No libraries or pre-written code is needed.

Difficulty/ possible time spent

1: Simple. May require research if unfamiliar with javascript.

2:  May require more research into the concept itself, as well as javascript.

3: Research and implementation is difficult.

\*: Time consuming.

|  |  |  |  |
| --- | --- | --- | --- |
| Game Aspects | Description | Details | Category |
| runGame() | Used to reposition objects, clear the canvas and draw new items. |  | 2\* |
| Player | A Player class must be created. | Instance variables:  **Int:** lives, fish  **Boolean:** fancySkates  addFish(int numFish)  setFish (int numFish)  getFish() | 1 |
| (within Player class) Lives | The player has a maximum of 3 lives.  A player’s lives cannot go below 0 or above 3. | addLife()    -if lives already at 3, do nothing  subtractLife()     -if lives reaches 0, **game over** (see Winning/Losing)  setLives(int numLives), getLives() | addLife(), setLives, getLives: 1  subtractLife(): may require research on loading a different screen/canvas (2) |
| Fish | A Fish class must be created.  Each fish object is the same. | checkPlayerCollision()  Instance variable: x position  Constructor: position is random (within x bounds of box, y stays the same)  changeX()    -sets x to random number within bounds of box | Creating class, constructor, changeX(): 1  checkPlayerCollision: may require research on collision mechanic itself (2) |
| Display lives/fish | 2 counters should show up at the top of the screen and remain during gameplay. | “Lives: “ + player’s lives  “Fish: “ + player’s fish | 1 |
| Multiplayer | 2 player objects must be created. The controls are instead w, a, s and d.  When one player loses or gains back a life, both do. Fish also increases for both of them.  Both players follow the same pattern of arrow keys. | handlePlayerMovement 2() | 2\*\* |

|  |  |  |  |
| --- | --- | --- | --- |
| Game Mechanics | Description | Details | Category |
| Movement with arrow keys | Function for controlling player’s movement (key down, key up)  If player moves offscreen, they appear at the opposite side. | handlePlayerMovement() | 2\* |
| Arrow sequences for each level/person | Function that draws a random arrow  The arrows stay in the same place for a certain amount of time before moving downward offscreen for a slightly longer amount of time, enabling the player to collect fish between arrows. | generateArrow()     -for random numbers 1-3, draw either a left, right, or up arrow key     -variables: x and y  arrowMovement()     -arrow stays in place for a certain amount of time before it moves downward off the screen     -global variable called activeArrow        -boolean     -set to true when arrow is in place, false when arrow moves | 2\*\* |
| Fish spawning/disappearing | If the player collides with the fish, the fish disappears.  If the fish is collected, a new one is spawned at a random position x.  If it is not, the changeX() function is called, moving it to a random location. |  | 3\*\* |
| Success/failure of the player | Correct key must be pressed before it begins to move. If not, the player loses a life.  If the correct key is pressed, the player either gains a life, or stays at a maximum of 3 lives. | If: correct arrow key not pressed while activeArrow is true, subtractLife()     -red “-1” displayed next to Lives  Else: addLife ()     -green “+1” displayed | 3 \*\* |
| Regaining lives with fish | In runGame(), if lives of any player is equal to zero, they get back a life if their fish is above a 1.Their fish is reduced by 1. |  | 2 |
| In between levels:  Shop, gaining more fish | “In between levels” screen: displays player’s lives at end of the level.  However many lives leftover is added to the player’s fish.  Player’s variable lives is set to 3.  Right arrow key to shop screen  Player.setLives(3)  If enough fish for skates: “Buy skates? (y/n)”  -if “y”: update player’s sprite to one with skates.  Else: “Save up fish to buy skates!”  Press right arrow key to continue to next level |  | 1\*  Shop feature: not feasible, leave for last to complete if there is time. |
| Winning/Losing | If at any point lives = 0, Game Over screen  If all levels completed, You Won screen | Both black screens that say either “Game Over” or “You Won” | 1 |