out quite well, the only problem being that the colliders on the meshes aren't custom, so the player is shoved back when they really shouldn’t. This could be fixed by setting up collision geometry in Maya but we all had enough to do in the meantime and the colliders aren't game breaking.

Gameplay



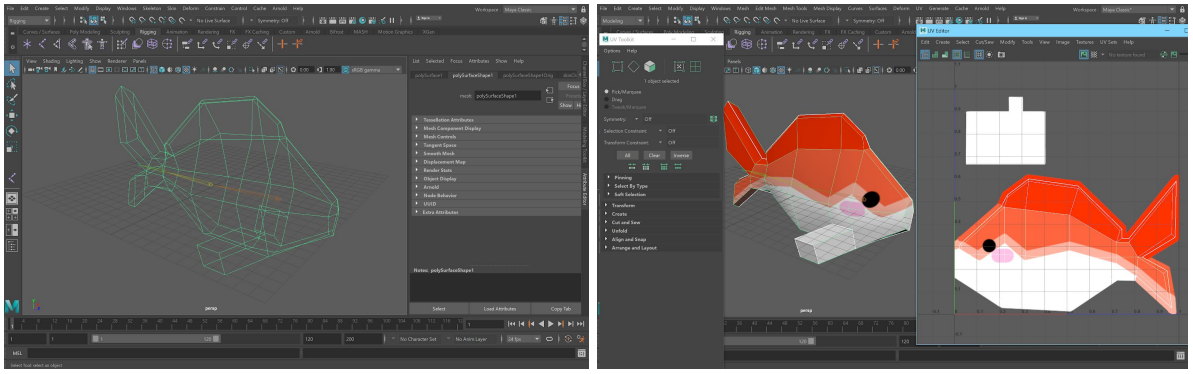
Since my head so far into rotation calculations and physics behaviour, by the time it came to switch mindsets and implement a quest for the player to conduct, fatigue set in and a dodgy, hard to edit/debug/understand state system was added to handle the different states the player could be in and what event should trigger when certain collision events take place. So, the idea was: when the player approaches the legs, dialogue starts between the mother and her child, explaining that the charms have been scattered across the map; when the player hits the legs, the child flees since a fish just touched their legs; and finally, when all charms are collected and returned, end dialogue plays out and the player is free to swim around (FOREVER IN AN EMPTY SCAPE). Putting these states together was a nightmare, all my own volition and minute details that, when added, broke the system. Like as a detail, if the fish touches the legs while the child is talking, it interrupts the dialogue, forcing the player to re-approach the legs after they come down. As stated, this was an unintuitive mess (probably from my lack of experience); and adding anything more to this would be a huge trouble and all its cases haven't been tested so it's prone to breakage.
In the end, the gameplay is extremely shallow (pun reluctantly intended). It's literally fetch quest under a cute-sty vies, which isn't enough to hold a player for a prolonged period of time. Making the player area larger, including puzzles to gain access to the charms and more varied environments were all goals but we couldn't to them in time since we needed to polish as we went to be ready for submission.

Effects

The level is entrenched with an underwater postprocessor that gives the look of being underwater. Three shaders were made but two were implemented: first was the water material, simulating flowing water using sine calculations mixed with 'panners' to achieve the look. The next was underwater distortion (applied to the post-processor), using similar techniques to the water but transparent. The final and unused one was water caustics, that used the above techniques but with an image and projected down as a light. This was omitted since we had to have the child's upper leg invisible which required a solid cube for the water, which then meant no light could penetrate it, rendering it useless.

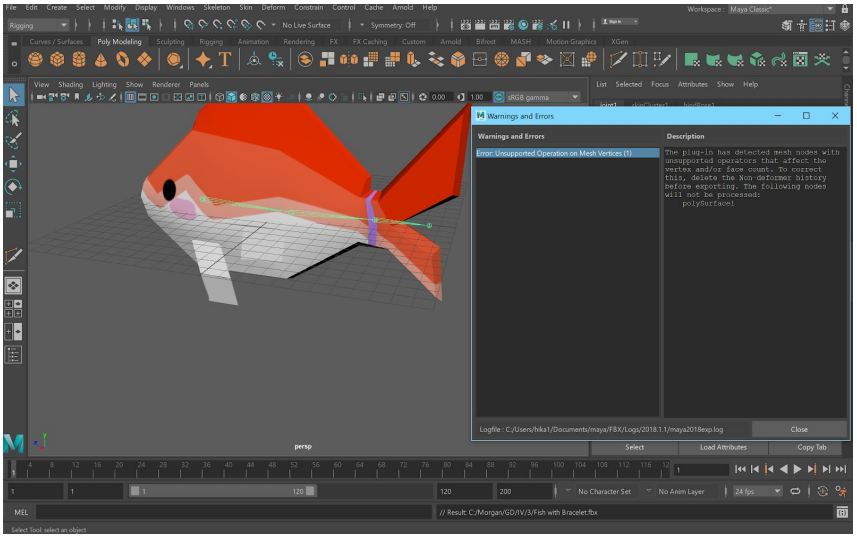
Modelling, Texturing, Rigging

The visual style of the game is quite reminiscent of ABZÛ, with soft yet bright colours and low-poly models. It seems effective for a relaxing environment, and compliments the simple gameplay well.



We learned how to properly UV map models in Maya after some trial and error, but it was probably the simplest of the new things to learn. The toughest part was figuring out how to get this model into Unreal with all its components, which we stumbled over a few times. Models need to be exported as an FBX file, but sometimes don't transfer with their UV or skeletal mesh attached. We would either must manually input these aspects, or re-export the file, which caused us to lose a lot of productivity time.

A persistent problem we faced was an error that appeared often when exporting FBX files. After a lot of manual debugging and trying to deduce what the issue was, we still don't really know even at this point. However, we compromised and adjusted the models to still do what we wanted them to.



Overall, a lot of our time spent on this project was attempting and failing, but at least we learned from it and will be more prepared in the future. For Jamie and Morgan, it was a great opportunity to get out of our comfort zones and experiment with new programs and engines.

Conclusion

In the end we are happy with the work, knowledge and experience gained from this project, allowing us to use these skills in future 3D, physics-based endeavours. We semi achieved the calming look and feel of the game but with our own style, along with implementing our own unique movement system. It would have been nice to implement more robust systems, polished models and animations and more intuitive gameplay but restricted time from overlapping projects, inability to debug and solve problems due to lack of experience have restricted us in what we could achieve by the due date. But the smaller scope allowed use to apply a variety of elements that we are proud of.