Copy this whole section to your OneNote Space & then fill it in

In this activity, students are introduced to:

- Opening a game, game controls, and playing a game
- Creating a new game
- The difference between blocks and script
- “My Projects”
- Sharing games by downloading, and creating a game link

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. What did the **say** block do when you added it to the game in task #2?
   
   Your response here

2. Compare and contrast the two different ways that we saved our game; what are the benefits of each choice? Which do you think you would use if you wanted to email a game you made to your friend, and why would you make that decision?
   
   Your response here

3. Include the image created in task #2.
   
   Your response here

4. Include the link created in task #4.
   
   Your response here
Activity: Math Operators with Variables

In this activity, students are introduced to:
- Expressing numeric operations with math operators (+, -, *, /)
- Storing the result of an equation in a variable
- Evaluating equations using variables
- Modifying and creating variable equations
- Displaying text withSplash by usingJoin
- Differentiating between the number and string data types

INSTRUCTIONS

Follow these instructions and answer the questions below

What did we learn?

1. In JavaScript what is the resulting value of answer = 5 + 3 * 2? Explain why the answer is not 16.

   Your response here

2. How is putting operators and numbers into a calculator different than writing in code (example: 2 + 3 + 4 * 4)? Explain.

   Your response here

3. Research and describe 2 other Math operators found in the math menu in Arcade (hover over values for more information).

   Your response here
Activity: Javascript Math Operators with Variables

In this activity, students are introduced to:

- the advantages of writing expressions out in JavaScript as opposed to blocks
- how the order in which an expression is evaluated can be changed
- how the different increment, decrement, and assignment operators can be useful for modifying a number variable.

INSTRUCTIONS

Follow these instructions and answer the questions below:

What did we learn?

1. What is an advantage of writing expressions out in JavaScript as opposed to blocks?
   
   Your response here

2. In JavaScript, how can the order in which an expression is evaluated be changed?
   
   Your response here

3. Explain how the different increment, decrement, and assignment operators can be useful for modifying a number variable.
   
   Your response here
Copy this whole section to your OneNote Space & then fill it in

Activity: Info Variables

In this activity, students are introduced to:

- Using the score and life properties
- Combining numeric values with math operators (*)
- The benefits of using score and life over other options
- The countdown block
- The pause block

INSTRUCTIONS

Follow these instructions and answer the questions below

What did we learn?

1. List one extra behavior you get for each of the three values we used in the info category (score, lives, and countdown).

   Your response here

2. List one potential downside of using score over just using your own variables to keep track of the state of your game.

   Your response here
Activity: Javascript Info Variables

In this activity, students are introduced to:

- Explaining the difference between declaring and assigning a value
- Explaining how the output is changed
- Identifying what type the variables need to hold

INSTRUCTIONS

Follow these instructions and answer the questions below:

What did we learn?

1. Explain the difference between declaring and assigning a value. It might be useful to try to come up with an analogy.

   Your response here

2. In task #3, the same variable was splash ed twice, with different results. Explain how the output was changed in the other lines of code.

   Your response here

3. Make a hypothesis on why using + between a string and a number results in a string, not a number.

   Your response here
4. In task #3, you had to look at code someone else wrote and figure out what types the variables needed to hold. What were the clues that helped you identify what type the variables needed to hold?

Your response here
Copy this whole section to your OneNote Space & then fill it in

**Activity: Sprites**

![Sprites W.A.L.T. Image](image1)

In this activity, students are introduced to:

- Using blocks
- Sprites and Images
- Using the Image Editor
- Pixels and pixel Colors
- Viewing JavaScript
- Color codes in the image editor, blocks and JavaScript

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. Make a table showing:
   - Color index (1-15)
   - Color (use an approximate color name like white, red, pink, ...)
   - Color representation in JavaScript

   ![Your response here](image2)

2. Explain what happens to the color index 0 in JavaScript (form a hypothesis).

   ![Your response here](image3)

3. Explain why we see only 14 colors at a time, despite the fact that there are 16 color indexes (0-15) in the image editor.

   ![Your response here](image4)

**Activity: Coordinate Walker**

![Coordinate Walker W.A.L.T. Image](image5)

In this activity, students will investigate the game screen.

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

Use X and/or Y in your answers

1. Describe how coordinates change when moving up and down. Which direction caused the coordinates to increase?
2. Describe how coordinates change when moving right and left. Which direction caused the coordinates to increase?

Your response here

**Activity: Collage**

- Designing their own images
- Use variables to create multiple sprites
- Using coordinates on the screen to arrange our sprites into a scene
- Changing a the background color

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. How could the use of multiple sprites in a single game enhance your ability to convey a story in your games?

2. What other blocks are available in the **Sprites** menu? List several blocks not used in collage and make a hypothesis of what one (or more) of these blocks does.

3. **Challenge:** switch to the JavaScript version of your collage, and take a look at the images you drew in the task above. What do periods (.) represent in the images?

**Activity: Hello Sprite**

In this activity, students will be introduced to:

- `console.log` (using JavaScript)

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. Discuss the different ways we can display a message in this exercise. Make a hypotheses for what each would be best used for.

2. Come up with a hypothesis and explain why `console.log()` has a larger limit on the length of text. Explain.

Your response here
Activity: Javascript Sprites

In this activity, students are introduced to:

- Explaining why drawing your own Sprites can be important when creating new games.
- Explaining how Sprite Properties allow you to interact with Sprites

**INSTRUCTIONS**

Follow these instructions and answer the questions below.

What did we learn?

1. In your own words, explain why drawing your own Sprites can be important when creating new games.

   [Blank]

2. How do Sprite Properties allow you to interact with Sprites?

   [Blank]
Activity: Sprite Motion and Events

In this activity, students are introduced to:
- Controller events
- Incrementing x and y coordinates
- Setting vx and vy velocity
- Short methods and functions with motion
- stay on screen
- Flipping (and switching) images

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. Describe how events can be used to run code using an example.

   Your response here

2. Describe the difference between changing position and changing velocity.

   Your response here

3. **Challenge:** when using flip in task #4, mySprite is not what is flipped. What is actually flipped? Explain how you know.
Activity: Javascript Sprite Motion and Events

In this activity, students are introduced to:
- Identifying how the player can move the Sprite in the game
- Identifying how sprite velocity is changed

INSTRUCTIONS

Follow these instructions and answer the questions below

What did we learn?

1. With an example, identify how a player can move the Sprite in the game

   Your response here

2. With an example, identify how sprite velocity is changed

   Your response here
Activity: Sprite Overlap & Events - Part 1

**W.A.L.T.**
*We Are Learning To...*

**W.I.L.F.**
*What I’m Looking For...*

In this activity, students are introduced to:
- Sprite Kind
- on overlap event with different kind
- ghost on and ghost off
- destroy sprite

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. Describe how a kind is used to detect overlap.

   Your response here

2. Suppose you have 2 or more sprites that have the same kind, and one of them triggers an overlap event. Explain how you can reference the sprite that was involved in the overlap event, rather than one of the other sprites of that kind.

   Your response here

Activity: Random Sprite Location

**W.A.L.T.**
*We Are Learning To...*

**W.I.L.F.**
*What I’m Looking For...*

In this activity, students will use:
- pick random to generate random numbers
- Setting random sprite positions
- Setting button press events
What did we learn?

1. Describe how the ability to generate a random value can make a game more interesting and/or challenging.

Your response here

2. Make a hypothesis of a good use of pick random that you would like to design into a future game - especially something we don’t know how to do yet. Be descriptive of the game and how a random value would be needed.

Your response here

Activity: Sprite Overlap & Events - Part 2

In this activity, the student will continue to work with:
- on overlap event with a kind applied to several identical sprites
- Overlap events

What did we learn?

1. Describe how a kind can improve code (for example, how it can make programming easier, more powerful, more efficient, ...).

Your response here

2. Explain why in creating a “bump” effect negative X and Y velocities are used to change the X and Y positions.

Your response here
Activity: Javascript Sprite Overlap & Events

In this activity, students are introduced to:

- Explaining why the event handler for the overlap event has two parameters.
- Explaining why is the Sprite Ghost flag is commonly used in overlap events

INSTRUCTIONS

Follow these instructions and answer the questions below.

What did we learn?

1. In your own words, explain why the event handler for the overlap event has two parameters.

   Your response here

2. Why is the Sprite Ghost flag commonly used in overlap events?

   Your response here

Activity: Generate Sprites using Create and On Create
In this activity, students are introduced to:
Describing how a kind label is used in generating a sprite
Explaining what the on created block does

INSTRUCTIONS

Follow these instructions and answer the questions below

What did we learn?

1. Describe how a kind label is used in generating a sprite by creating an empty sprite block.
   
   Your response here

2. Explain what the on created block does for you.
   
   Your response here
1. functional non-functional requirements

- defined functional requirements and decomposed Input, Processing, Output (IPO) chart
defined non-functional requirements and decomposed the user experience

Control Structures

- **on start**
  - start event
  - set background color to
  - set background colour

- set the game sprite

- move the sprite around

- **start countdown 10 (s)**
  - This will start a countdown

- create an **event** that occurs when a Player Sprite touches and Food Sprite.

- **change score by 1**
  - increase score

- **set mySprite position to x 0 y 0**
  - move to another position on the screen

- **pick random 0 to 10**
  - generate a random number within a certain range

Functional Requirements - Input, process, output (IPO) table

Work your way through the Chase the Pizza tutorial, filling in the table below as you go.

<table>
<thead>
<tr>
<th>Input (event/data)</th>
<th>Processing (coding)</th>
<th>Output (action)</th>
</tr>
</thead>
</table>
Non-functional requirements - UI and UX

Outline what the user interface (UI) and user experience (UX) will be like? Use the information below as a guide.

Your response here

Useability heuristics

Jakob Nielsen’s useability heuristics (Nielsen, 1994) can be used as a detailed checklist to evaluate the user interface of a digital solution:

☐ Visiblity of system status
   The system should always keep users informed about what is going on, through appropriate feedback and within a reasonable time frame.

☐ Match between system and the real world
   The system should speak the users’ language with words, phrases and concepts familiar to the user.

☐ User control and freedom
   Support undo and redo.

☐ Consistency and standards
   Users should not have to wonder whether different words, situations or actions mean the same thing.

☐ Error prevention
   Careful design prevents a problem from occurring in the first place.

☐ Recognition rather than recall
   Reduce memory load by making user choices visible.

☐ Flexibility and efficiency of use
   Provide shortcuts for expert users so the system can cater to both experienced and inexperienced users.

☐ Aesthetic and minimalist design
   Less is more. Keep the design as simple as possible. Do not display data that is rarely needed.

☐ Help users recognise, diagnose and recover from errors
   Error messages should be expressed in plain language, precisely indicate the problem and suggest a solution.

☐ Help and documentation
   Make the system easy to learn but provide help documentation that is easy to search and task-focused.

Source: Adapted with permission from Jakob Nielsen’s original 10 heuristics, available at https://www.nngroup.com/articles/ten-useability-heuristics/.

2. User Experience and Algorithm
design and evaluation of user experiences and algorithms

- sketched the user experience
- designed the algorithm in makecode
- evaluated the user experience to recommend changes
- evaluated algorithm to refine and make more efficient

Sketch or paste a picture of what the user experience will be like, along with a description of what will happen and what it will be like

Example

The player moves around using the direction pad. The pizza moves at random. When the player collides with the pizza, the score increases and the pizza moves to a new random position. If there is no collision within 3 seconds, the game is over.

Algorithm

Paste a screenshot from MakeCode

How would you refine your algorithm and make it more efficient?

Your response here
UX Recommendations

How would you modify the User Experience (UX)

Your response here

3. Testing algorithm

3. Testing algorithm

Refined Algorithm

How did you refine and improve your algorithm to make it work differently or more efficiently? Explain and show

Your response here

4. Implement with Javascript

4. Implement with Javascript

Make your refinements by editing the Javascript code
Copy the Javascript code below and use a // to comment each line of the code, explaining what it does.

Your response here

How did you test it?

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected response</th>
<th>Actual Response</th>
</tr>
</thead>
</table>

5. Evaluate

Evaluate & Refine

W.A.L.T
We Are Learning To...

evaluation of information systems and their solutions in terms of risk, sustainability and potential for innovation and enterprise

W.I.L.F
What I'm Looking For...

1. evaluated enterprise needs and opportunities
2. evaluated innovation
3. evaluated sustainability
4. made recommendations

Evaluate

Enterprise needs and opportunities
What needs or opportunities does the solution address? How well does it address these?

Innovative
How is the digital solution innovative? What potential impact does it have?

Sustainability
What is the lifespan of your solution? How long before better technology arrives and it is obsolete?
Is your solution easy to use and learn? Why/Why not?

Recommendations
Recommend at least one improvement that you would like to see made to the digital solution. Why?

Your response here
Create your own basic motion game

Now it's time to put everything that you have learned so far together and create your own game!

Concepts Learned

- Sprites
- Sprite Position (coordinates: x and y)
- Controller (dx) and move mySprite with buttons
- "Spawning" sprites: create and on create
- random
- Displaying numbers (1, 2, 3) as a string ("123")
- score and life
- countdown
- flip image
- stay in screen and ghost
- set image
- sprite say
- splash

From [https://arcade.makecode.com/courses/control/motion/project](https://arcade.makecode.com/courses/control/motion/project)

Use the Scamper Tool for ideas

With the "Chase the Pizza" or "Cherry Pickr" or "Lemon Leak" game as a basis, use Scamper to brainstorm a list of fresh ideas that develop and improve the original
1. functional non-functional requirements

Functional Requirements - Input, process, output (IPO) table

<table>
<thead>
<tr>
<th>Input (event/data)</th>
<th>Processing (coding)</th>
<th>Output (action)</th>
</tr>
</thead>
</table>

Non-functional requirements - UI and UX

Outline what the user interface (UI) and user experience (UX) will be like? Use the information below as a guide.

Your response here

Useability heuristics

Jakob Nielsen's usability heuristics (Nielsen, 1994) can be used as a detailed checklist to evaluate the user interface of a digital solution.
2. User Experience and Algorithm

- **W.A.L.T**
  - We Are Learning To...
  - design and evaluation of user experiences and algorithms

- **W.I.L.F**
  - What I'm Looking For...
  - - sketched the user experience
  - - designed the algorithm in makecode
  - - evaluated the user experience to recommend changes
  - - evaluated algorithm to refine and make more efficient

Sketch or paste a picture of what the user experience will be like, along with a description of what will happen and what it will be like

**Example**
The player moves around using the direction pad. The pizza moves at random. When the player collides with the pizza, the score increases and the pizza moves to a new random position. If there is no collision within 3 seconds, the game is over.

### Algorithm

Paste a screenshot from MakeCode

How would you refine your algorithm and make it more efficient?

Your response here

### UX Recommendations

How would you modify the User Experience (UX)

Your response here

### 3. Testing algorithm

**W.A.L.T**

*We Are Learning To*

testing and prediction of results and implementation of digital solutions

**W.I.L.F**

*What I'm Looking For*

- tested and refined make code algorithm, taking care to record iterations

### Refined Algorithm

How did you refine and improve your algorithm to make it work differently or more efficiently? Explain and show

Your response here

### 4. Implement with Javascript
design and implementation of modular programs, including an object-oriented program, using algorithms and data structures involving modular functions that reflect the relationships of real-world data and data entities

- implemented the solution in javascript

Make your refinements by editing the Javascript code

Copy the Javascript code below and use a // to comment each line of the code, explaining what it does.

Your response here

How did you test it?

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected response</th>
<th>Actual Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Gather feedback about the game from an outside source
1. Find someone to test the current game with
2. Give a 15-30 second description of what your game is supposed to be: describe the features that are not yet implemented to give them an idea of the ‘bigger picture’
3. Have them play your game
4. Ask for feedback:
   1. What do they like about the game?
   2. What would they like to be improved?
   3. How would they like it to be improved?

From <https://arcade.makecode.com/courses/csintro1/motion/project>

5. Evaluate

Evaluate

Enterprise needs and opportunities
What needs or opportunities does the solution address? How well does it address these?
**Innovative**
How is the digital solution innovative? What potential impact does it have?

**Sustainability**
What is the lifespan of your solution? How long before better technology arrives and it is obsolete?
Is your solution easy to use and learn? Why/Why not?

**Recommendations**
Recommend at least one improvement that you would like to see made to the digital solution. Why?

Convert the feedback into a list of features that to implement
  1. **Reflect** on the feedback you gathered from users testing your game
  2. **Identify** what features you would like to add based on what the users want
  3. **Prioritize** what features are most important to implement next

From [https://arcade.makecode.com/courses/csintro1/motion/project](https://arcade.makecode.com/courses/csintro1/motion/project)

Your response here
Copy this whole section to your OneNote Space & then fill it in

Activity: Loops Intro

In this activity, students are introduced to:
- Sprite motion with loops
- `repeat` loop

**INSTRUCTIONS**

Follow these instructions and answer the questions below

**What did we learn?**

1. Describe how a `repeat` block makes programming easier by reducing code repetition. Use an example.

   Your response here

2. Explain how it is easier (or harder) to add in a second sprite to the code inside of the loop than it would have been to add it in the prior (loop-less) version? Why?

   Your response here

3. Did you use more than one `repeat` in any of the tasks above? Why might you want to have one loop after another, rather than just combining them into a single loop?

Activity: Increment Loop

In this activity, students are introduced to:
- `repeat` loops
- `for index` loops
- Variables with `increase by`
- `on game update every`
- `set score`
- `countdown`

**INSTRUCTIONS**
What did we learn?

In task #3, you may have noticed that when you switched from a repeat loop to a for index loop, the sprite actually continued in its spiral for a little bit longer than it did before. Why is that?

Your response here

Is there a difference between how many times repeat 0 times and for index 0 to 0 will run? When might you want to choose to use a repeat loop over a for index loop?

Your response here

Activity: Projectile Sprites

In this activity, students are introduced to:
- projectile from side sprites
- on game update every
- pick random
- for loop

INSTRUCTIONS

Follow these instructions and answer the questions below

What did we learn?

1. Describe two benefits of using projectiles rather than normal sprites.

Your response here

2. How did using a loop in this section help reduce the amount of blocks that were used?

Your response here

Activity: Projectiles from Sprites

In this activity, students are introduced to:
- projectile from mySprite sprites
- on sprite destroyed
- pick random
- ghost on

INSTRUCTIONS

Follow these instructions and answer the questions below
What did we learn?
How can we make a sprite that came from a regular sprite act like a projectile sprite? Explain the block code to use.

Give examples of using ghost and/or on overlap events in a game to make a projectiles that are decorations, laser beams, and coins (reward). Explain each.

Challenge: create a hypothesis on why making projectiles have ghost on might make your game run faster than leaving it off.

Activity: Sprite Physics

INSTRUCTIONS
Follow these instructions and answer the questions below

What did we learn?
1. Why does making a sprite have a random velocity in both the x and y directions cause the sprite to move in a random direction? How would limiting the projectile to only positive directions change this?

2. In examples #1a and #1c, you likely noticed that the values were fluctuating more for less than for second, even though they were both increasing at the same rate on a second-by-second basis. Make a hypothesis on why that is.

3. Challenge: did either racer (example #1a or example #1c) have a clear winner? Make a hypothesis on why this might be, even if the rate of change is the same on a second-by-second basis.
Activity: Javascript Sprite Motion and Events

INSTRUCTIONS

Follow these instructions and answer the questions below.

What did we learn?

1. In your own words, describe the initialization, check, and update steps of a for loop.

2. In Arcade, create a new project and place a `repeat` and a `for index` loop in the `on start`. Convert the code to JavaScript: what is different between the two loops?

Activity: Projectiles

INSTRUCTIONS

Follow these instructions and answer the questions below.

What did we learn?

1. How are projectile `Sprites` different from `sprites` created with `sprites.create`?
2. Create a hypothesis on how `SpriteFlag.StayInScreen` will change the behavior of a `Sprite`. Test this hypothesis by creating a game that uses it.

Your response here

3. How do particle effects impact your game? When would particle effects not be useful?

Your response here
Copy this whole section to your OneNote Space & then fill it in

Create your own game with looping

Now it’s time to put everything that you have learned so far together and create your own game!

Concepts Learned

- Sprite Position (coordinates)
- Controller (dx) and move mySprite with buttons
- overlap Events / sprite kind
• “Spawning” sprites: create and on create
• random
• score and life
• countdown
• flip image
• stay in screen and ghost
• sprite say
• splash
• create / on create
• set background color
• projectile and projectile from sprite
• “Crash”, “bump” or other movements using repeat or for for a projectile

Use the Scamper Tool for ideas

With the "Level Up!" or a similar game ideas below, as a basis, use Scamper to brainstorm a list of fresh ideas that develop and improve the original

Think of at least 3 ideas for games that use some of the “concepts learned” listed above
Discuss your ideas with a partner. Talk about:

- What you like about the ideas
- What you don’t like about the ideas
- Any suggestions you have for extending the ideas

With your partner, discuss what features would be necessary to start each game

1. functional non-functional requirements
definition and decomposition of complex problems in terms of functional and non functional requirements

- defined functional requirements and decomposed Input, Processing, Output (IPO) chart
- defined non-functional requirements and decomposed the user experience

Functional Requirements - Input, process, output (IPO) table

Thinking about your game, fill in the table below.

<table>
<thead>
<tr>
<th>Input (event/data)</th>
<th>Processing (coding)</th>
<th>Output (action)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Non-functional requirements - UI and UX

Outline what the user interface (UI) and user experience (UX) will be like? Use the information below as a guide.

Useability heuristics

Jakob Nielsen's useability heuristics (Nielsen, 1994) can be used as a detailed checklist to evaluate the user interface of a digital solution.

- **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback and within a reasonable time frame.

- **Match between system and the real world**
  The system should speak the users' language with words, phrases and concepts familiar to the user.

- **User control and freedom**
  Support undo and redo.

- **Consistency and standards**
  Users should not have to wonder whether different words, situations or actions mean the same thing.

- **Error prevention**
  Careful design prevents a problem from occurring in the first place.

- **Recognition rather than recall**
  Reduce memory load by making user choices visible.

- **Flexibility and efficiency of use**
  Provide shortcuts for expert users so the system can cater to both experienced and inexperienced users.

- **Aesthetic and minimalist design**
  Less is more. Keep the design as simple as possible. Do not display data that is rarely needed.

- **Help users recognise, diagnose and recover from errors**
  Error messages should be expressed in plain language, precisely indicate the problem and suggest a solution.

- **Help and documentation**
  Make the system easy to learn but provide help documentation that is easy to search and task-focused.

Source: Adapted with permission from Jakob Nielsen's original 10 heuristics, available at https://www.nngroup.com/articles/ten-useability-heuristics/.
2. User Experience and Algorithm

- sketched the user experience
- designed the algorithm in makecode
- evaluated the user experience to recommend changes
- evaluated algorithm to refine and make more efficient

Sketch or paste a picture of what the user experience will be like, along with a description of what will happen and what it will be like

Example
The player moves around using the direction pad. The pizza moves at random. When the player collides with the pizza, the score increases and the pizza moves to a new random position. If there is no collision within 3 seconds, the game is over.

Algorithm

Paste a screenshot from MakeCode

How would you refine your algorithm and make it more efficient?

Your response here

UX Recommendations

How would you modify the User Experience (UX)?

Your response here

3. Testing algorithm
testing and prediction of results and implementation of digital solutions

- tested and refined make code algorithm, taking care to record iterations

**Refined Algorithm**

How did you refine and improve your algorithm to make it work differently or more efficiently? Explain and show

Your response here

4. Implement with Javascript

- implemented the solution in javascript

**Make your refinements by editing the Javascript code**

Copy the Javascript code below and use a // to comment each line of the code, explaining what it does.

Your response here

How did you test it?

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected response</th>
<th>Actual Response</th>
</tr>
</thead>
</table>

Gather feedback about the game from an outside source
1. Find someone to test the current game with
2. Give a 15-30 second description of what your game is supposed to be: describe the features that are not yet implemented to give them an idea of the ‘bigger picture’
3. Have them play your game
4. Ask for feedback:
   1. What do they like about the game?
   2. What would they like to be improved?
   3. How would they like it to be improved?

From <https://arcade.makecode.com/courses/csintro1/motion/project>

5. Evaluate

Evaluate & Refine

W.A.L.T
We Are Learning To...

evaluation of information systems and their solutions in terms of risk, sustainability and potential for innovation and enterprise

W.I.L.F
What I’m Looking For...

1. evaluated enterprise needs and opportunities
2. evaluated innovation
3. evaluated sustainability
4. made recommendations

Evaluate

Enterprise needs and opportunities
What needs or opportunities does the solution address? How well does it address these?

Innovative
How is the digital solution innovative? What potential impact does it have?

Sustainability
What is the lifespan of your solution? How long before better technology arrives and it is obsolete? Is your solution easy to use and learn? Why/Why not?

Recommendations
Recommend at least one improvement that you would like to see made to the digital solution. Why?
Convert the feedback into a list of features that to implement

1. **Reflect** on the feedback you gathered from users testing your game
2. **Identify** what features you would like to add based on what the users want
3. **Prioritize** what features are most important to implement next

From <https://arcade.makecode.com/courses/csiro1/motion/project>
Copy this whole section to your OneNote Space & then fill it in

Create your own game

Incorporate functionality learned throughout the course sections to build a well functioning game of your own design.

Requirements

- Use at least one concept from each section of the course - for example, from Motion, you could use:
move mySprite with buttons
on A button pressed
on overlap events

• For each of the above concepts, write a short sentence on how it allows your game to be more interesting or challenging for the player. Focus on the Design Decisions you made; what about the way you used the blocks made the game more appealing?

Use the Scamper Tool for ideas

With the game ideas below, as a basis, use Scamper to brainstorm a list of fresh ideas that develop and improve the original

Paparazzi
Asteroid Blaster
Baseball Catching
Dodgy Duck

Discuss your ideas with a partner. Talk about:
• What you like about the ideas
• What you don’t like about the ideas
• Any suggestions you have for extending the ideas
With your partner, discuss what features would be necessary to start each game and write a short sentence on how it allows your game to be more interesting or challenging for the player. Focus on the Design Decisions you made; what about the way you used the blocks made the game more appealing?

1. functional non-functional requirements

**W.A.L.T.**

*We Are Learning To...*

definition and decomposition of complex problems in terms of functional and non functional requirements

**W.I.L.F.**

*What I’m Looking For...

- defined functional requirements and decomposed Input, Processing, Output (IPO) chart
- defined non-functional requirements and decomposed the user experience

**Functional Requirements - Input, process, output (IPO) table**

Thinking about your game, fill in the table below.

<table>
<thead>
<tr>
<th>Input (event/data)</th>
<th>Processing (coding)</th>
<th>Output (action)</th>
</tr>
</thead>
</table>

**Non-functional requirements - UI and UX**

Outline what the user interface (UI) and user experience (UX) will be like? Use the information below as a guide.

Your response here

**Useability heuristics**

Jakob Nielsen’s useability heuristics (Nielsen, 1994) can be used as a detailed checklist to evaluate the user interface of a digital solution.

- **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback...
Useability heuristics

Jakob Nielsen’s useability heuristics (Nielsen, 1994) can be used as a detailed checklist to evaluate the user interface of a digital solution.

- **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback and within a reasonable time frame.

- **Match between system and the real world**
  The system should speak the users’ language with words, phrases and concepts familiar to the user.

- **User control and freedom**
  Support undo and redo.

- **Consistency and standards**
  Users should not have to wonder whether different words, situations or actions mean the same thing.

- **Error prevention**
  Careful design prevents a problem from occurring in the first place.

- **Recognition rather than recall**
  Reduce memory load by making user choices visible.

- **Flexibility and efficiency of use**
  Provide shortcuts for expert users so the system can cater to both experienced and inexperienced users.

- **Aesthetic and minimalist design**
  Less is more. Keep the design as simple as possible. Do not display data that is rarely needed.

- **Help users recognise, diagnose and recover from errors**
  Error messages should be expressed in plain language, precisely indicate the problem and suggest a solution.

- **Help and documentation**
  Make the system easy to learn but provide help documentation that is easy to search and task-focused.

*Source:* Adapted with permission from Jakob Nielsen’s original 10 heuristics, available at https://www.nngroup.com/articles/ten-usability-heuristics/.

2. User Experience and Algorithm

![DEVELOP](DEVELOP.png)

![EVALUATE & REFINE](EVALUATE & REFINE.png)

**W.A.L.T**

*We Are Learning To...*

design and evaluation of user experiences and algorithms

**W.I.L.F**

*What I’m Looking For...*
- **sketched** the user experience
- **designed** the algorithm in makecode
- **evaluated** the user experience to recommend changes
- **evaluated** algorithm to **refine** and make more efficient

Sketch or paste a picture of what the user experience will be like, along with a description of what will happen and what it will be like

**Your response here**

**Example**

![Game Screen](image)

The player moves around using the direction pad. The pizza moves at random. When the player collides with the pizza, the score increases and the pizza moves to a new random position. If there is no collision within 3 seconds, the game is over.

**Algorithm**

**Paste a screenshot from MakeCode**

How would you refine your algorithm and make it more efficient?

**Your response here**

**UX Recommendations**

How would you modify the User Experience (UX)

**Your response here**

3. Testing algorithm
testing and prediction of results and implementation of digital solutions

- tested and refined make code algorithm, taking care to record iterations

**Refined Algorithm**

How did you refine and improve your algorithm to make it work differently or more efficiently? Explain and show

Your response here

**4. Implement with Javascript**

design and implementation of modular programs, including an object-oriented program, using algorithms and data structures involving modular functions that reflect the relationships of real-world data and data entities
- implemented the solution in javascript

**Make your refinements by editing the Javascript code**

Copy the Javascript code below and use a // to comment each line of the code, explaining what it does.

```
Your response here
```

**How did you test it?**

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected response</th>
<th>Actual Response</th>
</tr>
</thead>
</table>

Gather feedback about the game from an outside source

1. Find someone to test the current game with
2. Give a 15-30 second description of what your game is supposed to be: describe the features that are not yet implemented to give them an idea of the ‘bigger picture’
3. Have them play your game
4. Ask for feedback:
   1. What do they like about the game?
   2. What would they like to be improved?
   3. How would they like it to be improved?

From [https://arcade.makecode.com/courses/csintro1/motion/project](https://arcade.makecode.com/courses/csintro1/motion/project)

5. Evaluate

**evaluation** of information systems and their solutions in terms of risk, sustainability and potential for innovation and enterprise

1. evaluated enterprise needs and opportunities
2. evaluated innovation
3. evaluated sustainability
4. made recommendations

Evaluate

Enterprise needs and opportunities
What needs or opportunities does the solution address? How well does it address these?

Innovative
How is the digital solution innovative? What potential impact does it have?

Sustainability
What is the lifespan of your solution? How long before better technology arrives and it is obsolete? Is your solution easy to use and learn? Why/Why not?

Recommendations
Recommend at least one improvement that you would like to see made to the digital solution. Why?

Convert the feedback into a list of features that to implement
1. Reflect on the feedback you gathered from users testing your game
2. Identify what features you would like to add based on what the users want
3. Prioritize what features are most important to implement next

From <https://arcade.makecode.com/courses/csintro1/motion/project>

Your response here
Online resources:

- Official [Online Guide](https://learn.adafruit.com/makecode-arcade-with-samd51-m4/what-is-makecode-arcade) for HackerBox #0041
- HackerBox #0041 [Box Video](https://learn.adafruit.com/makecode-arcade-with-samd51-m4/what-is-makecode-arcade)

Hacked Joy Controller
https://learn.adafruit.com/joy-controller-feather/overview

https://learn.adafruit.com/makecode-arcade-with-samd51-m4/feather-m4-express

Picade

https://learn.adafruit.com/makecode-arcade-with-raspberry-pi-zero
Getting Started with Arcade

**What did we learn?**

1. **What did the `say` block do when you added it to the game in task #2?**
   
   *The "say" block puts text into a sprite*

2. **Compare and contrast the two different ways that we saved our game; what are the benefits of each choice? Which do you think you would use if you wanted to email a game you made to your friend, and why would you make that decision?**
   
   *Saving the game is the best because I can save it to my computer and then import it. Sharing is good for sharing but I wouldn’t rely on it for a major project.*

3. **Include the image created in task #2.**

   ![Image](image.png)

4. **Include the link created in task #4.**

   [https://makecode.com/_4gfKaR8ipYkw](https://makecode.com/_4gfKaR8ipYkw)

**Variables and Maths**

**What did we learn?**

1. **In JavaScript what is the resulting value of answer for `answer = 5 + 3 * 2`? Explain why the answer is not 16.**

   *It is calculated using order of operation*

2. **How is putting operators and numbers into a calculator different than writing in code (example: `2 + 3 + 4 * 4`)? Explain.**

   *The above will be calculated in order that you put it into the calculator. Code will calculate by order of operation*

3. **Research and describe 2 other **Math** operators found in the math menu in Arcade (hover over values for more information).**
Javascript Math Operators with Variables

What did we learn?

1. What is an advantage of writing expressions out in JavaScript as opposed to blocks?

In Blocks, each step of an equation needed its own block. This can make formulas difficult to properly express, as it can be hard to identify (or change) the order in which they are evaluated.

In JavaScript, the same formulas can be easier to express, as the syntax (structure) is much closer to what is used when evaluating math by hand or with a scientific calculator.

2. In JavaScript, how can the order in which an expression is evaluated be changed?

JavaScript uses a PEMDAS structure to determine the order in which operations are evaluated. This standards for Parentheses, Exponents, Multiplication or Division, Addition or Subtraction.

To change from this, use brackets. Eg (10 - 6) / 2

3. Explain how the different increment, decrement, and assignment operators can be useful for modifying a number variable.

Using blocks, we must use num = num + 1. With javascript we can use a shortcut num++ or num--

Info Variables

What did we learn?
1. List one extra behavior you get for each of the three values we used in the \textit{info} category (score, lives, and countdown).

The score appears in the top right, the lives appears with a heart in the top left, countdown makes the screen melt when the game is over and game over message appears.

2. List one potential downside of using \texttt{score} over just using your own variables to keep track of the state of your game.

If you use score, then it will always be in the top right, while if you use your own variable, then you can make it appear anywhere in any format.

\textbf{Javascript Info Variables}

\textbf{What did we learn?}

1. Explain the difference between \texttt{declaring} and \texttt{assigning} a value. It might be useful to try to come up with an analogy.

When a variable is first created, it needs to be declared using the \texttt{let} command. Eg \texttt{let cat}. A value can also be assigned to the variable at the same time. Eg \texttt{let cat = "Meow";}; if you want to change the value of the variable, then you assign a new value. Eg \texttt{cat = "purr";} Therefore, you only use the \texttt{let} command once when declaring a variable.

2. In task \#3, the same variable was \texttt{splashed} twice, with different results. Explain how the output was changed in the other lines of code.

The word variable was assigned a different value before being splashed.

3. Make a hypothesis on why using + between a \texttt{string} and a \texttt{number} results in a \texttt{string}, not a \texttt{number}.

This must mean that when you add a string and a number value together, you always get a string type as the answer.

4. In task \#3, you had to look at code someone else wrote and figure out what types the variables needed to hold. What were the clues that helped you identify what type the variables needed to hold?

The clues came from the values being assigned. The only tricky one was \texttt{y = x + z} as we needed to realise that it was adding a number and a string and therefore was a string type.

\textbf{Sprites}

\textbf{What did we learn?}

1. Make a table showing
   - Color index (1-15)
   - Color (use an approximate color name like white, red, pink, ...)
   - Color representation in JavaScript
It turns into a dot, as it is a blank colour

1. Explain why we see only 14 colors at a time, despite the fact that there are 16 color indexes (0-15) in the image editor.

   The black colour blends in white the background, the blank is transparent

Coordinate Walker

What did we learn?

Use X and/or Y in your answers

1. Describe how coordinates change when moving up and down. Which direction caused the coordinates to increase?

   Your response here

2. Describe how coordinates change when moving right and left. Which direction caused the coordinates to increase?

   Your response here

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>0,130</td>
<td>170,130</td>
</tr>
</tbody>
</table>

Collage
What did we learn?

1. How could the use of multiple sprites in a single game enhance your ability to convey a story in your games?

Any game has a background and usually more than one character as part of the narrative of the game. These elements all need to be represented by sprites.

2. What other blocks are available in the **Sprites** menu? List several blocks not used in collage and make a hypothesis of what one (or more) of these blocks does.

3. Challenge: switch to the JavaScript version of your collage, and take a look at the images you drew in the task above. What do periods (.) represent in the images?

   The period represents transparency

Hello Sprite

What did we learn?

1. Discuss the different ways we can display a message in this exercise. Make a hypotheses for what each would be best used for.

   You can display messages using the sprite say block or console.log command. The say block is good for in-game messages, while console.log is better for debugging.

2. Come up with a hypothesis and explain why console.log() has a larger limit on the length of text. Explain.

   The say block is used when characters in the game want to say short sentences that won't take up the whole screen. Console log may need to display larger text strings or data output which would take up the screen and obscure the game.

Javascript Sprites

What did we learn?

1. In your own words, explain why drawing your own **Sprites** can be important when creating new games.

   This is related to personalising your own sprites so that they have the image you want and are the kind you need to manipulate them

2. How do **Sprite Properties** allow you to interact with **Sprites**?

   Sprites have properties and functions that keep track of the different values related to the sprite. For example, an important property is the x,y position of the sprite. To move it, you need to change these values.
Sprite Motion and Events

What did we learn?
1. Describe how events can be used to run code using an example.
   Your response here

2. Describe the difference between changing position and changing velocity.
   Your response here

3. Challenge: when using flip in task #4, mySprite is not what is flipped. What is actually flipped? Explain how you know.
   Your response here

Javascript Sprite Motion and Events

What did we learn?
1. With an example, identify how a player can move the Sprite in the game
   Your response here

2. With an example, identify how sprite velocity is changed
   Your response here

Sprite Overlap & Events - Part 1

What did we learn?
1. Describe how a kind is used to detect overlap.
   Your response here

2. Suppose you have 2 or more sprites that have the same kind, and one of them triggers an overlap event. Explain how you can reference the sprite that was involved in the overlap event, rather than one of the other sprites of that kind.
   Your response here

Random Sprite Location

What did we learn?
1. Describe how the ability to generate a random value can make a game more interesting and/or challenging.
   Your response here

2. Make a hypothesis of a good use of pick random that you would like to design into a future game - especially something we don’t know how to do yet. Be descriptive of the game and how a random value would be needed.
Sprite Overlap & Events - Part 2

What did we learn?

1. Describe how a **kind** can improve code (for example, how it can make programming easier, more powerful, more efficient, ...).

Your response here

2. Explain why in creating a “bump” effect negative X and Y velocities are used to change the X and Y positions.

Your response here

Javascript Sprite Overlap & Events

What did we learn?

1. In your own words, explain why the **event handler** for the **overlap event** has two parameters.

Your response here

2. Why is the **Sprite Ghost** flag commonly used in **overlap** events?

Your response here

Generate Sprites using Create and On Create

What did we learn?

1. Describe how a **kind** label is used in generating a sprite by creating an empty sprite block.

Your response here

2. Explain what the **on created** block does for you.

Your response here