Girls Who Code At Home

Meteor Catcher Game: Part 1

Reference Guide

Meteor Catcher Game: Part 1 - Reference Guide

In this document you will find all of the answers to some of the questions in the activity. Follow along with the activity and when you see this icon, stop and check your ideas here.

Step 1: Identify the Parts of A Game

The parts of a meteor catcher game

➔ Describe the **goal** of Meteor Catcher, the game you just played in the last step. What does a player or team have to do to win the game?

The goal of Meteor Catcher is to catch as many meteors as possible. Right now, this game doesn't have a very defined goal. We left it open so you can customize your own after you build the base game.

➔ The **components** of Meteor Catcher include the meteor, catcher, walls, and player. Each component has unique properties (e.g. size, color, shape, etc) and actions (i.e. the things it does - the verbs you associate with that component) that contribute to the game's system. For example, a meteor property would be round and a meteor action would include falling from top of screen to the bottom.

*Spend **2-3 minutes** thinking about the properties and actions for each component in the table below.*

COMPONENT	PROPERTIES	ACTIONS
What are the essential pieces for play?	*What are the **attributes** or **characteristics** of the component?*	*What does it **do**? What **verbs** do you associate with it?*
meteor	➔ Round ➔ Teal ➔ Random diameter between 10 and 40 pixels	➔ Fall from the top of the screen to the bottom ➔ Moves at a random speed ➔ Starts in different locations at the top of the screen ➔ Can intersect with the catcher ➔ Can intersect with the ground (i.e. bottom of screen)
catcher	➔ Round ➔ Transparent white ➔ 40 pixels in diameter	➔ Follows the mouse. ➔ Can "catch" or collect meteors by intersecting with them.
walls	➔ 400 pixels in width ➔ 400 pixels in height	➔ Intersects with meteors. ➔ The bottom wall causes a new meteor. to appear if a meteor touches it.
player	➔ Likes space! And Meteor showers!	➔ Moves the mouse to catch the falling meteors.

➔ Describe the **space** of the game. Where does it take place? (Note that sometimes the space can be more than one thing. For example, chess takes place on the chess board, but it also takes place in a living room, park, or cafeteria.)

The game takes place in a 400 by 400 square window on a webpage. From a story perspective, it takes place in actual outer space.

➔ Define the **challenge**. What obstacles are in the player's way of reaching the goal?

The challenge is to catch the falling meteors. Each one starts at a different location, falls at a different speed, and renders at a different size each time a new meteor appears on screen.

➔ Describe the game's **core mechanic**. What core actions or moves does the player need to make to play the game? What core actions or moves does the player do to power the play of the game?

Players catch meteors. They move their mouse around the screen to collect them.

➔ Write a list of the game's **rules**. Rules determine what we can and cannot do in our game. They can be applied to players, components, the space, etc.

- Meteors fall from the top of the screen to the bottom.
- Only one meteor falls at a time.
- The catcher follows the mouse.
- The player's catcher must intersect with the meteor in order to catch the meteor.
- If a meteor is caught, it disappears and a new meteor is drawn at the top of the screen.
- If a meteor touches the bottom of the screen, it disappears and a new meteor is drawn at the top of the screen.

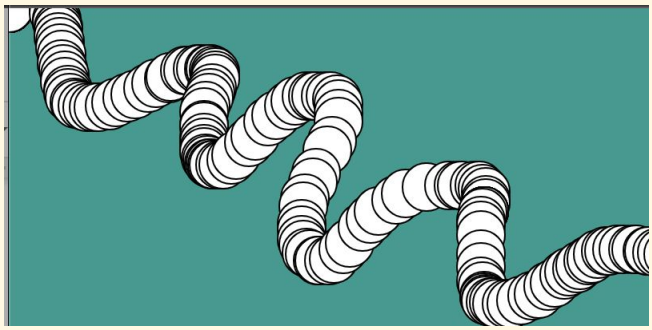
Step 2: Write pseudocode for your game

```
Declare any variables

DO THIS ONCE
  Set the size of the canvas to 400 pixels by 400 pixels

DO THIS EVERY LOOP
  Set background color
  Draw the meteor
  Make the meteor fall
  Draw the catcher to follow the mouse
  Find/Calculate the distance between meteor and the catcher
  Test to see if meteor and catcher have intersected. If they intersect,
    Redraw the meteor at the top of the screen to a random location,
    Give it a new speed,
    Set a new diameter.
  Test to see if meteor and bottom wall have intersected. If they intersect,
    Redraw the meteor at the top of the screen to a random location,
    Give it a new speed,
    Set a new diameter.
```

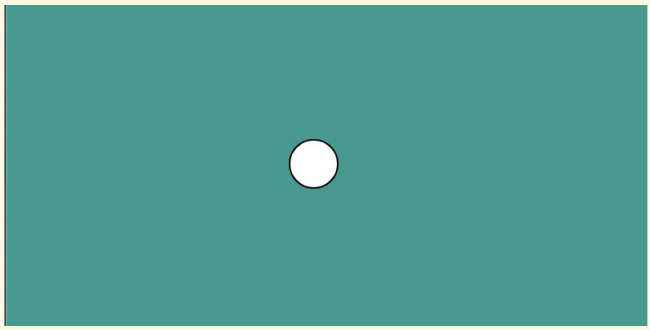
***Remember**: There is more than one way to write a program, so there is more than one way to write your pseudocode as well!*

Step 5: Learn about program flow

SKETCH 1



SKETCH 2



Sketch 1 has the `background()` function in `setup()`. This means that the background is only drawn one time. Since the `ellipse()`function is in `draw()`, p5 draws a new circle at the mouse position on the screen every time the program runs through a loop. You could describe it like this: Fill background, draw circle, draw circle, draw circle, draw circle, draw circle, etc.

```
// Sketch 1
function setup() {
  createCanvas(400, 400);
  background(220);
}

function draw() {
  ellipse(mouseX, mouseY,50,50);
}
```

Sketch 2 has the `background()` function in `draw()`. This means the program draws the background and the circle every time the program loops through. This gives the appearance that the circle moves smoothly through space as we move the cursor, even though the program is drawing new circles just like in the first sketch. You could describe it like this: Fill background, draw circle, fill background, draw circle, fill background, draw circle, fill background, etc.

```
// Sketch 2
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  ellipse(mouseX, mouseY,50,50);
}
```

Step 6: Check for Understanding

The actions in `setup()` only need to happen one time since you are only making one batch. The actions in `draw()` need to happen for each dumpling you make. Since you are making multiple dumplings, we place these actions in `draw()`.

```
setup() {
  Measure filling ingredients
  Mix filling ingredients
  Collect dumpling wrappers
}

draw() {
  Spoon filling into wrapper
  Close wrapper
  Place dumpling in pan
  Cook dumping
  Remove dumpling from pan
  Eat dumpling
}
```