Introduction:
Now you’re familiar with the DIY Gamer and editing code in an Arduino sketch, it’s time to write one from scratch. In this session you will write code that talks to the Light Dependent Resistor getting it to control the playback of an animation on the LED Display. This could be a pre-made animation, the one you created last session, or you can make a new one from scratch.

Goals
• Monitor light values from the Light Dependent Resistor (LDR).
• Use the changing light values to control an animation.
Keep track of your progress by ticking off the boxes below:

**Activity Checklist**

1. Open up the Arduino Software
2. Open a new sketch
3. Import the DIY Gamer library

This includes a library that allows us to talk to the gamer. It should already be installed so we can import it from the sketch menu. To import the library go to sketch > import library > Gamer  It will look then like this at the top of the sketch.

```cpp
#include <Gamer.h>
```

4. Create a New gamer object

This creates a new object that contains all of the DIY Gamer library functions, and we are going to call it, gamer. Type this below your imported gamer library.

```cpp
Sketch_X

#include <Gamer.h>

Gamer gamer;
```
Lesson 5: LDR Functions

Step 1: Write your essential scripts

Activity Checklist

5. Type set up script

This is the setup script that happens once at the beginning. Inside the curly bracket is where we setup any functions that we are going to use.

```cpp
#include <Gamer.h>
Gamer gamer;

void setup()
{
}
```

6. Type the void loop function

This is the start of our basic loop code which will be happen infinitely or until the DIY Gamer runs out of power. Type the following after your setup code.

```cpp
void setup()
{
}

void loop()
{
}
```
Step 1: Write your essential script

Activity Checklist

7. Edit setup script between curley brackets

This tells the gamer object to start doing all of its button sensor handling functions, and opens up a connection with the serial port with a speed of 9600.

```cpp
void setup()
{
    gamer.begin();
    Serial.begin(9600);
}
```

Test your project

8. Test your code

Click to verify that everything that has been typed so far is understood. It should say done compiling in the bottom menu bar if successful. Errors? Check for:

1. Missing capital letters
2. Missed or wrong brackets
3. Missed semi colons.

Ok, so that is the basic structure in place, we can start to pad it out to make it do the interesting stuff.
Step 1: Write your essential script

9. Monitor LDR output

Edit the void loop function.

```cpp
void loop()
{
    Serial.println(gamer.ldrValue());
    delay(10);
}
```

This outputs the light reading from the LDR to the serial monitor every 10 milliseconds, i.e. how much light is hitting its surface and then converting that to a number.

Test your project

10. Test your code

Click verify code to check that everything typed so far is understood.

Activity Checklist

11. Go to Menu > Tools > Board > Arduino Uno

This checks that we have the right Arduino board selected

12. Go to Menu > Tools > Serial Port > USB

Select the port that the Arduino is connected to. This name should be something like `/dev/tty.usbmodem` on a mac, or COM3 or higher on a PC.
Step 1: Write your essential script

Test your project

13. Test your code

Click ![arrow] to transfer the code onto the Arduino in the DIY Gamer.

The red LED should be blinking to show a transfer is taking place. The screen will be blank, but the DIY Gamer is working.

Activity Checklist

14. Click on magnifying glass in top right corner to open serial monitor

The serial monitor is a window that lets the Arduino communicate with us and tell us information about what is going on in the code.

We asked the Arduino to tell us the LDR value (how much light the LDR is detecting). This should now be shown as a stream of numbers in the serial monitor. Put your hand over the LDR, the numbers go up in value. Take your hand off, the numbers go down in value.

15. Write down highest and lowest values of LDR

Cover the LDR with your thumb. Write down the lowest reading here. Uncover the LDR. Write down the highest reading here.

These are important, we are going to use these numbers to control our animation. The numbers we are using in this example are 200 and 860, which is a room lit with lightbulbs.
Step 1: Write your essential script

Activity Checklist

16. Type the following above the setup script

```cpp
int currentFrame;
void setup()
{
```

This is a variable that stores the current frame of which our animation is going to play.

Save your project

Click the arrow facing downwards that looks like this to save your code.

Call your project YourNameLDR. It will save to the Arduino folder.
Keep track of your progress by ticking off the boxes below:

**Activity Checklist**

1. **Open > Gamer > AnimationWithLDR**
   
   Click up arrow icon to open sketch from Arduino folder

2. **Can you find the Void setup?**

   This set up all the functions the gamer needs.

```c
void setup(){
    //Set up Gamer
    gamer.begin();
}
```
Keep track of your progress by ticking off the boxes below:

**Activity Checklist**

3. **Can you find the Void loop?**

This is the main part of the code that is looped through over and over again.

```cpp
void loop(){
  //Convert LDR value to a frame.
  currentFrame = map(gamer.ldrValue(), 0, 1023, 0, NUFRAMES);

  //Print current frame!
  gamer.printImages(frames[currentFrame]);
  delay(10);
}
```

4. **Can you find the Arrays?**

These 8 rows of 8 one’s and zero’s are the part of the code which tell the Gamer’s LED matrix which LEDs to turn on or off. They directly represent the DIY Gamers screen.

5. **Remember how the 1’s represent a pixel that is on?**
6. Select the arrays by clicking and dragging with your mouse

Select ALL the arrays down from and including: # define NUMFRAMES

```c
#define NUMFRAMES 6
byte frames[NUMFRAMES][8] = {
    {B00000000,
    B00000000,
    B00000000,
    B00011000,
    B00011000,
    B00000000,
    B00000000,
    B00000000},

    {B00000000,
    B00000000,
    B00100100,
    B00011000,
    B00000000,
    B00000000,
    B00000000,
    B00000000},

    /* Add more frames as needed */
};
```

ONLY SELECT THE ARRAYS, NOT THE VOID SETUP CODE.

```c
void setup(){
    //Set up Gamer
    gamer.begin();
}
```

7. Copy the selected arrays

Go to edit > copy. Or use the shortcut cmd + c (mac osx) or ctrl + c (Windows)

8. Open > yourNameLDR sketch

Click up arrow icon   to open sketch from Arduino folder.

This will open the LDR animation sketch you were just working on.
11. Edit void loop

Type the following code at the start of the void loop.
This is where you need to use the LDR values that you wrote down earlier. The values you need to change to your own are circled below in blue.

```cpp
void loop()
{
    currentFrame = map(gamer.ldrValue(), 200, 860, 0, NUMFRAMES);
}
```

This line of code maps (using clever maths) the Light reading to the number of frames in your animations. Fully uncovered, the LDR will display the first frame of the animation. Fully covered up, it will show the last frame.
Activity Checklist

12. Edit void loop (continued)

Look at the example below to give you an idea of how it works. Remember to use the light readings from the space you are in!

<table>
<thead>
<tr>
<th>Light on LDR</th>
<th>LDR light reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>330</td>
<td>1</td>
</tr>
<tr>
<td>460</td>
<td>2</td>
</tr>
<tr>
<td>590</td>
<td>3</td>
</tr>
<tr>
<td>720</td>
<td>4</td>
</tr>
<tr>
<td>850</td>
<td>5</td>
</tr>
</tbody>
</table>

### DIY Gamer screen

13. Type in your void loop script on next line

```cpp
void setup(){
    //Set up Gamer
    gamer.begin();
    Serial.begin(9600);
}

void loop(){
    //Convert LDR value to a frame.
    currentFrame = map(gamer.ldrValue(), 0, 1023, 0, NUFRAMES);

    //Print current frame!
    gamer.printImage(frames[currentFrame]);
    Serial.println(gamer.ldrValue());
    delay(10);
}
```
Lesson 5: LDR Functions

Step 2: Control the screen using the LDR

Test your project

14. Test your code
Click verify code   to check that everything typed so far is understood and has compiled successfully.

15. Upload code to Gamer
Click    to transfer the code onto the Arduino in the DIY Gamer. The red LED should be blinking to show a transfer is taking place. The screen will be blank, but the DIY Gamer is working.

16. Test on DIY Gamer
Put your thumb over the LDR on the DIY Gamer and see what happens. The animation should now be on its first frame if the LDR is detecting light, and progress through the animations frames to the final frame when it is fully covered and in the dark.

Errors, check your light readings. Perhaps try entering ones that are 50 higher and lower than your readings to provide a safety margin.

Challenge:
How could you make the same animation play backwards, so it is at the end of the animation when receiving light?

Well Done
You have just coded the DIY Gamer to use its LDR to control an animation. Awesome! You are now well on your way to coding your first game.
This is an annotated transcript of the Light Dependant Resistor animation sketch to help you understand the functions of the different lines of code. When you put // before text in code, it means it is to be ignored.

```
#include <Gamer.h>

Gamer gamer;

int currentFrame;

for (int i = 0; i < NUMFRAMES; i++) {
    frames[i] = map(gamer.ldrValue(), 40, 800, 0, NUMFRAMES);
}

void setup() {
    gamer.begin();
    Serial.begin(9600);
}

void loop() {

    currentFrame = map(gamer.ldrValue(), 40, 800, 0, NUMFRAMES);
    gamer.printImage(frames[currentFrame]);
    Serial.println(gamer.ldrValue());
    delay(10);

} 
```