Hour of Code: Code the Eclipse

Intermediate Tutorial

Big Idea To solve a problem, break it down into smaller parts.

Module Code the Eclipse: https://app.vidcode.io/project/hoc-eclipse

Time **1 hour**
10 minutes background
45 minutes coding
5 minutes reflection

Standards

NGSS-MS-ESS1-1 Develop and use a model of the Earth-sun-moon system to describe the cyclic patterns of lunar phases, eclipses of the sun and moon, and seasons.
NGSS-HS-ESS1-4 Use mathematical or computational representations to predict the motion of orbiting objects in the solar system.
NGSS-MS-ETS1-1 Define the criteria and constraints of a design problem with sufficient precision to ensure a successful solution, taking into account relevant scientific principles and potential impacts on people and the natural environment that may limit possible solutions.
NGSS-MS-ETS1-2 Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.

NGSS-MS-ETS1-3 Analyze data from tests to determine similarities and differences among several design solutions to identify the best characteristics of each that can be combined into a new solution to better meet the criteria for success.
NGSS-MS-ETS1-4 Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved.

CSTA 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.
CSTA 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
CSTA 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
CSTA 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.

Background (10 minutes)

Let's code a simulation of a solar eclipse! For this simulation, we'll take the point of view from the Earth, assume the sun is stationary, and model how much light is cast on Earth as the moon moves across the sun.

If you have time, have your class consider the Earth-moon-sun system using the Parts-Purposes-Complexities thinking routine from Harvard's Project Zero.
http://www.pz.harvard.edu/resources/parts-purposes-complexities-0

Parts: (Objects and Properties)
Sun, Moon, Earth

Purposes: (Methods)
Sun: Cast light
Moon: Move relative to the Earth and Sun, obscure light
Earth: Reflect light

Complexities: (Loops, Conditionals, Method calls)

- The amount of shade cast on the Earth is a function of the relative positions of the sun and moon.
- When the moon completely obscures the sun (a total eclipse), there is a corona effect.

All of this will be covered in the online tutorial, but students will get more out of it if they come prepared.

Code Challenge (45 minutes)

Students will create a simulation of a solar eclipse.

Sample Solutions
Simple: https://app.vidcode.io/share/YPpdJN1KEy
Advanced: https://app.vidcode.io/share/e5pVFZyN2w

```
// Part: Earth
function earth(){
  movie = video();
  movie.source = "night-stars.mp4";
  this.shadow = tint("black",0);

  // Purpose: be lit or shaded
  this.dim = function(perc){
    this.shadow.amount = perc;
  }
}

// Part: Sun
function sun(x,y) {
  this.x = x;
  this.y = y;
  this.body = circle(this.x,this.y, 100, "#eff28c", "clear");
  this.label = text("sun",this.x-40, this.y-40);
  this.label.color = "#1d0c7b";
  this.corona = blur(0);

  // Purpose: totality
  this.total = function(){
    mySun.body.radius = 110;
    this.corona.amount = 20;
  }

  // Purpose: partiality
  this.partial = function(){
    mySun.body.radius = 100;
    this.corona.amount = 0;
  }
}
```

```
//Part: Moon
function moon(x,y) {
  this.body = circle(x,y,100, "blue", "clear");
  this.label = text("moon");
  this.label.y = y-40;
  this.label.color = "white";

  // Purpose: transit
  this.goto = function(x){
    this.body.x = x;
    this.label.x = x-60;
  }
}

// Instantiate parts
var myEarth = new earth();
var mySun = new sun(300,150);
var myMoon = new moon(0,150);

// Complexities
repeat(function() {
   // The Moon is controlled by the mouse
   myMoon.goto(mouse.x)
   // The amount the Earth is lit depends on the positions of the Sun and
Moon
   myEarth.dim(100 - Math.abs(myMoon.body.x-mySun.body.x)/2);
   // If the Sun and Moon are lined up, we have a total eclipse
   if (myMoon.body.x === mySun.body.x){
     mySun.total();
   }
   else{
     mySun.partial();
   }
}, 1);
```

What You Learned

JavaScript is based on objects. Almost anything can be an object. Objects have **properties**, which are things ABOUT or BELONGING TO them, and **methods**, which are actions that object can DO.

To define an object, we use an **object constructor** function. It is just a function that says what the properties and methods of an object are.

To define properties and methods, we use the keyword **this**.

```
function sun(){ <- start object constructor
  this.shape = circle(200, 200, 100, “yellow”); <- property
  this.label = text(“sun”, 200, 200); <- property

  this.corona = function(){ <- start method
    this.shape.radius = 110;
  } <- end method
} <- end object constructor
```

To instantiate an object, we use the keyword **new**.
var mySun = new sun(); <- instantiate a sun object and call it mySun

To get at properties of the object, we use **dot notation**. The dot can be thought of as showing possession (‘s).

`mySun.label.color = “blue”;` <- the sun’s label’s color is blue

We also use dot notation to **call** a method.

`mySun.corona();` <- run the sun’s corona method with no arguments

Remember, it’s not enough just to define an object with an object constructor function, you have to instantiate it with **new. Also, use the variable name you supplied when you instantiated your object to reference any properties or call any methods.

`sun.corona();` <- will not work! The name of your instantiated sun is mySun.

Objects are useful because they can be reused! If you wanted to, you could instantiate multiple suns, give them all different names, and control them separately.

Sharing (5 minutes)

Because most student projects will be more or less identical, sharing in class is not vital. Encourage students to copy the link to their published project to share with their families. You may want to put it in an email or Google doc.

Reflection (5 minutes)

Questions to be answered in discussion or essay form:
What is the difference between a property and a method?
What keywords does JavaScript use, and what do they mean? (new, this, var, if, else, function)
What do the different colors in your program mean?
How do you choose good names for properties, methods, objects, and variables?

Extensions (Beyond an Hour)

Most students will immediately propose improvements to this simulation. Some will need encouragement. Use the reference tab to learn how to do more things. When in doubt, just Google the problem. You know all the search terms to use: JavaScript, object, method, function, instantiate, object constructor, this, new, var.

Now that you've made a simple simulation, what else can you do?

- Create a new simulation from the point of view of the sun.
- Add comments that explain what each part of the program does.
- Show the corona for a small range of inequalities, instead of just one equality.
- Use graphics instead of shapes for the sun and moon.
- Shoot your own video of people watching the eclipse.
- Add labels for each of the phases of the eclipse.
- Make the moon move automatically, instead of following the mouse.
- Use the arrow keys to control lunar transit.
- Total Eclipse of the Heart?