My incremental game is about mining for fossils and using them to revive dinosaurs to fight other dinosaurs in an arena. The game has two major loops: a loop to gather resources and a loop to improve dinosaurs. I chose a panel approach for the game, since resources in the early parts of the game are still being used later.

The game opens on the first panel, titled Excavation, with a single resource: ore. Ore is manually collected by the player until they get 10, at which point they unlock mining rigs, which each cost 10 ore to build. Initially I had each rig collect at a flat rate of 1 ore per tick, but I felt that the linear increase was not very interesting to watch, so I instead implemented a RNG system where the ore collected by the rigs is a random number between one and the total number of rigs, so that there is some more variance. It also introduces a little bit of risk-reward when fossils are introduced (see below).

Once the player collects 100 ore (which would ideally be easier once mining rigs are in play), they discover their first fossil. From this point, the player can clean ore to uncover fossils at a rate of 1 to 1. Once the player collects 20 fossils, they unlock the collector for them, which are called fossil cleaners, and each cost 20 ore. Fossil cleaners function similarly to mining rigs, where they collect a random number of fossils between 1 and the number of cleaners, however they also subtract that same random number from the ore. Initially I had it so that the fossils gained is random, but the ore subtracted is always the number of cleaners. This quickly proved to be way too punishing, and I found myself softlocked when there were too many cleaners (or at least more cleaners than mining rigs) where the rate of ore consumption was much, much faster than the rate it was gained at. Balancing that made it so that there was still a risk of having cleaners outnumber rigs, but it was actually possible to recover from.

Once the player collects 200 fossils, they unlock the second panel: Paleontology. Here, fossils are consumed to identify which of 10 possible species of dinosaur they could belong to. There is a 50% success rate for this process, and it costs 100 fossils to do. Initially, this is a steep number to slowly introduce the dinosaurs, but later in the game it becomes very trivial as fossils are automatically collected. Because the number of identified fossils per species becomes very important in the final phase of the game, I wanted the player to pay attention to it and deliberately increase it, so it does not get automated. This is also where I implemented my procedural commentary: when the player successfully identifies a fossil, it randomly picks one of the 10 possible species, and then one of 9 possible body parts that the fossil could be.

When the player finds 5 out of the 10 different dinosaur species, the third and final panel unlocks: Combat. Here the number of identified fossils per species gets recontextualized as experience levels. On this panel, the player's collected dinosaurs and their levels are displayed, and below it is the main focus of the panel: the arena. This is the visual minigame. To play it, there is a dropdown menu where the player can select which dinosaur is in play, and there is a corresponding sprite in the canvas. Clicking anywhere in the canvas moves the player's dinosaur to that position. Enemy dinosaurs will spawn in from the right side and move towards the left. The highest possible level that an enemy can spawn at is 1.5 times the player's highest level dinosaur. Enemies that are of equal or lower level to the player's dinosaur are labeled with green, while more powerful enemies are labeled with red. Colliding with a red enemy will cause the current dinosaur to lose 1 level. Colliding with a green enemy will earn money (labeled winnings, which is the final resource of the game). The money earned is determined by the following formula: a random number between 1 and 1000, multiplied by the current dinosaur's level, divided by the difference between the player and defeated enemy dinosaur. I had to do a lot of experimentation with this formula, but I ultimately decided on this one because each of the three parts serves a different purpose. The random number works similarly to the way mining rigs work, where there is some variance in the increasing number. Multiplying by the dinosaur's level encourages the player to continue increasing their dinosaurs' levels (and since this increase happens at random, it encourages switching between them to whichever is highest at the time). Dividing by the level difference rewards the player for going after similarly-leveled enemies without explicitly punishing them for defeating weaker enemies.