# Part 1 – declaration etc. 1 mark

**Windmill**

Declare the constants and variables etc.

**/1 Mark**

# Part 2 – window.onLoad 1 mark

Setup the order of operations

**/1 Mark**

init();

setupCameraAndLight();

createGeometry();

createWindmillMaterials();

setupDatGui();

render();

# Part 3 – Initialization 2 marks

1. Configures the renderer and adds it to the dom

**/1 Mark**

**/1 Mark**

1. Initialize an OrbitControls object

# Part 4 – Setup the Camera and Lights 10 marks

The setupCameraAndLight**()** function will do the following:

**/1 Mark**

**/1 Marks**

**/3 Marks**

**/2 Marks**

**/3 Marks**

1. Sets the position and lookAt properties of the camera
2. Create and add an AmbientLight to the scene
3. Create and add a DirectionalLight to the scene which must be able to cast shadows
4. Create and add a HemisphereLight to the scene color
5. Create and add a SpotLight to the scene color which must be able to cast shadows

# Part 5 – Add the required geometry to the scene 2 marks

The createGeometries() function will do the following:

**/1 Mark**

**/1 Mark**

1. AxisHelper – Create and add an AxisHelper to your scene length 100
2. Plane – Create and add a plane to your scene that is able to receive shadows

# Part 6 – Add the materials 2 marks

The createWindmillMaterials() function will do the following:

**/1 Mark**

**/1 Mark**

1. Create a MeshLambertMaterial for the arms
2. Create a MeshStandardMaterial for the tower and axle

# Part 7 – Setup the necessary gui controls 12 marks

The initGuiControls() function will do the following:

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

**/1 Mark**

1. “Scene rotation” – checkbox. Only stops the scene rotation
2. “enableSpotlight” – function
3. “Arm color” – color
4. “Arm visibility” – bool
5. “Body color” – color
6. “Body visibility” – bool
7. “Rotation speed” – numeric slider range 0 to 0.04. Sets the speed for both the arms and scene
8. “Windmill” – folder
9. “Arm length” – numeric slider. (min=12, max=18 and step=2). Default =16.
10. “Number of Arms” – numeric slider. (min=2, max=5 and step=1) . Default =3.
11. “Tower height” – numeric slider. (min=18, max=27 and step=3) . Default =22.
12. “Create” – function calls the createWindmill() with the above values to obtain   
    an object and adds it to the scene

# Part 8 – Add the object to the scene 18 marks

The createWindmill() function will take armLength. bodyLength, numberOfArms and the armInclination = 0.25 and do the following:

**/½ Mark**

**/3 Marks**

**/½ Mark**

**/2 Marks**

**/9 Marks**

**/3 Marks**

1. Object3D – To serve as the container for the component of this object
2. CylinderGeometry – that is scaled appropriately to be the body of the hovercraft.   
   Uses the towerMaterial that was created in the Part 6
3. Object3D – To serve as the container for the arms and axle. You will have position it  
   properly and then add it to the container in step 1
4. BoxGeometry – use 4 for the width, 0.2 for the thickness and the length is specified  
   by the argument.   
   Uses the armMaterial that was created in Part 6  
   This must be repeated as many times as specified by the argument
5. CylinderGeometry –to be the axle of the object.   
   Uses the bodyMaterial that was created in the Part 6
6. Returns the completed 3-D object.

# Part 9 – Render the scene 3 marks

The render() function will do the following:

**/½ Mark**

**Mark**

**/1 Mark**

**/1.5 Marks**

1. Update the orbitController object
2. Rotate the arm based on the value in the dat.gui controller
3. Rotate the scene or stop based on the dat.gui controller

Name:

Date: