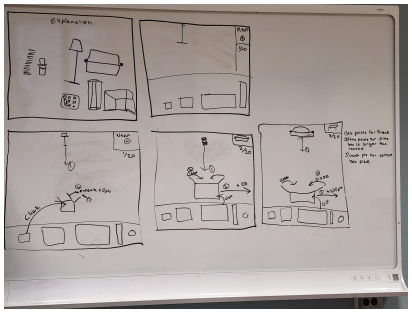
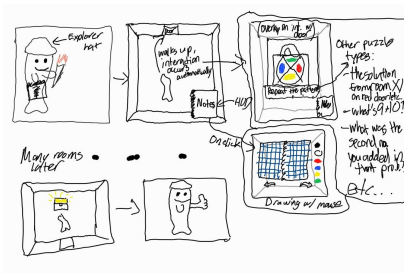
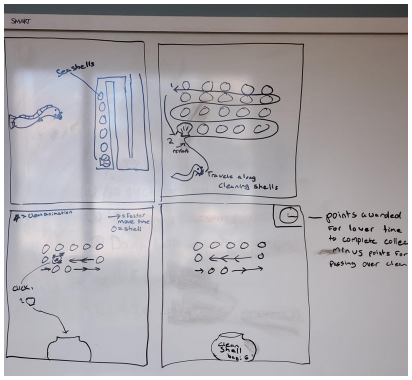
Each Game Last About 5-10 minutes

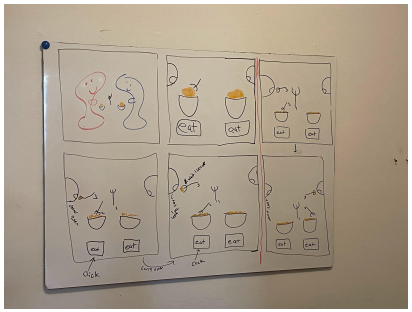
Inefficiency	Storyboard	Game Flow	Sprites needed
Inefficient Data Structure (IDS) - Refers to a suboptimal choice of data structure used in a project that causes extra processing that would take longer for loading or adding bloat to the functions in the project.		Basic Flow: - There are several (20) pieces of furniture that the user is presented with, and is told that they must package the furniture in the appropriate (5) boxes. - Each piece of furniture, one at a time, falls from the top of the screen, and the user must drag one of the boxes from the bottom of the screen under the furniture to package it. - If the box cannot hold the furniture, then the furniture breaks and the user is awarded 0 points. - If the box holds the furniture, but is not the optimal box, then the box closes, and the user is awarded half (5) points. - If the box holds the furniture and is the optimal box, then the box closes, and the user is awarded full (10) points. - After the user has placed all of the furniture into boxes, the game displays an ending animation where the moving truck contains all of the items which were placed into the optimal boxes in the truck, any broken items in a dumpster (or on the curb) next to the truck, and any items in suboptimal boxes on the curb next to the truck (because they do not fit in the truck). - Then, the points are tallied and the user is awarded a corresponding star rating (3 stars for a perfect score, 2 stars for a score	● “Moving fishes” holding/moving the boxes you click on ● Furniture (x20; x10 w/ one recolor each if 20 is too much) ○ Broken animation for each ● Boxes (x5); open, closing, closed ● Moving truck: see-through, open; see-through, closing; see-through, closed; normal, closed ○ (back-view) ● Generic fish driving truck

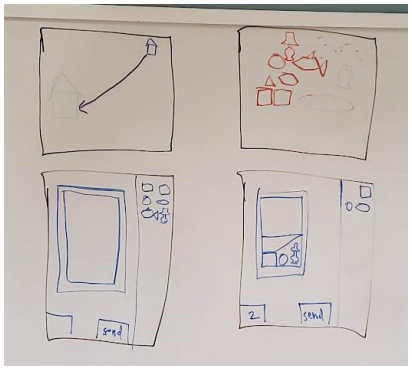
		between perfect and half-perfect, 1 star for a score between half-perfect and 0, and 0 stars for a score of 0)	

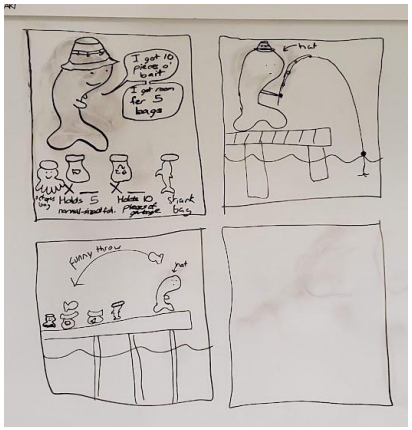
Repeated Computation (RC) - Refers to performing the same computation that gives the same result.		Basic Flow:
		- The user starts at the beginning of a maze with a bunch of doors that can only be opened by solving puzzles (memory game, etc.).
		- The user also has access to a notepad, where they can write anything (meant to store results of puzzles).
		- The goal is to get through the maze as quickly as possible, with some treasure at the end (golden fish).
		- Later puzzles will use the results from previous puzzles, so the user will either have to have marked that result in their notepad, or go all the way back to that previous puzzle and solve it again.
		- Scoring:
		  - 3 stars: optimal number of transitions between rooms
		  - 2 stars: user makes an error and has a few extra transitions
		  - 1 star: user makes several errors and has many extra transitions
		  - 0 stars: user makes every error possible
			● Adventure Fish Sprite
			● Doors
			● Cave Background
			● Puzzle Station
			  ○ Each of the symbols for the memory game (4?)
			● Golden Fish (Treasure)

Inefficient Iteration (II) - Refers to loops iterating beyond what is necessary to solve the problem or find the necessary value. **For first demo Vinny and Ryan**		Basic Flow: - The user is presented with a series of dirty seashells belonging to a catfish. They are asked to help the catfish clean all of the shells as quickly as possible. - The catfish begins traveling along the path of seashells, partially cleaning each as it passes by. Each shell starts with a dirtiness level of 10, and the catfish has a chance of cleaning between 1 to 4 levels each pass. - As the shells become less dirty, dirt and algae is visually removed from the shell. Once it is fully clean, it begins to sparkle, and the user can click it to place it into the clean shell bag. The clean shell is then replaced by a current, which allows the catfish to swim faster over that spot. - Once all of the shells are clean, the shells do a little dance and the catfish swims up to you and thanks you for this help.- - Scoring: The star rating is determined purely based on the time taken to complete the activity. If the user achieves a perfect time, they are awarded 3 stars. If the time is between perfect and X seconds longer, 2 stars. Between X and Y seconds longer, 1 star. Longer than Y seconds, 0 stars.	• Seashell; random colors, stages of dirtiness (0-10) • Sparkles to indicate dirtiness=0 • Current • Catfish, eating sprites (bird's eye view, MAYBE a standing view) • Arrows to indicate catfish direction • Bag for clean shells • Seabed background

Inefficient API Usage (IAU) - Refers to suboptimal choices in APIs for a given project which add unnecessary bloat. For example, using a class that is designed to handle both date and time when just date would suffice.		Basic Flow:
		- User is presented with the task of toasting a piece of bread
		- User is shown a collection of tools which can be used to toast the bread
		- User selects a tool, and is then shown an animation of the tool being used to toast the bread, along with the result of how efficient that tool was at completing the task
		- Star rating is given based on the tool that the user chooses. 3 stars (toaster), 2 stars (oven, waffle iron, grill), 1 star (campfire, matches), 0 stars (sun, flamethrower).
			• Oven (off/closed, on/open, on/closed
			• Toaster (empty, bread down, toast up)
			  ○ Perfect toast
			• Flamethrower (fish holding flamethrower, flames)
			  ○ Ashes
			• The sun
			• Campfire (fish holding bread on stick, fire)
			  ○ Bread on a stick (on fire)
			• Waffle iron (open, closed)
			  ○ Perfect toast (with black waffle-shaped holes)
			• Grill
			  ○ Perfect toast (with blackened grill lines)
			• Matches (unlit, lit)
			  ○ Bread (with

			burnt circles) • Bread • Old fish (beard, walker)
Inefficient Synchronization (IS) - Refers to inefficiency resulting from different threads in multithreaded applications needing to wait for each other. **For first demo Jonathan and Andrew**		Basic Flow: - Two users are presented with two bowls of mac and cheese but only one fork - Each user can click a specified keyboard button to make their character eat - If the fork is empty, either user pressing their button will draw the fork toward their fish - If the fork has mac from either bowl, it will go to that user's fish - Once the fork places mac in one fish's mouth, it will return to the middle, and will not react to input until it has reached the middle - If the button for player 1 is clicked whilst player 1 is in process of chewing (3 second timer) the fork will be stuck in limbo in front of player 1 until timer ends - Scoring: The star rating is based upon the amount of time taken to completely eat both bowls of mac. Perfect time = 3 stars, etc. (same as II)	• fork (with/without Mac) • Two fish (different colors) + chew animation • Mac bowl (filled, unfilled, quarter-filled, half-filled, ¾-filled)

Redundant Data Processing (RDP) - Describes inefficiency caused by repeatedly converting data from one type to another, rather than working with the original data type directly.		- The user is presented with a fish who is moving from one house to another. They need to move all of their furniture, and would like to do it in as few trips as possible. - The user is shown 50 boxes / pieces of furniture which must be moved. - All of the items are shown in an area on the right, and a side-view of the moving truck is shown on the left. The user can drag the items into the truck in any orientation they like. Once the user is happy with the truck, they can click the “Send” button in the bottom-right to move those items to the other house. The truck will then return, and the user will continue this process until all items have been moved. - Scoring: The user is scored based on the number of trips needed to move all of the furniture. (exact scoring TBD)	● Boxes & furniture (reused from IDS) ● Side-view of the truck ● House (x2, recolored) ● Landscape between houses

Inefficiency under Special Cases (ISC) - Describes general inefficiency in unexpected circumstances, such as when data input is either empty or much larger than anticipated.		- The user is presented with a fish who is about to go fishing. The user is asked to pick 5 bags in which to store the items they catch while fishing. The bags can each hold different types and amounts of items. There are 4 types of bags, the user can pick any amount of each totaling 5 bags.
		- The fish then starts fishing, and catches items (fish, trash, shark, octopus). They will always catch 1 shark OR octopus, 9 fish, and 10 pieces of trash in a random order.
		- Each time the fish catches an item:
		  - If the user has chosen the appropriate bag, the item goes into that bag.
			● Fish bag
			● Trash bag
			● Octopus bag
			● Shark bag
			● Fishing fish
			  ○ Fishing rod and bucket hat on original fish
			● Catching fish
			● Random trash
			● Shark
			● Octopus
			● Background

		- Else, the item goes past all of the bags and back into the water. - Scoring: - The user gains X points for each item successfully put into the appropriate bag. Perfect score = 3 stars, etc.	○ Dock with goo ●

General Inefficient Computation (GIC) - Refers to computational inefficiency that is derived from poor algorithm design.		Basic Flow:
		- The user is helping a shrimp detective (Shrimplock Holmes) try to find as many criminal fish as possible.
		- The user is shown a school of fish (40) along with several (3) options of how to sort them (size, color, name, hats, shape of fins, etc.)
		- After sorting, a timer starts counting down (30 seconds), and the user is shown a particular fish within the school that they must find
		- Once a fish is found, a new fish is shown, etc. and the user must find as many fish as they can within the time limit
		- Scoring: (figure out actual numbers after testing)
		  - 3 stars: X+ amount of fish
		  - 2 stars: X to Y amount of fish
		  - 1 stars: Y to Z amount of fish
		  - 0 stars: Z- amount of fish
			● Shrimplock Holmes
			● Fish Sprites
			  ○ Red
			  ○ Blue
			  ○ Yellow
			  ○ Green
			  ○ A bunch of different colors
			  ○ Hats
			  ○ Different details
			● Background ocean
			● **Handcuffs**

All games			● Stars for star rating (filled, empty)