Topic	SOLAR ENERGY UTILIZATION IN AUTOMATED GREENHOUSE	
Class Description	**The student will understand the application of solar energy in agriculture for remote areas while understanding how to use technology to automate the farming process via simulation.**	
Class	**Bonus Class post C21**	
Class time	**55 mins**	
Goal	• Learn about the climate issues faced in Ladakh. • Understand the role of solar panels with help of greenhouse simulation. • Use solar power to maintain the temperature in the greenhouse.	

Resources Required	• Teacher Resources:
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	  ○ Visual Studio Code
	• Student Resources:
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	  ○ Visual Studio Code	

Class structure	**Warm-Up** **Teacher-led Activity 1** **Student-led Activity 1** **Wrap-Up**	**10 mins** **20 mins** **20 mins** **05 mins**

WARM-UP SESSION - 10 mins	
Teacher starts slideshow  **from slides 1 to 15** Refer to speaker notes and follow the instructions on each slide.	
Teacher Action	**Student Action**
The following are the warm-up session deliverables: • **Greet the student.** • **Introduction to Sonam Wangchuk and his initiatives.** Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? ***NOTE: Please present the VA from slide 1 to 15. It is necessary to understand the context of the class activity. It should take only 8-10 min. Stick to the content mentioned in the VA for timely completion of the class.***	**ESR**: Hi, thanks, Yes I am excited about it!
Teacher ends slideshow 	
TEACHER-LED ACTIVITY - 20 mins	
Teacher Initiates Screen Share	
ACTIVITY • **Explain the solar greenhouse.** • **Explain the activity.**	
Teacher Action	**Student Action**

Let us begin to create a simulation to charge and switch on the fans using solar heat when the temperature inside the greenhouse rises beyond 30 deg.	
*The teacher downloads Teacher Activity 1, unzips the folder, and saves it as "**Solar_greenhouse**".* *Open the **sketch.js** file.* ***NOTE: The code present in the boilerplate is for declaring variables, preload() function, and the setup() function which is known to the student and is provided, hence kindly explain that section in 5-7 minutes so that there is enough time for the Student Activity**.* Various variables are defined for loading images, creating sprites and groups. We can also see the variables created for temperature as **temp = 10.** Initially, the voltage generated by the solar panels is kept at **0**. There is one more variable, **power_gen**; where we can store the total voltage generated. Images for a greenhouse, background, fan, and solar panels are loaded. As you can see, the images are loaded inside the **preload()** function using **loadImage**. Whereas, **loadAnimation** is used for creating a moving fan animation.	
Variables:	

```
//Global variables for images
var bg, sun, s_pan, fan_anim,fan_img,display, g_house_img;

//Global variables for Sprites
var g_house, pan1,pan2,fan,fan2;

//Creating a ray group
var rayGroup;

//Creating temprature and voltage variables
var temp = 10
var panel1_voltage =0;
var panel2_voltage = 0;
var power_gen = 0;
```

The **preload()** function.

```
function preload()
{
  sunR = loadImage("sunrays.png");
  sunL = loadImage("sunrays1.png");
  bg = loadImage("bgimage.png")
  s_pan = loadImage("s_panel.png");
  fan_img = loadImage("fan01.png");
  display = loadImage("disp.png");
  g_house_img = loadImage("greenhouse.png")
```

Remember, we had created moving animation for Trex while creating a Trex game. Can you help me add animation for the moving fan? We can make use of 5 different images given to us to create an animation using **loadAnimation().** *The teacher writes the code to create animation for fans.*	**ESR:** Yes

```
function preload()
{
  sunR = loadImage("sunrays.png");
  sunL = loadImage("sunrays1.png");
  bg = loadImage("bgimage.png")
  s_pan = loadImage("s_panel.png");
  fan_img = loadImage("fan01.png");
  display = loadImage("disp.png");
  g_house_img = loadImage("greenhouse.png")

  fan_anim = loadAnimation("fan01.png","fan02.png","fan03.png","fan04.png","fan05.png");

}
```

In the **setup()** function you can see we have already created various sprites for **pan1, pan2, fan, fan2, g_house**, and their respective images for each sprite are added.

```
function setup()
{
  createCanvas(800, 500);

  g_house = createSprite(380,300,100,100);
  g_house.addImage(g_house_img);
  g_house.scale = 0.75;
  g_house.debug = true;
  g_house.setCollider("circle",-10,0,185)

  pan1 = createSprite(100,height-50,80,80);
  pan1.addImage(s_pan);
  pan1.scale = 0.75;

  pan2 = createSprite(width-150,height-50,80,80);
  pan2.addImage(s_pan);
  pan2.scale = 0.75;


  fan = createSprite(280,300,20,20);
  fan.addImage(fan_img);
  fan.scale = 0.3;
  fan.addAnimation('run',fan_anim);

  fan2 = createSprite(450,300,20,20);
  fan2.addImage(fan_img);
  fan2.scale = 0.3;
  fan2.addAnimation('run',fan_anim);
  textSize(15);
```

Now let us move to function **draw().**

```
function draw()
{
  background(220);


  power_gen = panel1_voltage + panel2_voltage

  push();
  noStroke();
  fill(255,255,0)
  text("Voltage : ",620,37)
  text(power_gen,680,37)

  text("Temprature : ",620,56)
  text(temp,710,56);
  pop();
```

Can you explain to me this part of the code?

ESR:

1. We are adding both the voltage and assign to **power_gen.**
2. We're printing text for the temperature and power voltage.

Yes, and we are also using **push()** and **pop()** we learned in last class to give special styling to this text.

Now let us make the output more interesting by adding some images.

We will add images using function **image().**

Till now, we have created sprites to show images.

The .**image()** function allows us to add images directly on the canvas without creating a sprite. It also allows us to put text or any other object on this image. It takes the following parameters: 1. Variable name where the image is loaded; 2. The x & y position for the image to be displayed 3. Width and height of an image We are using the **.image()** function to give a background image and display board, on which we will add details about temperature and voltage. *The teacher will write the **image()** function for background and for showing temperature.* *The teacher will also call the **makeRay()** function as shown below:*	

```
function draw()
{
  background(220);

  image(bg,0,0,width,height);
  image(display,600,10,200,60)


  power_gen = panel1_voltage + panel2_voltage

  push();
  noStroke();
  fill(255,255,0)
  text("Voltage : ",620,37)
  text(power_gen,680,37)

  text("Temprature : ",620,56)
  text(temp,710,56);
  pop();

  makeRay();
  drawSprites();
}
```

Next, we are calling the **makeRay()** function.

This function is similar to our **spawnclouds()** and **spwanobstacles()** functions:

1. Create 2 sprites **raysL** & **raysR** at random **x** positions, after every 60 **frameCount.**
2. Add image to both the sprites.
3. Add both sprites to the same group, **raysGroup().**
4. **raysGroup** is created using **createGroup()** in function **setup().**
5. Set velocity x to random from **(-1,1)**.
6. Assign velocity x and y to all the sprites in the group.
7. Assign **lifetime** to each sprite in the group.

Why do we give a lifetime? Correct, let's run the code once and see what we are getting.	**ESR**: To avoid memory leakage.

Inside function **setup().**

```
raysGroup = createGroup()
```

After function **draw().**

```
function makeRay()
{
  if (frameCount % 60 === 0)
  {
   var x = Math.round(random(10,350));
   rayL = createSprite(x,50,10,10);
   var xr = Math.round(random(350,750));
   rayR = createSprite(xr,50,10,10);
   rayL.addImage(sunL);
   rayR.addImage(sunR);
   rayL.scale = 0.08;
   rayR.scale = 0.08;
   vx = random(-1,1);
   raysGroup.add(rayL);
   raysGroup.add(rayR);
   raysGroup.setVelocityYEach(2)
   raysGroup.setVelocityXEach(vx)
   raysGroup.setLifetimeEach(134)
  }
```

OUTPUT

As you can see, we have all assets placed correctly, and our sun rays are falling as yellow lines at random positions with random speed.

What should we do next?

ESR: We need to find when the sun's rays collide with greenhouse or panels.

Yes, next we will check:

1. If the sun rays collide with the greenhouse, we will increase the temperature and remove those particular sun rays from the group.
2. Similarly, if the sun rays collide with solar panels, we will increase the voltage and remove those particular sun rays from the group.
3. When the temperature of the greenhouse rises

above 30 deg, and the voltage is generated up to 4 volts. We will power up one fan. 4. Similarly, when the voltage reaches up to 8 volts, we can power both the fans. 5. Once the fans start, we will reduce the temperature by half a deg to show the effect of fans.	
Are the steps clear for you? Great, so start sharing your screen, and we will continue to write the code.	**ESR:** Yes
Teacher Stops Screen Share	
STUDENT-LED ACTIVITY - 20 mins	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Fullscreen.**	
ACTIVITY • **Detect the sun rays touching panels and greenhouse** • **Write a condition to start fans using power generated by Solar panels.**	
Teacher starts slideshow :Slide 16 & Slide 18	
Teacher Action	**Student Action**
The teacher guides the student to download the code from STUDENT ACTIVITY 1.	*The student downloads the code from STUDENT ACTIVITY 1. Unzip the folder and save it as "**Solar_greenhouse**"* *Open the folder in VSC.*
Let us start first by checking when the group of sun rays (the yellow images falling randomly) **(raysGroup)** touch	

the panel **(pan1 or pan2)** or greenhouse **(g_house)** and increase the voltage or temperature accordingly. Here, we will use the **overlap()** function. The advantage of using the **overlap()** function over **isTouching()** or **collide()** function is that if the target is a group of sprites, the function will traverse to check for each sprite for overlapping, and helps to take action on that particular sprite. If you remember, in the Trex game when we used **isTouching()** function to detect when Trex is colliding with an obstacle group, we were not taking further action on the obstacle group. Here, once we detect collision between rays and panels or greenhouse, we want to remove that particular sprite from the game. The **overlap()** function allows us to do that. Let us open the link to understand the **overlap()** function a bit more. Syntax: ***sprite.overlap**(otherSprite, newfunction);* ***function** newfunction(spriteA, spriteB) {* * **sprite**A.remove();* * **sprite**B.score++;* *}* We can pass the name of the sprites between which we want to check the collision, at the same time we can call different functions to take action based on collision. *The teacher guides the student to write the code to check an overlap between **SpriteGroup** and **greenhouse** and **panels**, and remove the sprite from the group.*	*The student opens Student Activity 2.*

We will write these functions inside the **makeRay()** function. We need to use the **overlap()** function thrice, for the greenhouse and two panels. **raysGroup.overlap(pan1,charge1);** The statement checks if the **raysGroup** is overlapping/touching **pan1,** then moves to the **charge1()** function**.** It also passes the particular sprite details which is touching **pan1** Similarly, **raysGroup.overlap(pan2,charge2);** The statement checks if the **raysGroup** is overlapping / touching **pan2,** then moves to **charge2()** function**.** It also passes the particular sprite details which is touching **pan2.** **raysGroup.overlap(g_house,temp_rise);** The statement checks if the **raysGroup** is overlapping / touching the **g_house** sprite image**,** then moves to the **temp_rise()** function**.** It also passes the particular sprite details which is touching **g_house.**	

```
function makeRay()
{

  if (frameCount % 60 === 0)
   {
    var x = Math.round(random(10,350));
    rayL = createSprite(x,50,10,10);
    var xr = Math.round(random(350,750));
    rayR = createSprite(xr,50,10,10);
    rayL.addImage(sunL);
    rayR.addImage(sunR);
    rayL.scale = 0.08;
    rayR.scale = 0.08;
    vx = random(-1,1);
    raysGroup.add(rayL);
    raysGroup.add(rayR);
    raysGroup.setVelocityYEach(2)
    raysGroup.setVelocityXEach(vx)
    raysGroup.setLifetimeEach(134)
   }
  raysGroup.overlap(pan1,charge1);
  raysGroup.overlap(pan2,charge2);
  raysGroup.overlap(g_house,temp_rise)
}
```

What should happen in the **charge1() & charge2()** function?

Yes, let us create a function **charge1()**, **charge2()** and **temp_rise()** after the **makeRay()** function.
These functions are created to take actions as per the overlap detected.

In order to increase the value of voltage, we will keep counting how many **raysGroup** sprites are overlapping with the **pan1** & **pan2.**

ESR: Voltage should increase, and we need to remove the sprite.

To keep a count of that, create a variable **absorbed1 & absorbed2**. We will increase it by one every time it overlaps with the solar panels.

Create Global Variable (before function **preload()**):

```
var absorbed1= 0;
var absorbed2= 0;
```

If the overlap is detected between raysGroup() and pan1 or pan2.
Increment the value of absorbed by one and remove the particular sprite from the group using the **.remove()** function.

If the overlap is detected between raysGroup() and g_house.
Increment the value of **temp** by one and remove the particular sprite from the group using the **.remove()** function.

```
function charge1(sprA)
{
    sprA.remove()
    absorbed1+=1;
}
function charge2(sprA)
{
    sprA.remove()
    absorbed2+=1;
}
function temp_rise(sprb)
{
  sprb.remove();
  temp+=1;
}
```

sprA.remove() and **sprb.remove()** will remove the rayL/rayR from the **raysGroup** after the collision.

The teacher can ask the student to check the output to see the temperature increasing. What should we do next? Yes, but before that, we will also convert the **absorbed** into voltage. If we keep increasing absorbed by one and assign it to voltage, it will increase very fast. Let us multiply it by **0.15** and assign it to **panel1_voltage** & **panel2_voltage**.	**ESR:** We need to start the fan once the temperature increases up to **30**.

Inside function **draw().**
 	
 
`text("Temprature : ",620,56)`
`text(temp,710,56);`
`pop();`
`//TA`
`makeRay();`
 	
`//calculate wattege`
`panel1_voltage = round(absorbed1* 0.15);`
`panel2_voltage = round(absorbed2* 0.15);`	

Now, let us add the condition to start the fan. Which conditions should be checked to start the fan? Yes, we need to check for temperature, but we also need to check if we have enough power generated to start the fan.	**ESR:** Temp should be greater than 30.

We will start one fan if we have **power_gen >= 4**, and we can start both the fans if the **power_gen >= 8**.

So our condition to start the first fan will be **temp** should be more than **30** and **power_gen > =4.**

And the second fan will start when **temp >= 30** and **power_gen >= 4.**

Inside function **draw():**

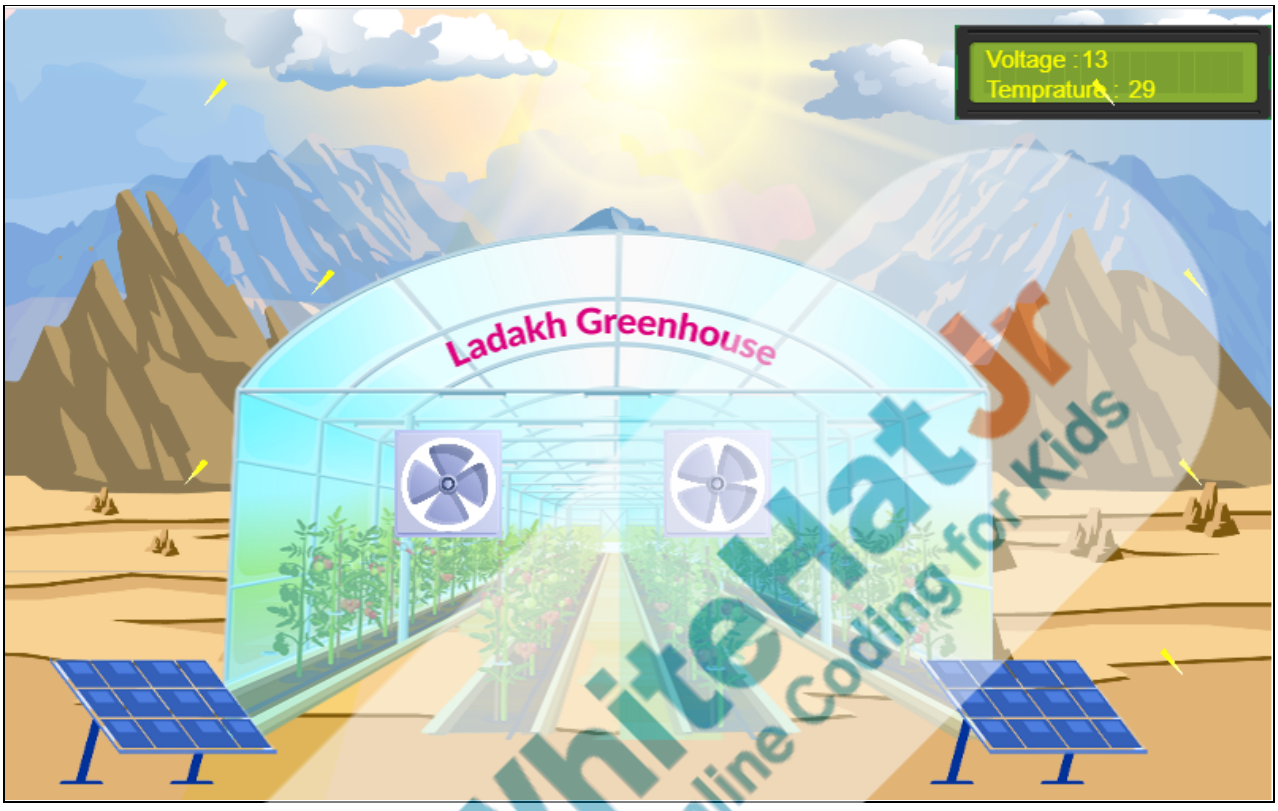
```
  panel1_voltage = round(absorbed1* 0.15);

  panel2_voltage = round(absorbed2* 0.15);


if(power_gen>=8 && temp>=30)
{
  fan.changeAnimation('run');
  temp-=1;
  panel2_voltage-=1
}

if(power_gen>=4 && temp>=30)
{
  fan2.changeAnimation('run');
  temp-= 0.5;
  panel1_voltage-=1
}
```

When the conditions are met,

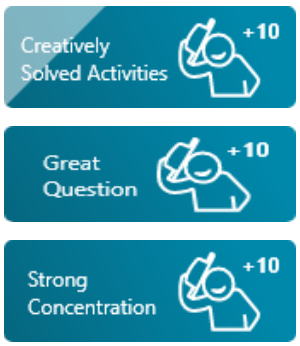
1. Change animation of the fan to moving.
2. Reduce **temp** by **1**.
3. Reduce panel voltage by **1**.

Let us run the code and check the output.

The teacher can check the output at Teacher Activity 3.



We can see the fans moving and maintaining temperature up to 30 deg. We will stop here for today. You can further research into renewable energy sources and how we can improvise global warming using them.	**ESR:** Ok
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 5 mins	
Teacher starts slideshow  **from slide 19 to slide 21**	
Activity details	**Solution/Guidelines**

Run the presentation from slide 19 to slide 21. **The following are the wrap-up session deliverables:** • **Explain the facts and trivias.**	
End the quiz panel	
FEEDBACK • **Appreciate student's efforts in the class.** • **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**	
Teacher Action	**Student Action**
You get "hats-off" for your excellent work! In the next class, we will create a new game using Physics Engine concepts.	*Make sure you have given at least 2 hats-off during the class for:* 
Teacher ends slideshow 	
Teacher Clicks 	

ADDITIONAL ACTIVITIES	
Encourage the student to write reflection notes in their reflection journal using Markdown. Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the Markdown editor to write her/his reflections in the reflection journal.*

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Boilerplate Code	https://github.com/procodingclass/Pro_Solar_Greenhouse_TeacherActivity1
Teacher Activity 2	Reference Code	https://github.com/procodingclass/PRO_SolarGreenhouse_ReferenceCode
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/Pro_Solar_Greenhouse_StudentActivity1
Student Activity 2	overlap() Reference	https://molleindustria.github.io/p5.play/docs/classes/Sprite.html#method-overlap
Visual-Aid	Visual-Aid	https://curriculum.whitehatjr.com/Visual+Project+Asset/PR

		O_VD/slides-pro-sonam-wangchuk-withspeakernotes.html
Teacher Activity 3	Output Video.	https://drive.google.com/file/d/13LRTuVFWzXILyRyNoJraYzBU1pUzT0oz/view?usp=sharing