Immersify Part Viewer

Overview

Immersify part view is a lightweight mobile app capable of loading and displaying complex 3D representations machines and parts.

Controls

* Click and drag to rotate the model
* Click the model to expand into its pieces
* Click again to compress back to original state

Scripts

ExpandOrderHandler

public class ExpandOrderHandler : MonoBehaviour

{

public List<GameObject> Group1 = new List<GameObject>(); //split the parts into groups to allow for the expansion to happen in stages

public List<GameObject> Group2 = new List<GameObject>();

public List<GameObject> Group3 = new List<GameObject>();

public List<GameObject> Group4 = new List<GameObject>();

public bool Expandable = false;

int layer = 0;

float timer = 0;

void Update()

{

timer += Time.deltaTime;

if (timer > 0.5) { layer++; timer = 0; } //makes the expansion happen in 0.5 second increments

if (layer == 1) { for (int i = 0; i < Group1.Count; i++) { Group1[i].GetComponent<Part>().Transition(0.25f); } } //moves first layer out 25% of total distance

if (layer == 2)

{

for (int i = 0; i < Group1.Count; i++) { Group1[i].GetComponent<Part>().Transition(0.5f); } //moves first and second layers out 50% of total distance

for (int i = 0; i < Group2.Count; i++) { Group2[i].GetComponent<Part>().Transition(0.5f); }

}

if (layer == 3)

{

for (int i = 0; i < Group1.Count; i++) { Group1[i].GetComponent<Part>().Transition(0.75f); } //moves first, second and third layers out 75% of total distance

for (int i = 0; i < Group2.Count; i++) { Group2[i].GetComponent<Part>().Transition(0.75f); }

for (int i = 0; i < Group3.Count; i++) { Group3[i].GetComponent<Part>().Transition(0.75f); }

}

if (layer == 4)

{

for (int i = 0; i < Group1.Count; i++) { Group1[i].GetComponent<Part>().Transition(1.0f); } //moves all layers fully out

for (int i = 0; i < Group2.Count; i++) { Group2[i].GetComponent<Part>().Transition(1.0f); }

for (int i = 0; i < Group3.Count; i++) { Group3[i].GetComponent<Part>().Transition(1.0f); }

for (int i = 0; i < Group4.Count; i++) { Group4[i].GetComponent<Part>().Transition(1.0f); }

}

if(!Expandable) { reset(); }

}

void reset() //back to original positions

{

for (int i = 0; i < Group1.Count; i++) { Group1[i].GetComponent<Part>().Transition(0f); }

for (int i = 0; i < Group2.Count; i++) { Group2[i].GetComponent<Part>().Transition(0f); }

for (int i = 0; i < Group3.Count; i++) { Group3[i].GetComponent<Part>().Transition(0f); }

for (int i = 0; i < Group4.Count; i++) { Group4[i].GetComponent<Part>().Transition(0f); }

layer = 0;

}

}

This handles the expansion process and ensures all the individual parts move the correct amount at the correct time.

public class Part : MonoBehaviour

{

public Vector3 OriginalPos;

public Transform ExpandedPos;

float targetExpansion = 0f;

float currentExpansion = 0f;

public string Name;

void Start()

{

OriginalPos = this.transform.localPosition; //saves the original position on start

}

void Update()

{

if(currentExpansion != targetExpansion)

{

currentExpansion += 0.025f;

this.transform.localPosition = (OriginalPos + (ExpandedPos.transform.localPosition \* currentExpansion));

}

if(currentExpansion > targetExpansion) { currentExpansion = targetExpansion; }

if(targetExpansion == 0f) { this.transform.localPosition = OriginalPos; currentExpansion = 0f; }

}

public void Transition(float amount)

{

targetExpansion = amount;

}

}

This Stores and controls the transform and name values of each part.

public class InputHandler : MonoBehaviour

{

float RotationSpeed = 15;

public ExpandOrderHandler ExpandHandler;

bool TimerOn = false;

float timer = 0;

void OnMouseDrag()

{

float xRotation = Input.GetAxis("Mouse X") \* RotationSpeed \* Mathf.Deg2Rad;

float yRotation = Input.GetAxis("Mouse Y") \* RotationSpeed \* Mathf.Deg2Rad;

this.GetComponent<Transform>().RotateAround(Vector3.up, -xRotation);

this.GetComponent<Transform>().RotateAround(Vector3.right, yRotation);

}

void OnMouseUp() { TimerOn = false; timer = 0; }

void OnMouseDown() { TimerOn = true; }

void OnMouseUpAsButton()

{

if ( timer<0.2)

{

ExpandHandler.Expandable = !ExpandHandler.Expandable;

}

}

void Update() { if (TimerOn) { timer += Time.deltaTime; } }

}

This detects mouse input for clicks to expand the model and drag to rotate the model. Some workarounds were required to prevent the click at the beginning of a click and drag event from being interpreted as just a click.

public class NameHandler : MonoBehaviour

{

string RawInput;

string[] Names;

public ExpandOrderHandler Expander;

string Tag;

GameObject[] Parts;

void Start()

{

RawInput = System.IO.File.ReadAllText(Application.dataPath + "/CSV/PartNames.csv");

Names = RawInput.Split(',');

Parts = GameObject.FindGameObjectsWithTag("P1");

Name(Names[0]);

Parts = GameObject.FindGameObjectsWithTag("P2");

Name(Names[1]);

Parts = GameObject.FindGameObjectsWithTag("P3");

Name(Names[2]);

//ETC up to P12

}

void Name(string name)

{

for(int i = 0; i < Parts.Length; i++)

{

Parts[i].GetComponent<Part>().Name = name;

}

}

}

This read the list of names from the CSV file and matches them to parts with the appropriate tags, the names can be editted in the CSV without any changes to the code needed. At the moment it only sets the “name” value that is attached to each part to the correct name as I ran out of time to add in floating text over each part.