Order of Classes

- Traps, Lava, Platforms, Level, SplitCheese, BigCheese

Class BigCheese

Let splitArr = [] //array of split cheeses, gets populated when Split is called
Let isAlive = true
Let jumpCooldown
Let canJump
Let isSplit = false initially
Let xPos
Let yPos
Let xVelo
Let yVelo
Const gravity
Const friction
Let isMoving = false initially

<u>Event Listeners</u>

If W is pressed, call Move(up)
If A is pressed, call Move(Left)
If D is pressed, call Move(right)
If S is pressed, call Split()

Function Move(direction) // (0,0) is in top left

Check ifAlive = true

- isMoving = true;
- If W pressed, decrease y-coordinate (Event listener)
    - Set canJump to false
    - yVelo becomes negative and is set to specific value, every second add yVelo to yPos of cheese
    - yVelo increases due to the acceleration of gravitation until the cheese lands or collides
- If A pressed, decrease x-coordinate (Event listener)
    - xVelo becomes negative and is set to specific value, every second add xVelo to xPos of cheese
- If D pressed, increase x-coordinate (Event listener)
    - xVelo becomes positive and is set to specific value, every second add xVelo to xPos of cheese
- Call checkForCollisions()
- Call checkForLand()

Function landOnPlatform()

yVelo = 0;

If xVelo is negative, friction increase xVelo until xVelo = 0

If xVelo is positive, friction decrease xVelo until xVelo = 0

isMoving = false

start jumpCooldown

After cooldown canJump = true

Function Split

- If S split, check if cooldown is true.
    - If True, end method call
- If false:
    - Play splitting sound
    - Switch cheese assets to three small cheeses, each with initial velocities:
        - Make three splitCheese objects
            - Left, Right, Middle
            - Add them to splitArr
    - BigCheese gets hidden but not deleted, add Hidden modifier (below)
    - splitArr will still exist and you will still use BigCheese to access it.
    - BigCheese will remain hidden until summonCheese is called, it will then reappear

Function checkForCollision()

- While isMoving = true, loop through lava, traps to check for collisions (Check if xPos & yPos = xPos & yPos of lava or trap)
    - If true, call Die
    - If not, check if isMoving = true and if so, recall CheckForCollision (recursive)

Function checkForLand()

- While isMoving = true, loop through platforms to check for collisions (Check if platform.isInRange(xPos,yPos)
    - If true, call Land()
    - If not, check if isMoving = true and if so, recall checkForLand (recursive)

Function Die

- Let isAlive = false;
- Probably cue some sort of graphics once micah's figures out his s$*%

Class SplitCheese

Let xPos

Let yPos

Let xVelo

Let yVelo
Let splitCooldown = false
- Set three second timeout

Let isAlive = true

Function summonCheese(which splitCheese)
- Check If isAlive = true
- If == 1(left), trigger button is 1, then put BigCheese at position of leftCheese
- If == 2(middle), trigger button is 2, then put BigCheese at position of rightCheese
- If == 3(right), trigger button is 3, then put BigCheese at position of middleCheese
- Play sound
- Check if there are other smaller cheeses and get rid of them

Class LeftCheese extends splitCheese

CreateEvent listener to check if number key 1 is pressed
- Check isAlive=true
- If pressed, call summonCheese(1)

Class RightCheese

CreateEvent listener to check if number key 2 is pressed
- Check isAlive=true
- If pressed, call summonCheese(2)

Class MiddleCheese extends splitCheese

Create Event listener to check if number key 3 is pressed
- Check isAlive=true
- If pressed, call summonCheese(3)

Function moveMiddle
- Create event listener if W is pressed

Modifiers
- Hidden
  - Turn off BigCheese controls
  - Make invisible
  - This makes things easier as you will not have to create a new instance each time and splitCheese data gets saved in the BigCheese class

Class Platform // moving platforms

Let xPos
Let yPos
Let width
Let height

Function inLandingRange (xPos, yPos)
{
Checks if the cheese will be able to land based on its final position
Returns true or false
}

Function collision

- Constantly check:
- If cheeseHitbox.intersects(platformObject)
- Stop cheese y-acceleration
- Add inputs/outputs

Class Traps

Let xPos
Let xPos
Let width
Let height
Let launchTime
Const xVelo
Const yVelo

Function inCollidingRange (xPos, yPos)
{
Checks if the cheese will be able to land based on its current position
}

Class Lava

Let xPosMin
Let xPosMax
Let yPosMin
Let yPosMax
Let launchTime
Const xVelo
Const yVelo

```
        Function inCollidingRange (xPos, yPos)
        {
                Checks if the cheese will collide based on its current position
        }

Class Level
        Let dimensions;
        Let platforms[];
        Let traps[];
        Let lava[];

        Constructor
        Function addPlatform(platform)
                Fill platform array
                Returns void
        Function populateTraps(trap)
                Fill trap array
                Return void
        Function populateLava(lava)
                Fill lava array
                Return void
```