Unit 3 Lesson 1

Programming for Entertainment

Resources

Name(s)_____ Period _____ Date _____

Activity Guide - CS in Entertainment

Entertainment Exploration

Computer Science has played a huge role in the evolution of many (if not all) fields of entertainment. To better understand how CS has changed these fields, and to get a feel for how you might use programming to create entertainment with your programming skills, you're going to look into some of the ways people are developing software that entertains.

Topics (check the one you've selected)

- ❑ **Movies and Television** (eg. programming 2d or 3d animations, computer generated special effects)
- ❑ **Music** (eg. computer generated music, digital instruments)
- ❑ **Games** (eg. video game programming, animation)
- ❑ **Art** (eg. interactive art, algorithmic art)
- ❑ **Other** (describe) _____

Researching your Topic

With your chosen field as guidance, go online to search for how computer science has impacted your field of entertainment. Try to focus on how CS or programming is used to *create* entertainment, instead of just places where computers are *used* in entertainment. Head to Code Studio for a list of useful sites to begin your search.

As you move beyond the provided sites, consider using the following patterns to find information (use your topic in place of the blank):

Programming for _____
Computer Science in _____
Algorithmic _____
Creating _____ with code

Use the space below to record notes about interesting products you find, patterns that you're seeing, or problems within your chosen topic that people are trying to address.

Research Notes

Interesting Information

Based on the research your group did on the last page, select **one** of the uses of CS you found to focus on. Answer the following questions for your chosen use.

You may need to head back online to gather more details about your chosen use.

What Problem Does it Solve?

Why is CS used in this way? How is it solving a problem in the creation of entertainment?

How is it an Improvement?

What makes the use of CS better than prior approaches? Does it allow creators to do things easier, faster, or in ways that were impossible before?

An Interesting Fact or Use

What's an interesting fact about the use of CS in this field, or an interesting product that was created with CS?

An Open Question

What questions are you left with after exploring the use of CS in your chosen field?

Unit 3 Lesson 2

Plotting Shapes

Resources

Name_____ Period _____ Date _____

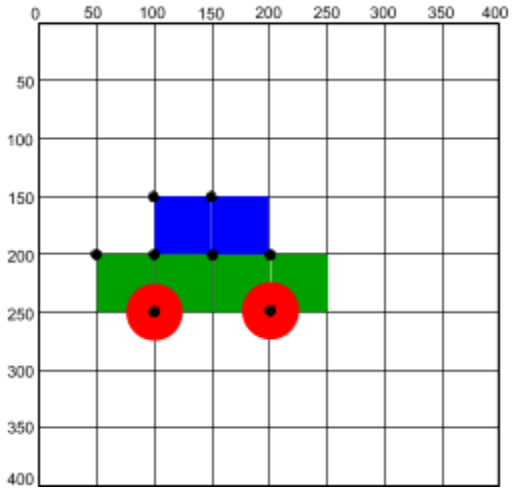
Activity Guide - Plotting Shapes B

IMPORTANT!! DON'T LET YOUR PARTNER SEE THIS PAPER!

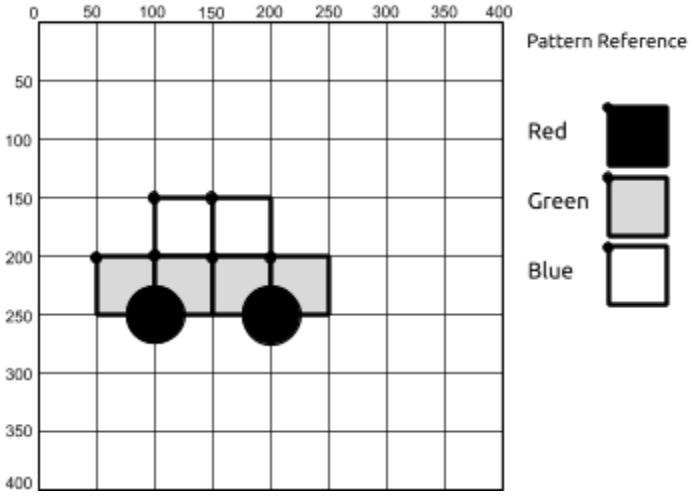
Overview

Your partner should have the Drawing Tool open on a computer where you cannot see it. Alternate turns trying explaining how to draw your image. Afterwards check their work but make sure to keep your drawings hidden.

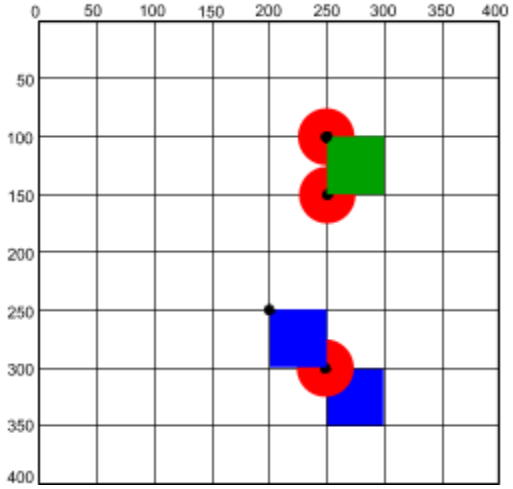
Drawing 1B (Color)



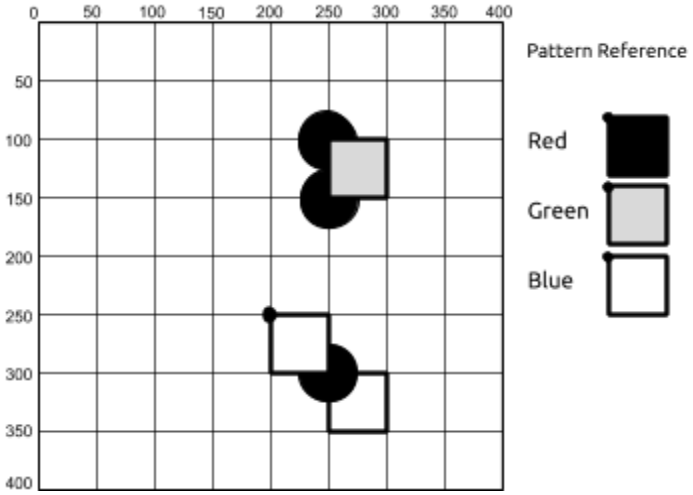
Drawing 1B (Black and White)



Drawing 2B (Color)

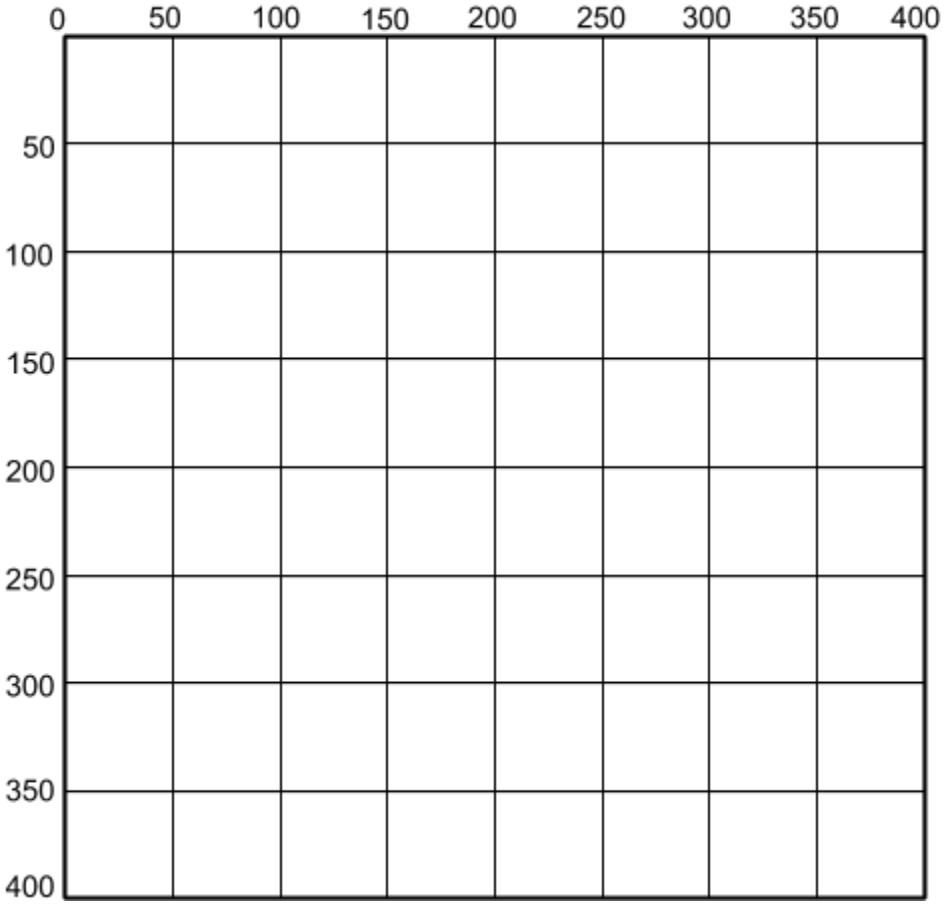


Drawing 2B (Black and White)



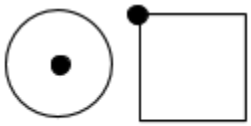
Draw Your Own

Use the space below to draw your own image with the shapes. Then see if you can communicate it to your partner to draw using the shape drawing tool in Game Lab. You can also give your drawing to another group to use as a challenge.



Shape Size

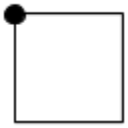
These shapes are the correct size



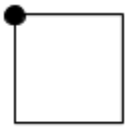
Pattern Reference

If you don't have red, green, or blue to draw with fill in the patterns or colors you'll use instead

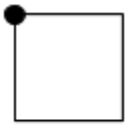
Red



Green



Blue



Reflect

What things were important in communicating about position, color, and order of the shapes in this activity?

What's another way you have seen similar problems solved in the past?

Name_____ Period _____ Date _____

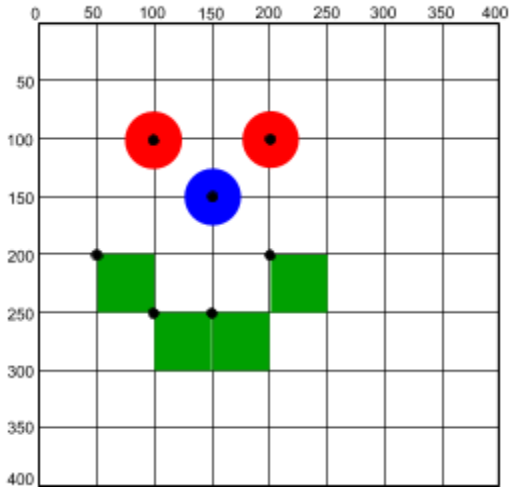
Activity Guide - Plotting Shapes A

IMPORTANT!! DON'T LET YOUR PARTNER SEE THIS PAPER!

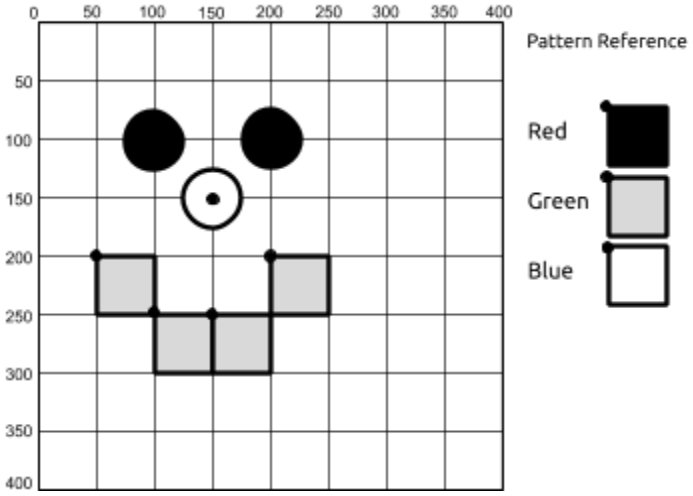
Overview

Your partner should have the Drawing Tool open on a computer where you cannot see it. Alternate turns trying explaining how to draw your image. Afterwards check their work but make sure to keep your drawings hidden.

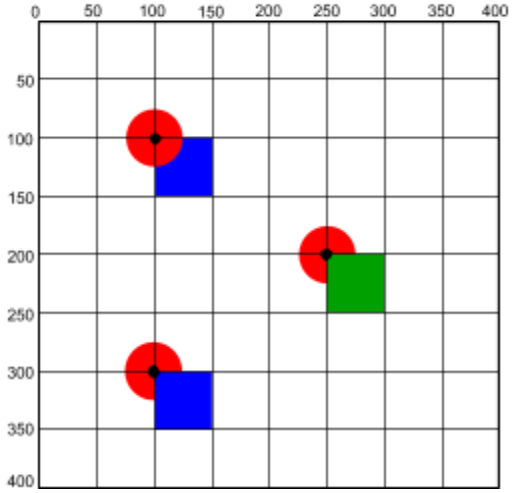
Drawing 1A (Color)



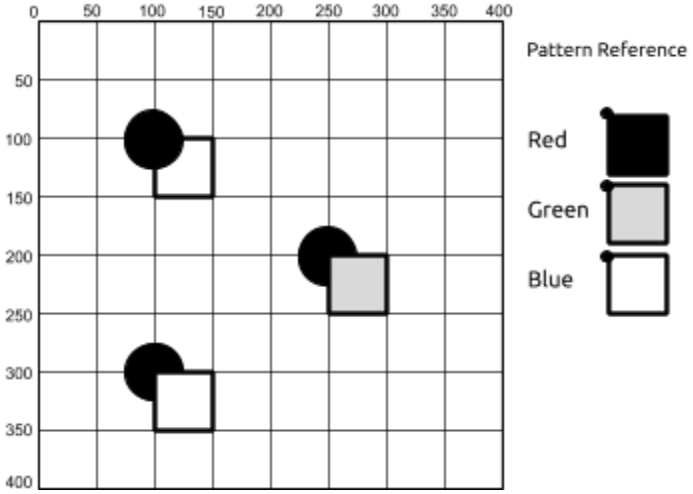
Drawing 1A (Black and White)



Drawing 2A (Color)

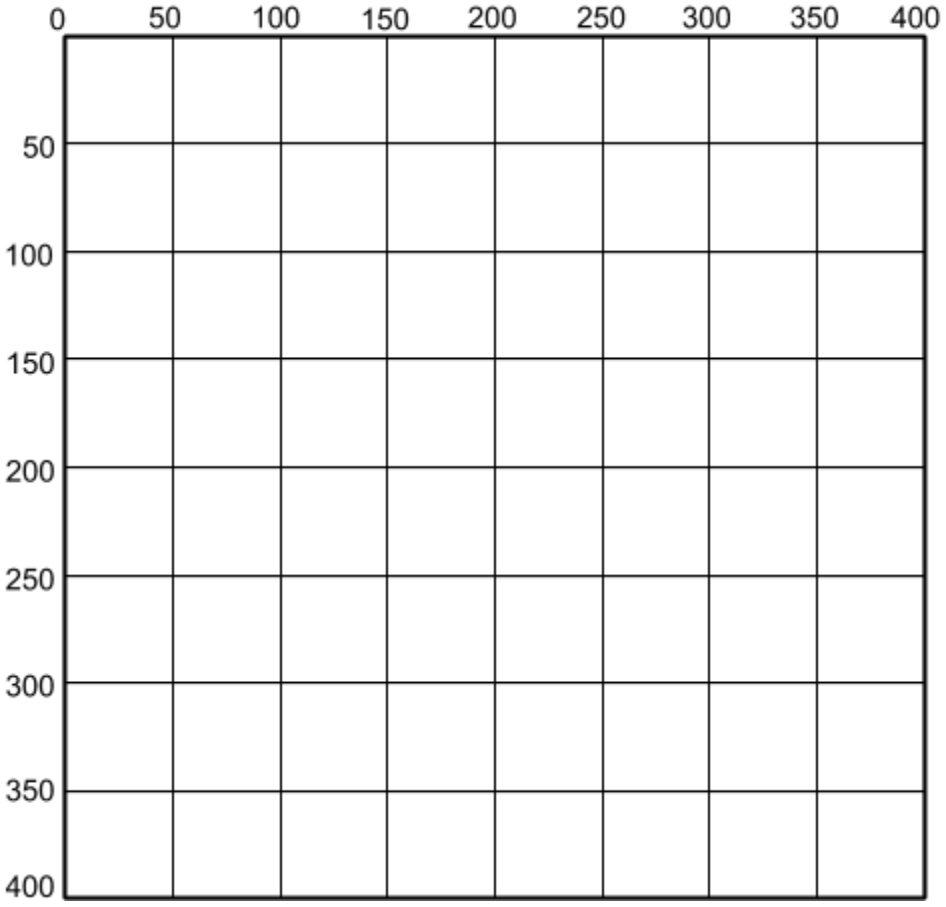


Drawing 2A (Black and White)



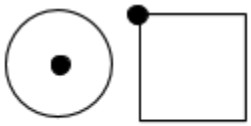
Draw Your Own

Use the space below to draw your own image with the shapes. Then see if you can communicate it to your partner to draw using the shape drawing tool in Game Lab. You can also give your drawing to another group to use as a challenge.



Shape Size

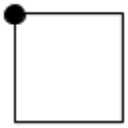
These shapes are the correct size



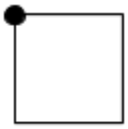
Pattern Reference

If you don't have red, green, or blue to draw with fill in the patterns or colors you'll use instead

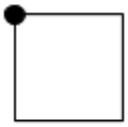
Red



Green



Blue



Reflect

What things were important in communicating about position, color, and order of the shapes in this activity?

What's another way you have seen similar problems solved in the past?

Unit 3 Lesson 3

Drawing in Game Lab

Resources

Unit 3 Lesson 4

Shapes and Randomization

Resources

Unit 3 Lesson 5

Variables

Resources

Unit 3 Lesson 6

Sprites

Resources

Name(s)_____ Period _____ Date _____

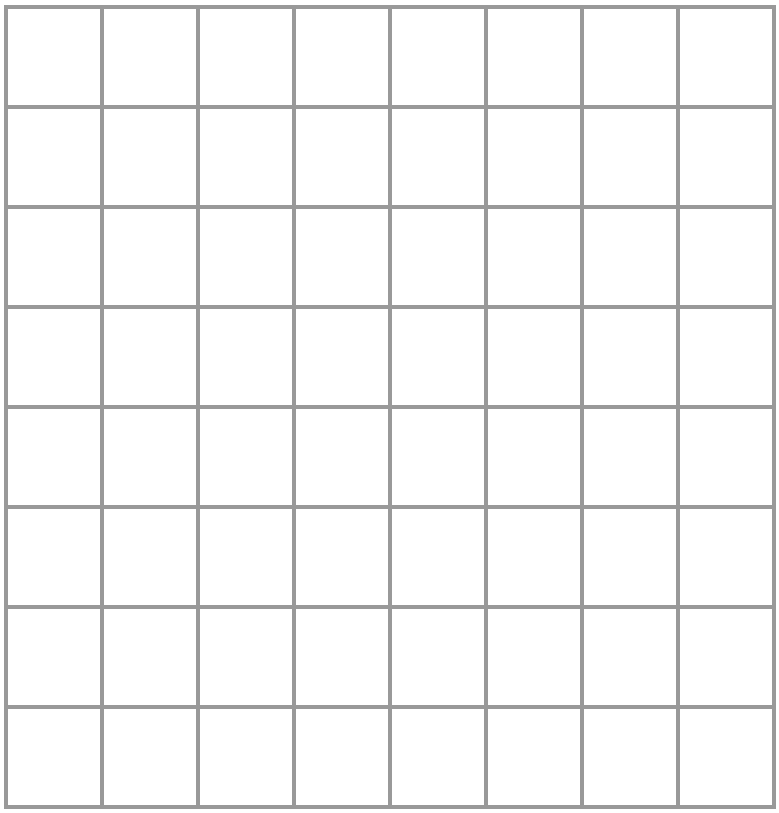
Activity Guide - Sprite Scene Planning

Sprite Scene Planning

Using shapes, sprites, and text, you're going to create a simple scene. You can think of this like a page in a story, a panel in a comic, or just a standalone scene.

Sketch your Scene

The first thing to consider when designing your scene is what your background will look like. You can use the drawing commands that you've used in the past to layout a simple background over which you will place your sprites.
The space below is 50 by 50. That means each square on the paper will map to 50 pixels on the computer. Sketch out your background using only the drawing commands (reference provided to the right). List the sprites you'll use below.



Shapes:
background(color)
rect(x, y, width, height)
ellipse(x, y, width, height)
line(x1, y1, x2, y2)
text(string, x, y, width, height)
textSize(pixels)
Color and Style:
fill('color')
noFill()
stroke('color')
noStroke()
strokeWeight()

Sprite Label	Description

Unit 3 Lesson 7

The Draw Loop

Resources

Unit 3 Lesson 8

Counter Pattern Unplugged

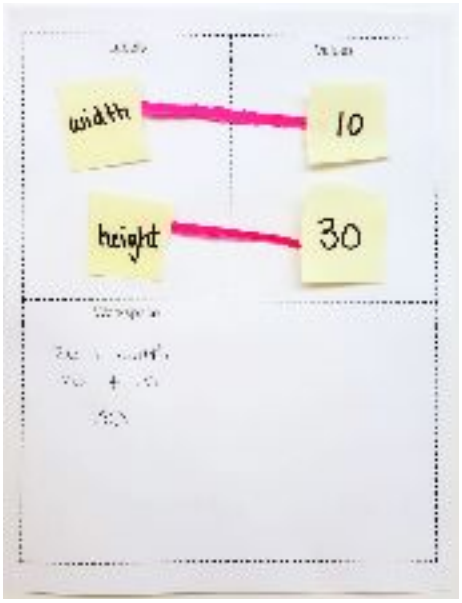
Resources

Name(s)_____ Period _____ Date _____

Activity Guide - Variables Unplugged

The Activity

You will work with your partner to connect variable labels and their values based on different programs.



Setup

- **The Board:** A piece of paper split into 3 sections: Labels, Values, and Workspace
- **Variable Label and Value Cards:** Small pieces of paper (~50)
- **Connectors:** Something to connect the labels and values

Rules

1. Label cards must always have one and only one connector
2. You may not create the same label card twice
3. You must create a new value card for every "=" command
4. Value cards can only have one number written on them.
5. Any value card not connected to a label card at the end of a command is thrown away.

Commands

Run the programs in order by following the steps for each command.

Step 1: Find or create the label card and connector

A. If the command starts with "var", **create a new** label card with the word that comes before the "="
   a. You should also give it a connector, which you will use later
B. If the command doesn't start with "var", **find** the label card with the word that comes before the "="

Step 2: Calculate the number for your value card

A. Copy what's on the right of the "=" to your Workspace
B. Cross off any variable labels, and replace them with the values (numbers) connected to them
C. Do any calculations (addition, subtraction) on the numbers
D. Write down the final number on a **new** value card

Step 3: Connect your label card and value card

A. Place your new value card in the "Values" section
B. Move the connector of the label card from Step 1 to connect to this new value card

Step 4: Remove leftover values

A. Take any value cards that are not connected to label cards, and throw them away.

Try It Out!

Run the programs below. To help you along in the first two programs, check out the before and after examples to the right which show how different commands run.

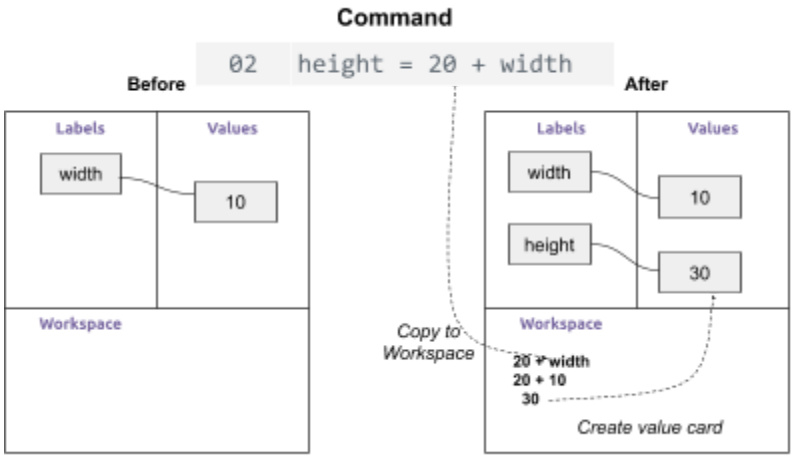
Program 1

```
01  var width = 10
02  var height = 20 + width
```

Ending State

width:

height:

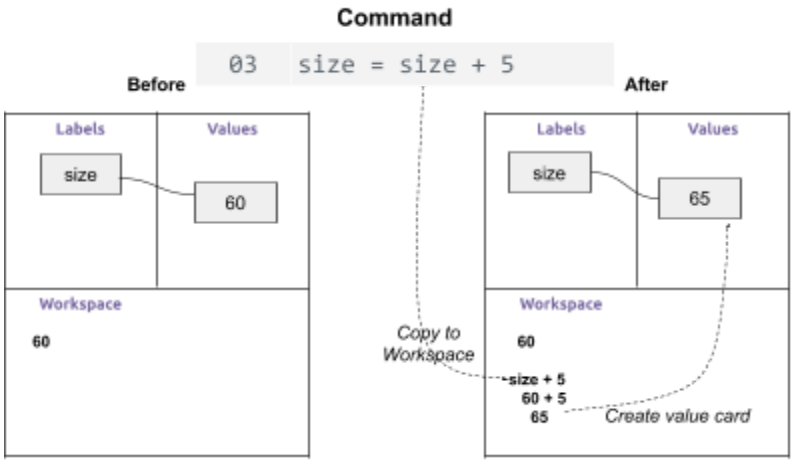


Program 2

```
01  var size = 30
02  size = 60
03  size = size + 5
```

Ending State

size:



Program 3

```
01  var age = 11;
02  var height = 60;
03  age = age + 1;
04  height = height + 5;
```

Ending State

age:

height:

Program 4

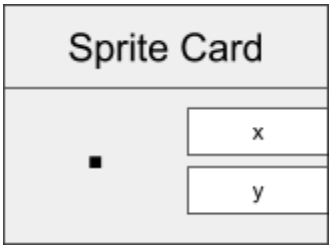
```
01  var xPosition = 100
02  var yPosition = xPosition
03  xPosition = yPosition + 30
04  yPosition = yPosition + 50
```

Ending State

xPosition:

yPosition:

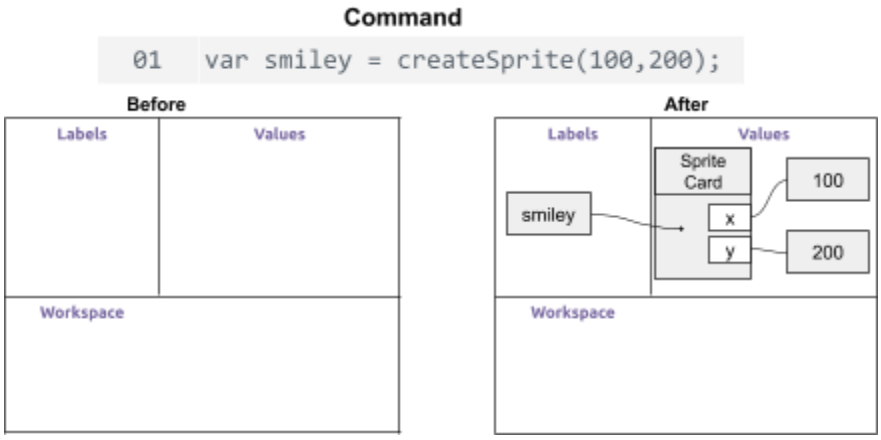
Sprite Properties



You can change your sprite properties in the same way that you change your variables, which allows you to control how the sprite moves across the screen.

To keep track of a sprite's properties, you'll need a sprite card. The sprite card goes in the values section, connected to a variable label. On the bottom of the card, list the properties that you want to keep track of. **You will use one connector for each property.**

In the example to the right, the card keeps track of the x and y properties of the sprite, which tell the computer where to place it on the screen.

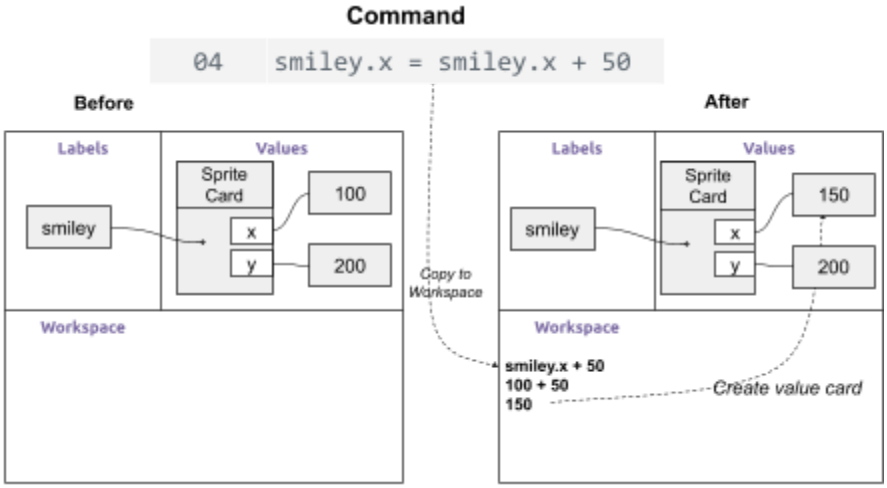


Creating a Sprite Card

Every time you see the `createSprite` command, you'll need to create a new sprite card with the properties you want to keep track of.

Using Sprite Properties

You can use your sprite property values the same way you used your variable values.

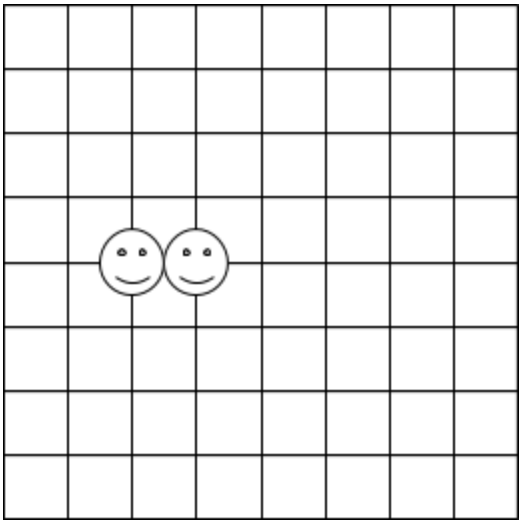


Drawing Your Sprite

Whenever you see the `drawSprites` command, draw your sprite on the grid according to its x and y coordinate values.

You can use a smiley face for your sprite's animation.

You can make your own sprite cards or cut out the ones at the end of this activity guide.



Program 5

```
01  var smiley = createSprite(100,200);
02  smiley.setAnimation("smileyFace");
03  drawSprites();
05  smiley.x = smiley.x + 50;
06  drawSprites();
```

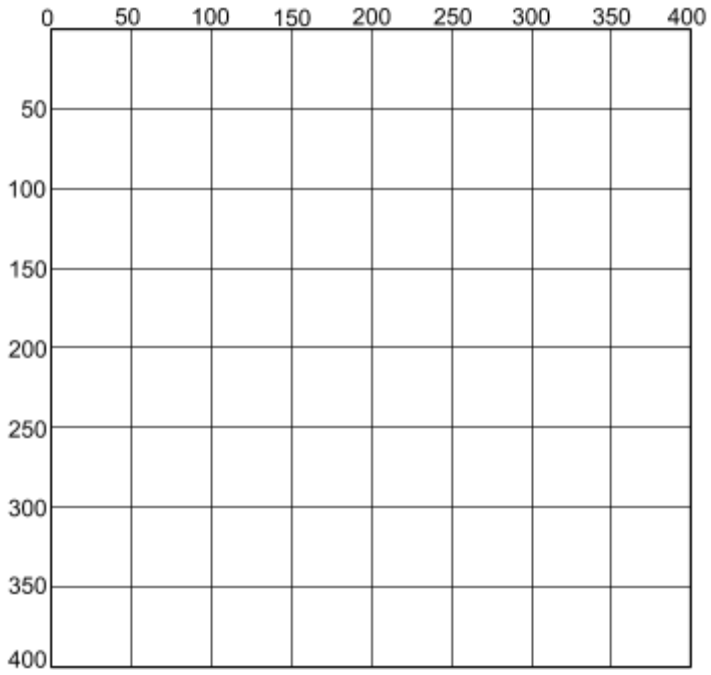
Try it Out!

Run these programs with your partner. Don't forget to draw your sprites when you see the `drawSprites` command.

Program 6

```
01  var smiley = createSprite();
02  smiley.setAnimation("smiley");
03  smiley.x = 50;
04  smiley.y = 100;
05  drawSprites();
06  smiley.x = smiley.x + 50;
07  drawSprites();
08  smiley.x = smiley.x + 50;
09  drawSprites();
10  smiley.x = smiley.x + 50;
11  drawSprites();
```

Ending State
smiley.x
smiley.y



How did the sprite move across the grid in Program 6?

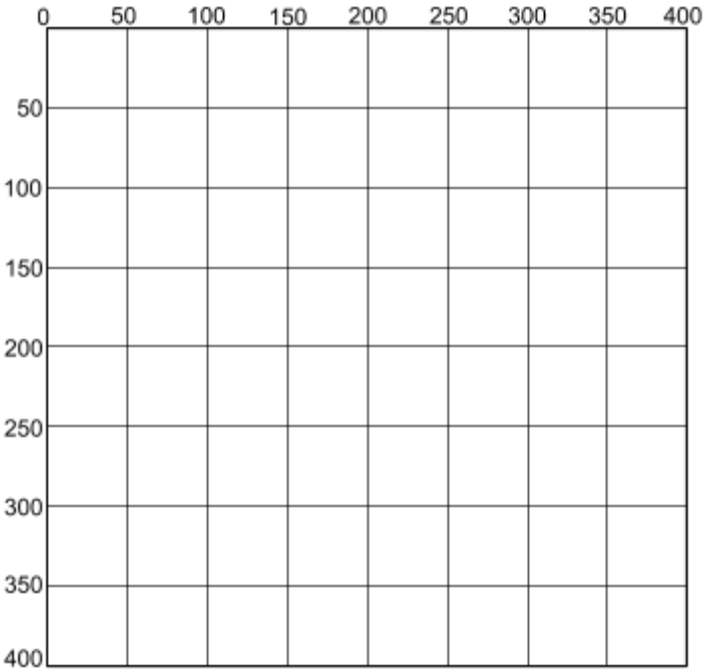
_____

_____

Program 7

```
01  var smiley = createSprite();
02  smiley.setAnimation("smiley");
03  smiley.x = 200;
04  smiley.y = 300;
05  drawSprites();
06  smiley.y = smiley.y - 30;
07  drawSprites();
08  smiley.y = smiley.y - 30;
09  drawSprites();
10  smiley.y = smiley.y - 30;
11  drawSprites();
```

Ending State
smiley.x
smiley.y



How did the sprite move across the grid in Program 7?

_____

_____

---------- Cut out the sprite cards below ----------

Sprite Card
■

Sprite Card
■

Sprite Card
■

Labels

Values

Workspace

Unit 3 Lesson 9

Sprite Movement

Resources

Unit 3 Lesson 10

Booleans Unplugged

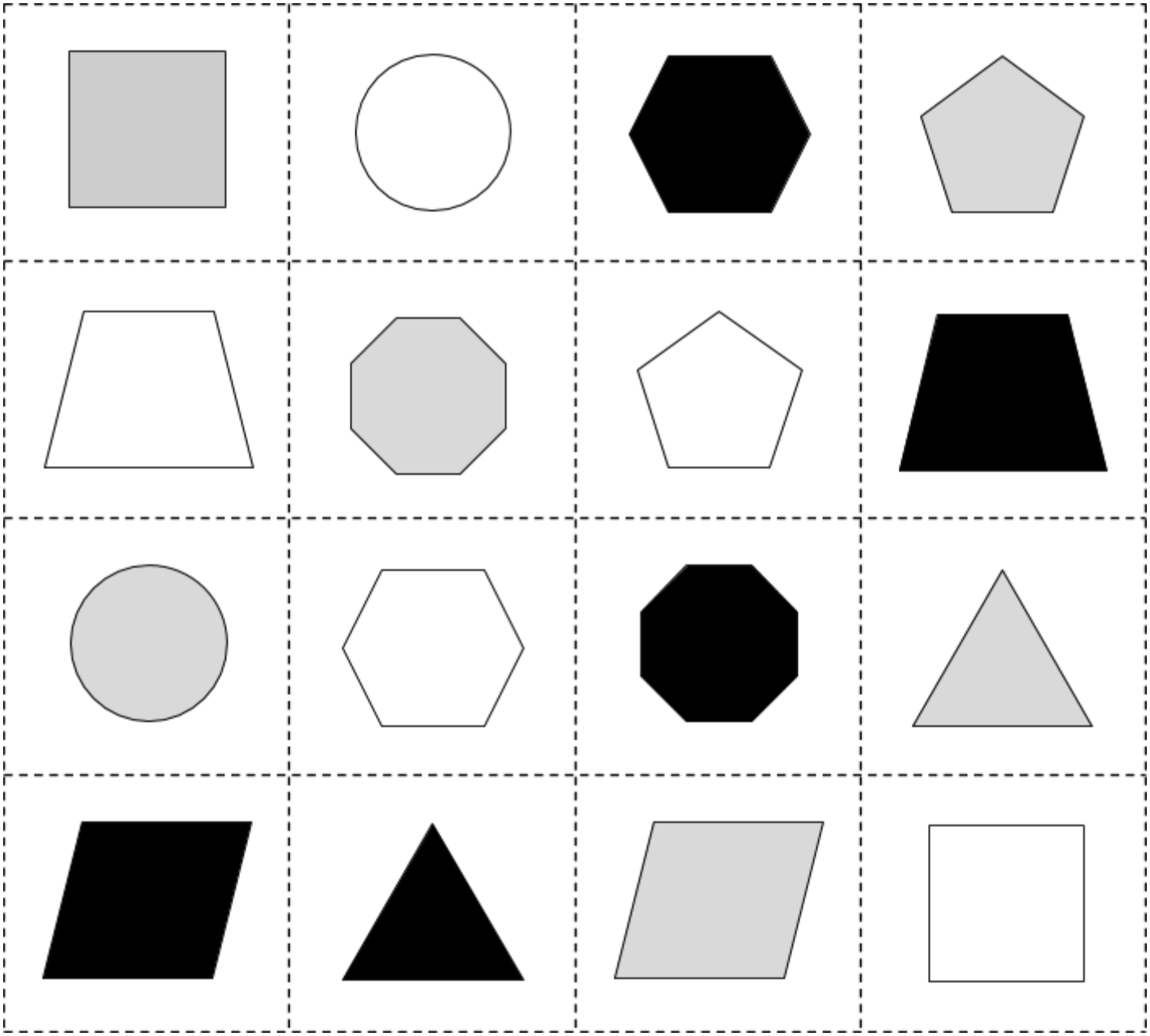
Resources

Name(s)_____ Period _____ Date _____

Activity Guide - Boolean Properties

Prepare Your Shapes

Cut out the following grid of shapes. These will be the cards that you will use for the Boolean Properties sorting activity



Unit 3 Lesson 11

Booleans and Conditionals

Resources

Unit 3 Lesson 12

Conditionals and User Input

Resources

Unit 3 Lesson 13

Other Forms of Input

Resources

Unit 3 Lesson 14

Project - Interactive Card

Resources

Name(s)_____ Period _____ Date _____

Project Guide - Interactive Card

Overview

You're going to be developing an interactive digital card to share with someone you care about, but you'll need to do some planning before you start to program.

Planning your Sprites

Use the table below to plan out your sprites. (You need at least three, but you can use as many as you want!) Next to each sprite, plan out the image it will use and which properties will be changing.

Sprite Label	Image(s)	Properties

Developing Interactions

The final element of your card to consider is how the user will interact with it, and how the sprites may interact with each other. You'll want to include conditionals that respond to keyboard input (such as `keyDown()`) as well as conditionals that respond to changing variables or sprite properties (such as `sprite.y > 300`). Use the table below to plan out all of your conditionals and the corresponding action

If / Else if / Else	Condition	Action

Develop Your Card

Once your teacher has approved your design, go to Code Studio to program your card.

Check Your Card

Check your card to make sure it has everything it needs.

Reflect

What part of your project are you most proud of? _____

Why? _____

_____

If you had more time, what improvement would you make to your card?

_____

_____

Unit 3 Chapter 1 Project Rubric

Key Concept	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Program Development	Gave thoughtful feedback to peers and responded to peer feedback by making appropriate changes to program	Gave and responded to peer feedback.	Gave some feedback to peers.	Did not give feedback to peers
Modularity	Multiple sprites, with multiple properties updated in the draw loop	Multiple sprites, each with at least one property updated inside the draw loop	At least one sprite, with at least one property updated after sprite creation.	No sprites, or no sprite properties are updated after the sprite is created.
Algorithms and Control Structures	Program responds to multiple types of user input and uses at least one random number	Program responds to user input and uses at least one random number	Program uses a random number or responds to a user input.	Program does not use random numbers or respond to user input.
Algorithms and Control Structures	Program is well sequenced and properly separates code in and out of the draw loop.	Program correctly separates code in and out of the draw loop to create animation. May contain some incorrectly sequenced code.	Program is animated through the draw loop, but some code is improperly placed in or out of the loop.	Draw loop is not used to create animation
Algorithms and Control Structures	Uses multiple conditionals inside the draw loop, at least one of which is triggered by a variable or sprite property	Uses a conditional that is triggered by a variable or sprite property inside the draw loop.	Uses at least one conditional inside the draw loop.	No conditionals.
Position and Movement	Multiple elements are placed on the screen using the coordinate system, and move in different ways.	At least one element is placed on the screen using the coordinate system and moves during the program.	At least one element is placed on the screen using the coordinate system.	No elements (sprites or shapes) are placed on the screen using the coordinate system.
Variables	Multiple variables are used and their values are updated during the program. At least one variable or property uses the counter pattern.	At least one variable is used, and its value is updated during the program.	At least one variable is used in the program.	No variables.

Peer Review - Interactive Card

Pre-Review

Creator's Name: _____

One thing I want feedback on is… _____

_____

Reviewer Section

Reviewer's Name: _____

Evidence I Found	Types of Evidence	Ideas for More
	Multiple sprites, with multiple properties updated in the draw loop	
	Program responds to multiple types of user input and uses at least one random number	
	Program is well sequenced and properly separates code in and out of the draw loop.	
	Uses multiple conditionals inside the draw loop, at least one of which is triggered by a variable or sprite property	
	Multiple elements are placed on the screen using the coordinate system, and move in different ways.	
	Multiple variables are used and their values are updated during the program. At least one variable or property uses the counter pattern.	

Free Response Feedback

I like… _____

_____

I wish… _____

_____

What if… _____

_____

Creator's Reflection

1. What piece of feedback was most helpful to you? Why?

   _____

   _____

   _____

   _____

2. What piece of feedback surprised you the most? Why?

   _____

   _____

   _____

   _____

3. Based on feedback, what changes will you make to your interactive card?

   _____

   _____

   _____

   _____

Name(s)_____ Period _____ Date _____

Practices Reflection

How I've grown	Practice	How I want to grow
	Problem Solving	
	Persistence	
	Creativity	
	Collaboration	
	Communication	

Unit 3 Lesson 15

Velocity

Resources

Unit 3 Lesson 16

Collision Detection

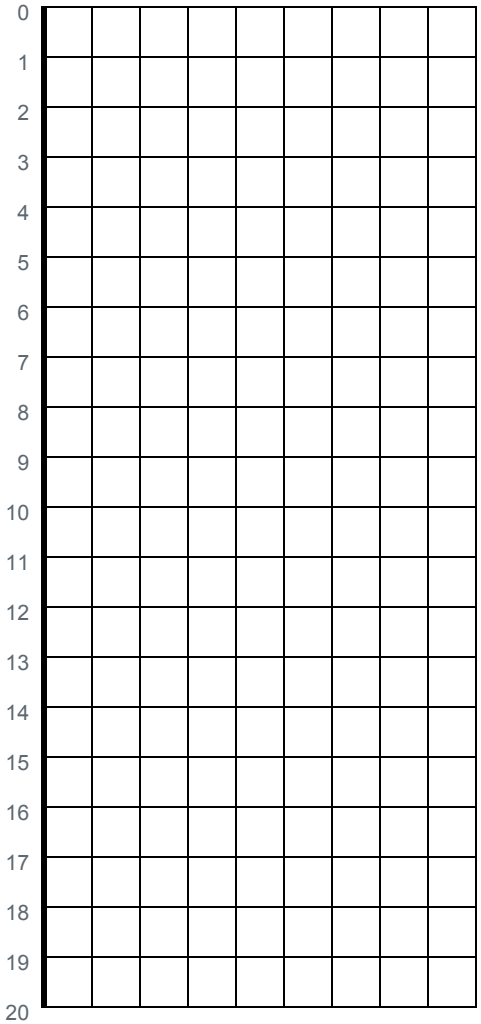
Resources

Name(s)_____ Period _____ Date _____

Activity Guide - Collisions B

Draw Your Sprites:

In the grid below, draw two square sprites on the left hand line. The sprites can be touching or not. Don't let your partner see your sprites.



Sprite Properties:

Using the grid, find the y position of the sprite (right in the middle of the square) and the sprite's height.

	y	height
sprite 1		
sprite 2		

Name(s)_____ Period _____ Date _____

Activity Guide - Collisions B

Collision Detection:

Copy the information from the first page onto this chart, then turn your picture over so your partner can't peek. The chart below is the only information your partner should see about your sprites. Once you've filled out the first chart, trade this worksheet with your partner so you can both try out your collision detectors.

	y	height
sprite1		
sprite2		

STOP!

Trade worksheets with your partner before continuing.

Look at the information on the chart above. Can you think of any way that you could use these numbers to figure out whether the two sprites are touching? Brainstorm your ideas below.

Do you think the sprites are touching?. Why or why not?

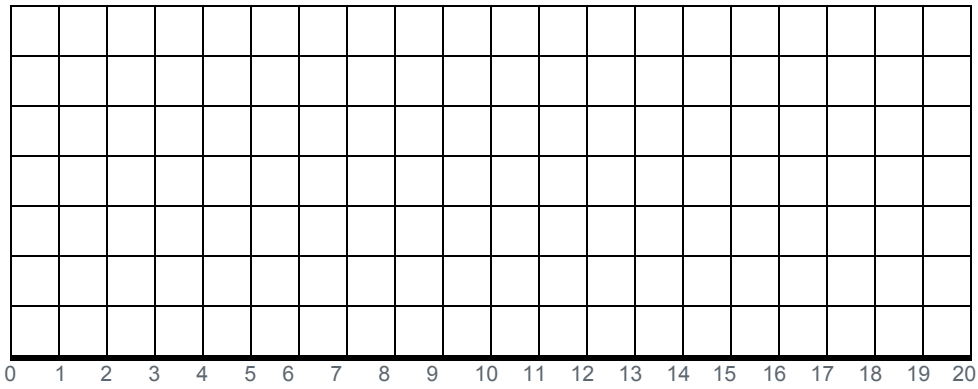
Check your partner's drawing to see whether you were correct. Do you think your strategy was effective?

Name(s)_____ Period _____ Date _____

Activity Guide - Collisions A

Draw Your Sprites:

In the grid below, draw two square sprites on the bottom line. The sprites can be touching or not. Don't let your partner see your sprites.



Sprite Properties:

Using the grid, find the x position of the sprite (right in the middle of the square) and the sprite's width. This is the only information you should tell your partner about the sprites.

	x	width
sprite 1		
sprite 2		

Name(s)_____ Period _____ Date _____

Activity Guide - Collisions A

Collision Detection:

Copy the information from the first page onto this chart, then turn your picture over so your partner can't peek. The chart below is the only information your partner should see about your sprites. Once you've filled out the first chart, trade this worksheet with your partner so you can both try out your collision detectors.

	x	width
sprite1		
sprite2		

STOP!

Trade worksheets with your partner before continuing.

Look at the information on the chart above. Can you think of any way that you could use these numbers to figure out whether the two sprites are touching? Brainstorm your ideas below.

Do you think the sprites are touching?. Why or why not?

Check your partner's drawing to see whether you were correct. Do you think your strategy was effective?

Unit 3 Lesson 17

Complex Sprite Movement

Resources

Unit 3 Lesson 18

Collisions

Resources

Unit 3 Lesson 19

Functions

Resources

Unit 3 Lesson 20

The Game Design Process

Resources

Name(s)_____ Period _____ Date _____

Project Guide - Defender Game

Overview

Building a larger piece of software like a game can quickly get complex. Starting with a plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin writing your actual code.

Gameplay and Visuals

Start by thinking about what your game actually does. What does it look like? How do you actually play it? What will make it fun, interesting, or relevant to the player?

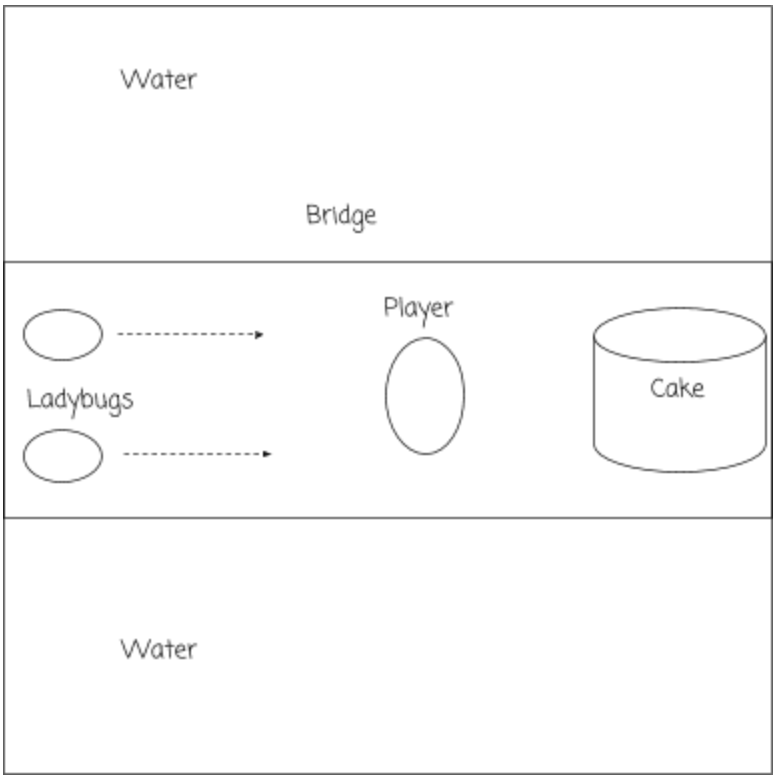
Describe Your Game

In a couple of sentences describe the game you are going to build and how it will work.

You are an alien defending your cake from evil ladybugs. Ladybugs march across a bridge towards your cake. You'll need to push the ladybugs off the bridge into the water. Get points for stopping ladybugs and lose points for letting ladybugs get through.

Draw Your Game

Draw a quick sketch of how your game will work. Who are the characters. What does the background look like? How do things move? Label things to make them more clear.



Sprites and Variables

Using the description of your game above, figure out what information and characters you'll need to keep track of through your game. Fill in a description for each in the space below.

Sprites

In the table below list information about the different sprites in your game. Where are they located? How do they move? How do they interact with other sprites?

Name (Label) and Appearance	**At Start of Game** (Animation, position, rotation, velocity, rotation speed)	**User and Sprite and Interactions** (Does the user control this sprite? How does it move? Does it ever need to reset its position? Does it interact with other sprites? How?)
player - alien	*x: 300* *y: 200* *No rotation or velocity*	*Arrow keys move the sprite in 4 directions* *Displaces ladybugs*
cake - cake	*x: 350* *y: 200* *No rotation or velocity*	*Never moves* *When ladybugs touch you lose points*
enemy1 - ladybug	*x: 0* *y: random between 150 and 250* *velocityX: 2 (may need to make faster)*	*Moves across the screen left to right* *Player sprite can displace the ladybugs* *If they hit the water reset and give the player a point.* *If they hit the cake reset, remove 5 points*
enemy2 - ladybug	*x: 0* *y: random between 150 and 250* *velocityX: 2 (may need to make faster)*	*Same as the other one*

Variables

Think about the information your game needs to keep track of. Is there a score? A number of lives? Describe each variable in the space below.

Name (Label)	**What It Keeps Track Of**	**How It Changes During the Game** (What's the starting value, when will it change?)
score	*Player's score*	*Starts at 0. When ladybugs pushed in water get 1 point. When ladybugs hit cake lose 5 points.*

Functions

Your draw loop shouldn't have a lot of complex code. Instead, break your program up into the major steps you'll need for your game to work. The different behaviors you described for your sprites and variables should help you decide what these steps should be. Then describe what the code for that function should do.

Function Name	**What Happens In This Function** What behaviors that you outlined for your sprites does this function include? Can this function be used at multiple places in your program?
gameBackground()	*Draws the background of the game*
enemiesTouchCake()	*Resets enemies when they touch the cake. Increases the player's score.*
movePlayer()	*Moves the player with arrow keys and changes animations*
displaceEnemies()	*Player pushes the ladybugs*
enemiesTouchWater()	*If either ladybug touches the water they get reset and the player's score is increased*
showScore()	*Shows the score on the screen*

Unit 3 Lesson 21

Using the Game Design Process

Resources

Name(s)_____ Period _____ Date _____

Project Guide - Planning Your Game

Overview

Building a larger piece of software like a game can quickly get complex. Starting with a plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin writing your actual code.

Gameplay and Visuals

Start by thinking about what your game actually does. What does it look like? How do you actually play it? What will make it fun, interesting, or relevant to the player?

Describe Your Game

In a couple of sentences describe the game you are going to build and how it will work.

_____

_____

_____

_____

Backgrounds

Draw a quick sketch of what you want the background(s) of your game to look like.

Sprites and Variables

Using the description of your game above, figure out what information and characters you'll need to keep track of through your game. Fill in a description for each in the space below.

Sprites

In the table below list information about the different sprites in your game. Where are they located? How do they move? How do they interact with other sprites?

Name (Label) and Appearance	**At Start of Game** (Animation, position, rotation, velocity, rotation speed)	**User and Sprite and Interactions** (Does the user control this sprite? How does it move? Does it ever need to reset its position? Does it interact with other sprites? How?)

Variables

Think about the information your game needs to keep track of. Is there a score? A number of lives? Describe each variable in the space below.

Name (Label)	**What It Keeps Track Of**	**How It Changes During the Game** (What's the starting value, when will it change?)

Functions

Your draw loop shouldn't have a lot of complex code. Instead, break your program up into the major steps you'll need for your game to work. The different behaviors you described for your sprites and variables should help you decide what these steps should be. Then describe what the code for that function should do.

Function Name	**What Happens In This Function** What behaviors that you outlined for your sprites does this function include? Can this function be used at multiple places in your program?

Unit 3 Lesson 22

Project - Design a Game

Resources

Name(s)_____ Period _____ Date _____

Project Guide - Planning Your Game

Overview

Building a larger piece of software like a game can quickly get complex. Starting with a plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin writing your actual code.

Gameplay and Visuals

Start by thinking about what your game actually does. What does it look like? How do you actually play it? What will make it fun, interesting, or relevant to the player?

Describe Your Game

In a couple of sentences describe the game you are going to build and how it will work.

_____

_____

_____

_____

Backgrounds

Draw a quick sketch of what you want the background(s) of your game to look like.

Sprites and Variables

Using the description of your game above, figure out what information and characters you'll need to keep track of through your game. Fill in a description for each in the space below.

Sprites

In the table below list information about the different sprites in your game. Where are they located? How do they move? How do they interact with other sprites?

Name (Label) and Appearance	**At Start of Game** (Animation, position, rotation, velocity, rotation speed)	**User and Sprite and Interactions** (Does the user control this sprite? How does it move? Does it ever need to reset its position? Does it interact with other sprites? How?)

Variables

Think about the information your game needs to keep track of. Is there a score? A number of lives? Describe each variable in the space below.

Name (Label)	**What It Keeps Track Of**	**How It Changes During the Game** (What's the starting value, when will it change?)

Functions

Your draw loop shouldn't have a lot of complex code. Instead, break your program up into the major steps you'll need for your game to work. The different behaviors you described for your sprites and variables should help you decide what these steps should be. Then describe what the code for that function should do.

Function Name	**What Happens In This Function** What behaviors that you outlined for your sprites does this function include? Can this function be used at multiple places in your program?

Program your Game

Once your teacher has approved your design, go to Code Studio to create your game.

Check Your Program

Check your program to make sure it has everything it needs.

Reflect

What part of your project are you most proud of? _____

Why? _____

_____

If you had more time, what improvement would you make to your game?

_____

_____

Unit 3 Chapter 2 Project Rubric

Key Concept	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Program Development	The project guide is complete and reflects the project as submitted.	The project guide is mostly complete and is generally reflective of the submitted project.	The project guide is filled out, but is not complete or does not reflect the submitted project.	The project guide is incomplete or missing.
Program Development	The program code effectively uses whitespace, good naming conventions, indentation and comments to make the code easily readable.	The program code makes use of whitespace, indentation, and comments.	The program code has few comments and does not consistently use formatting such as whitespace and indentation.	The program code does not contain comments and is difficult to read.
Modularity	At least three functions are used to organize code into logical segments. At least one of these functions is called multiple times in the program.	At least two functions are used in the program to organize code into logical segments.	At least one function is used in the program.	There are no functions in the program.
Algorithms and Control Structures	The game has at least three backgrounds that are displayed during run time, and at least one change is triggered automatically through a variable (e.g. score).	The game has multiple backgrounds that are displayed during run time (e.g. main background and “end game” screen)	The game has multiple backgrounds.	The game does not have multiple backgrounds.
Algorithms and Control Structures	The game includes multiple different interactions between sprites, responds to multiple types of user input (e.g. different arrow keys).	The game includes at least one type of sprite interaction and responds to user input.	The game responds to user input through a conditional.	The game includes no conditionals.
Position and Movement	Complex movement such as acceleration, moving in a curve, or jumping is included in multiple places in the program.	The program includes some complex movement, such as jumping, acceleration, or moving in a curve.	The program includes simple independent movement, such as a straight line or rotation.	There is no movement in the program, other than direct user control.
Variables	The game includes multiple variables that are updated during the game and affect how the game is played.	The game includes at least one variable that is updated during the game and affects the way the game is played	There is at least one variable used in the program.	There are no variables, or they are not updated.

Peer Review - Make Your Own Game

Pre-Review

Creator's Name: _____

One thing I want feedback on is…_____

_____

Reviewer Section

Reviewer's Name: _____

Evidence I Found	Types of Evidence	Ideas for More
	The program code uses spacing, good naming conventions, and comments to make the code easily readable.	
	At least three functions are used to organize code into logical pieces. At least one of these functions is called multiple times in the program.	
	The game has at least three backgrounds that are displayed while the game is running, and at least one change is triggered automatically through a variable (e.g. score).	
	The game includes multiple different interactions between sprites, responds to multiple types of user input (e.g. different arrow keys).	
	Complex movement such as acceleration, moving in a curve, or jumping is included in multiple places in the program.	
	The game includes multiple variables that are updated during the game and affect how the game is played.	

Free Response Feedback

I like… _____

_____

I wish… _____

_____

What if… _____

_____

Creator's Reflection

1. What piece of feedback was most helpful to you? Why?

   _____

   _____

   _____

   _____

2. What piece of feedback surprised you the most? Why?

   _____

   _____

   _____

   _____

3. Based on feedback, what changes will you make to your game proposal?

   _____

   _____

   _____

   _____

Name(s)_____ Period _____ Date _____

Practices Reflection

How I've grown	Practice	How I want to grow
	Problem Solving	
	Persistence	
	Creativity	
	Collaboration	
	Communication	