Basic Game #1 GDD-Lite: Darts

Today, we'll be making a game where the "player" is a horizontal dart that's moving up and down on its own. Clicking releases the dart which moves at a set speed toward a dartboard. Hitting the board with the dart gives you a point and plays a small fanfare.

Breaking it down

While making a coherent game (even a small one) seems daunting, they're usually just comprised of four constituent parts: Objects; Mechanics; Physics; and Goals.

Objects:

Each individual physical entity on-screen at any point. These require references to one another so that when they interact, the game calls a mechanic. They also usually require collision shapes, which are how the game knows that objects are interacting with one another. But some elements, like UI elements, are designed not to have a collision shape so that objects just pass beneath them harmlessly.

- Dart
- Dartboard
- Score UI (starting at 0)
- Wall

Mechanics:

Each individual interaction between objects whether it happens instantaneously, after a timer finishes, or via a physics calculation.

- Clicking throws the dart
    - Throwing the dart prevents you from throwing an additional dart
    - After the dart hits the wall or the dartboard, it can be fired again
- Hitting the dartboard awards you a point
    - Getting a point plays a happy noise and makes the Score UI increase by 1

Physics:

Each calculation that must be made to determine the speed, direction, and/or gravity of an object. These are typically calculated ahead of time and are called when a mechanic occurs, like clicking to throw the dart. However, some physics are also calculated only *after/when* a mechanic occurs (like a Pong ball getting faster over time)

- Dart moving up and down
- Dart moving horizontally when thrown
- Dart stopping when it hits the board or the wall

Goals:

The interaction that results in either a win condition or at least a step towards it. This is also the result of a mechanic occurring, but instead of causing something else to happen, it's usually the natural conclusion of a string of mechanical interactions when the player performs them correctly.

- Hit the dartboard with the thrown dart
    - Receive a point

Most games also include a loss state to contrast the goal, but we'll get to that in another module!

Now for the Coding

Now it's time to go through each object, mechanic, physics, and goal to detail the general code structure you might use in programming the game yourself! But this isn't your normal tutorial; we'll only be supplying code snippets that are relevant to get you started!

Objects:

- Dart – First, you'll need to make a 2D BODY that can move on command. Then you'll have to:
    - Attach a sprite to it;
    - Create a hitbox for covering the sprite
- Dartboard – You'll need to make a STATIC 2D BODY and:
    - Attach a sprite to it;
    - Attach a hitbox covering the sprite
- Score UI (starting at 0) – You'll just need to make a MESSAGE for this and place it in a corner
- Wall – Repeat the process from the Dartboard

Mechanics:

- Clicking throws the Dart – You'll need three main functions here:
    - Bind a button, key, or click to call the PHYSICS PROCESS function detailed below;
    - Add an IF function to stipulate that if the Dart has been thrown and is *still moving*, clicking does nothing
    - After the dart hits the wall or the dartboard, it can be fired again
- Hitting the dartboard awards you a point and makes the dart stop
    - We won't actually do anything here for now!

Physics:

- Dart moving up and down
    - You'll need to add a PHYSICS PROCESS function that occurs automatically when the game starts
- Dart moving horizontally when thrown
    - You'll need to add a PHYSICS PROCESS function to the Dart that moves it along the X axis at a constant speed;
    - You'll also need to ensure that there's a COLLISION variable in the function
- Dart stopping when it hits the board or the wall
    - Your Dart's PHYSICS PROCESS script will need to constantly check to see whether or not the Dart has collided with another object (either the Wall or the Dartboard) using the PHYSICS PROCESS, stopping when the dart *does* collide with something

Goals:

- Hit the dartboard with the thrown dart
    - You already have all the tools set up! You just need an IF function for when the Dart collides with the Dartboard! That will make your Message counter go up by 1

Now you've made your game!