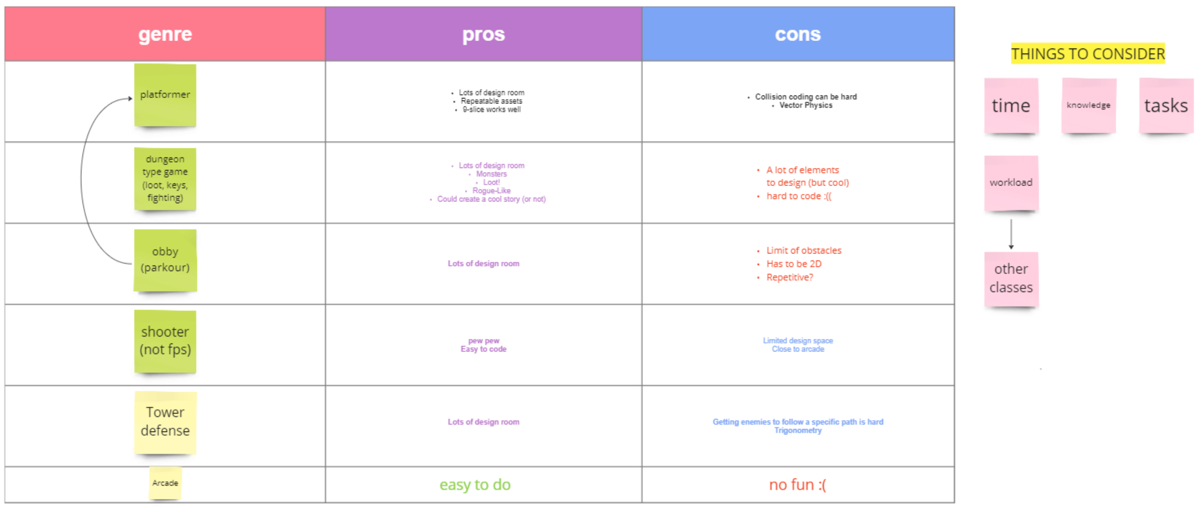
<u>**Assignment 4 – Process Work**</u>

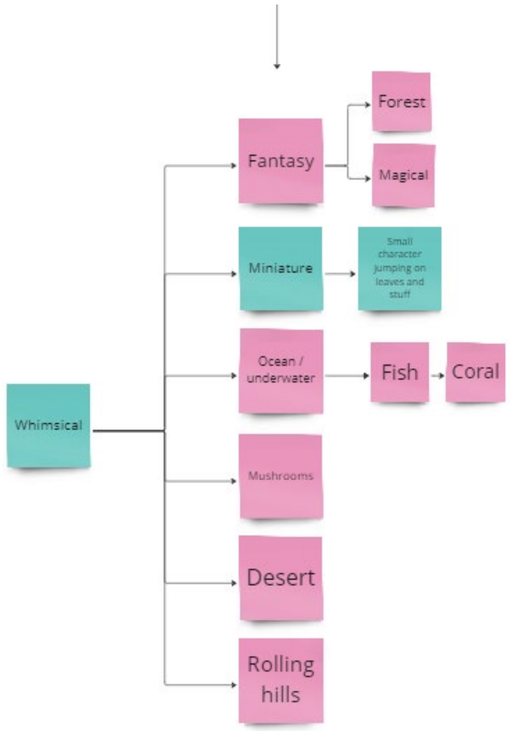
Initial planning:

We started planning our game by using Miro to brainstorm ideas and determine the best course of action for implementing our game. We decided to go with a platformer after weighing the pros and cons of each genre we came up with.



genre	pros	cons
platformer	• Lots of design room • Repeatable assets • 9-slice works well	• Collision coding can be hard • Vector Physics
dungeon type game (loot, keys, fighting)	• Lots of design room • Monsters • Loot! • Rogue-Like • Could create a cool story (or not)	• A lot of elements to design (but cool) • hard to code :((
obby (parkour)	Lots of design room	• Limit of obstacles • Has to be 2D • Repetitive?
shooter (not fps)	pew pew Easy to code	Limited design space Close to arcade
Tower defense	Lots of design room	Getting enemies to follow a specific path is hard Trigonometry
Arcade	easy to do	no fun :(

We then brainstormed ideas for the setting and theme of the game. We knew that we wanted it to have a whimsical theme, and we ultimately decided on a forest setting with miniature creatures and characters.



After assigning roles, we broke off to code our own parts and bring them together. I took on the role of making the game engine and implementing different parts. I decided to have an object class that every child class will inherit from, and a component class that different components will inherit from. The components can be attached to objects as a dictionary, and only one of a specific type can be attached at a time. I then created the player class to test the object class functionality and it worked!

Next, I added input to move the player, gravity, and the ground object. The player falls through the ground, so I made a collision component and attached it to both the player and the ground object. I made an on hit method for all objects and override it in each child class that uses it. The on hit method takes the object that was collided with as a parameter and does different things depending on the type of object.

My team finished creating the character sprite for the game and the animations, so I created a sprite component class to easily add the image to any object. I created a frame structure to get the individual frames from the sprite sheet and tested it out. After adjusting the scale and managing the source and target rectangles, the sprite was drawn properly. We then encountered a problem where the size of the monitor was different for each computer, where one person had a 1600 x 900 pixel screen and another had 4k. To solve this, we had to change the window from scaling with the window size, to making the window 1600 x 900 pixels in size.

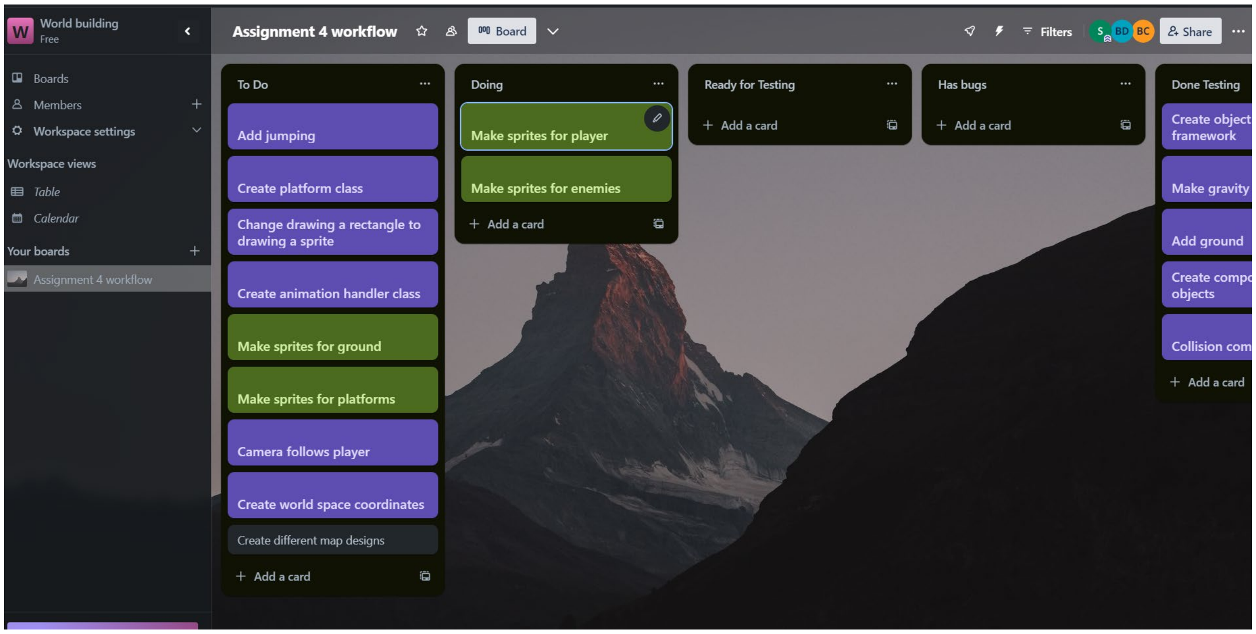
After we solved this issue, I created an animation component and attached it to the player. I also added a subroutine for the player to control which animation to display and when. We then had our first merge error, which we ended up resolving by keeping the original branch, and updating the merged branch from main, then copying the new code into that updated branch to have it work properly. This error was caused by not updating from main before writing new code, and we tried to be more careful about this in the future.

I then added a camera component class and attached it to the player so that the main window would follow the player. I then translated the location of all the objects based on the camera location, except for the player which stays in the middle of the screen. I also added the enemy class and the enemy fly child class to test the camera and game engine is working properly. After fixing some scaling issues it worked!

We added background images, and I added the ability to jump. There were a few issues with gravity and jumping, but I solved it without too many issues. We added the platform class and the ability to create any number of platforms at any size.

Although the game is unfinished, the basic elements of the game are all there, and all that is left is designing and developing content for the game. We will likely continue to update the game before the submission deadline. We plan to implement the other enemies, collision for platforms, and designing the actual level.

Throughout the project we used Trello to manage who did what and to signal to the team when things were done.



To Do
- Add jumping
- Create platform class
- Change drawing a rectangle to drawing a sprite
- Create animation handler class
- Make sprites for ground
- Make sprites for platforms
- Camera follows player
- Create world space coordinates
- Create different map designs
- Add a card

Doing
- Make sprites for player
- Make sprites for enemies
- Add a card

Ready for Testing
- Add a card

Has bugs
- Add a card

Done Testing
- Create object framework
- Make gravity
- Add ground
- Create compo objects
- Collision com
- Add a card