Lesson 4

Add-ons

Pre-Lesson Ideas:

* Use printed grid paper and have students draw and label a sketch of what they want to draw.
* Then, require them to get approval of their hand drawn sketches, before they can start programming.
* Have younger students create their landscape in beginner. This might be a good space to practice layering and design commands.

Post-Lesson Ideas:

Reflection Questions

1) What landscape did you pick and why?

Possible Answers: I chose to make a sunset because I wanted to work with layering of colors.

2) Did you have a communication mix up with the computer?

Possible Answers: I wanted to fill my sun red, but put the fill command after the eelipse command.

3) Did you use any commands in a way you hadn't before?

Possible Answer: I added another parameter to round out my rectangles to make more realistic clouds.

Further Development

* Give students a picture of a landscape and have them try to replicate it. Challenge students to use commands in new and creative ways to get the correct shapes.

Lesson 4

Set Up

The Landscape Project:

* Practice geometric layering and visual design with learned commands to create detailed and realistic landscapes.

Project Goal:

1) Generally, what should the project look like?

A complex and detailed landscape, like a mountain range or beach. Landscapes can also be make believe, like a magical forest.

2) What skill(s) are being learned/ practiced?

Navigating the coordinate grid

Geometric shapes & special reasoning

3) What concept are students gaining insight on?

Order matter - Each command layers on top of the other in the order that it is written.

Develop and use abstraction

Communicate Precisely in order to relay your intentions as intended

Programming/ Math Vocabulary:

Comment - A piece of specially tagged explanatory text within (a program). It is used to assist other users and organize the code.

Lesson 4

Outline

Introduction to Topic:

“Today we are going to be (making our own landscape). Talk about how these can latter be used in the backgrounds of games.

Project Breakdown:

1) Plan out your drawing

2) Program basic shapes to create outline (in advanced)

3) Use design commands to add details to the shapes

4) Problem solve and trouble shoot errors

Example Projects/ Basic Source Code:

```
draw = function( ) {
noStroke();
background(255,51,0);
//sun
fill(255, 100, 0);
ellipse(350, 350, 200, 200);
//3 layers of mountains
fill(102,51,0);
ellipse(100, 350, 200, 50);
ellipse(550, 350, 1000, 50);
rect(0, 350, 1000, 50);
fill(120, 51, 0);
ellipse(700, 400, 200, 105);
rect(0, 400, 1000, 500);
fill(150, 51, 0);
rect(0, 500, 1000, 200);
//clouds
fill(230,30,10);
ellipse(650, 250, 250, 50);
ellipse(550, 250, 400, 25);
ellipse(700, 275, 400, 25);
};
```

Lesson 4

Troubleshooting

Common Mistakes and Confusions:

1) Misspelling Commands

It is really easy for students to make silly mistakes, when they are working with a subject completely unfamiliar to them. Pay special attention to background, students often forget the g.

2) Missing capitals where appropriate

Students constantly forget to capitalize the W in strokeWeight( );

3) Putting a number in noStroke( );

noStroke is a function with no parameters, so nothing needs to go inside of the parentheses.

4) Ordering their shapes incorrectly

If you want a shape to be drawn ontop of another, you have to write the command after.

FAQ's:

1) How do I round corners?

Add more parameters to the rectangle command. That is the only shape you can round corners on.

2) How do I get rid of lines to cleanly combine shapes?

Use noStroke before the combination of shapes you are looking to combine. If you need outlines later in your project, add stroke(0); before the commands where you need outlines.