using UnityEngine;

using UnityEngine.UI;

public class MagnetismV2 : MonoBehaviour

{

 public float viseurSize = 100f;

 public float maxSpeed = 200f;

 public float \_attraction = 1500f;

 public float \_MagnetRange = 30f;

 public float \_maxSpeedRush = 100f;

 public float coefRecup = 50f;

 public float BubbleSlow = 0.5f;

 private Rigidbody rBody;

 public GameObject \_CloseM;

 private GameObject \_potentialTarget;

 public Image \_viseur;

 public Image \_Barres;

 public Image \_Dahs1;

 public Image \_Dahs2;

 public Image \_Dahs3;

 public Image \_Reticule;

 private Camera \_camera;

 private float Tdist;

 public float \_aGravity;

 public float \_aGDelay;

 private float \_cDelay;

 public Transform \_LeftHand;

 public Transform \_RightHand;

 public Transform SpTarget;

 public Transform SoundMagnet;

 public Transform SoundOrbit;

 public Vector3 \_LineRush;

 public Image WaterBlue;

 private float scoreReduc;

 private Vector3 Axis;

 public Transform \_center;

 public Transform \_oPoint;

 private bool \_orbiting;

 private float vitesseIn;

 private float ScreenRatio;

 private float \_timeNoHit;

 private void Start()

 {

 this.\_camera = Camera.main;

 this.rBody = this.GetComponent<Rigidbody>();

 this.\_CloseM = (GameObject) null;

 this.\_CloseM = (GameObject) null;

 this.Tdist = this.\_MagnetRange;

 this.\_potentialTarget = (GameObject) null;

 this.SpTarget.gameObject.SetActive(false);

 this.SoundMagnet.gameObject.SetActive(false);

 this.SoundOrbit.gameObject.SetActive(false);

 this.ScreenRatio = (float) Screen.width / (float) Screen.height;

 this.\_viseur.rectTransform.sizeDelta = new Vector2((float) ((double) this.viseurSize / (double) this.ScreenRatio \* (double) Screen.width / 1000.0), (float) ((double) this.viseurSize / (double) this.ScreenRatio \* (double) Screen.width / 1000.0));

 this.\_Reticule.rectTransform.sizeDelta = this.\_viseur.rectTransform.sizeDelta;

 this.\_Barres.rectTransform.sizeDelta = this.\_viseur.rectTransform.sizeDelta;

 this.\_Dahs1.rectTransform.sizeDelta = this.\_viseur.rectTransform.sizeDelta;

 this.\_Dahs2.rectTransform.sizeDelta = this.\_viseur.rectTransform.sizeDelta;

 this.\_Dahs3.rectTransform.sizeDelta = this.\_viseur.rectTransform.sizeDelta;

 this.\_orbiting = false;

 this.scoreReduc = 0.0f;

 this.\_timeNoHit = 0.0f;

 }

 private void Update()

 {

 this.CheckCloseMagnets();

 if ((double) this.\_timeNoHit <= 5.0 && (double) this.\_timeNoHit + (double) Time.deltaTime > 5.0)

 ScoreManager.\_score.PNotBad();

 this.\_timeNoHit += Time.deltaTime;

 if ((Object) this.\_potentialTarget != (Object) null)

 {

 this.SpTarget.position = this.\_potentialTarget.transform.position;

 this.SpTarget.gameObject.SetActive(true);

 Vector2 viewportPoint = (Vector2) this.\_camera.WorldToViewportPoint(this.\_potentialTarget.transform.position);

 this.\_Reticule.rectTransform.localPosition = new Vector3((float) ((double) viewportPoint.x \* (double) Screen.width - (double) Screen.width \* 0.5), (float) ((double) viewportPoint.y \* (double) Screen.height - (double) Screen.height \* 0.5), 0.0f);

 this.\_Reticule.color = new Color(0.0f, 1f, 0.0f, 0.784f);

 }

 else

 {

 this.SpTarget.gameObject.SetActive(false);

 this.\_Reticule.rectTransform.localPosition = Vector3.zero;

 this.\_Reticule.transform.eulerAngles = this.\_camera.transform.eulerAngles;

 this.\_Reticule.color = new Color(1f, 0.0f, 0.0f, 0.784f);

 }

 if ((Object) this.\_CloseM != (Object) null)

 this.Tdist = Vector3.Distance(this.transform.position, this.\_CloseM.transform.position);

 if ((double) this.Tdist > (double) this.\_MagnetRange)

 this.\_CloseM = (GameObject) null;

 this.InputMagnets();

 if (!this.\_orbiting)

 this.DetachMagnet();

 this.\_cDelay += Time.deltaTime;

 }

 private void FixedUpdate()

 {

 if ((double) this.\_cDelay < (double) this.\_aGDelay || (double) this.rBody.velocity.y <= -20.0)

 return;

 this.rBody.AddForce(Vector3.down \* this.\_aGravity, ForceMode.Acceleration);

 }

 private void DetachMagnet()

 {

 if (!((Object) this.\_CloseM != (Object) null) || this.\_orbiting)

 return;

 Vector3 viewportPoint = this.\_camera.WorldToViewportPoint(this.\_CloseM.transform.position);

 if ((double) viewportPoint.z > 0.0 && (double) viewportPoint.x < 1.0 && ((double) viewportPoint.x > 0.0 && (double) viewportPoint.y < 1.0) && (double) viewportPoint.y > 0.0)

 return;

 this.\_CloseM = (GameObject) null;

 }

 public void DetachOrbit()

 {

 if (!this.\_orbiting)

 return;

 this.\_orbiting = false;

 this.rBody.velocity = this.\_oPoint.forward \* this.vitesseIn;

 this.rBody.useGravity = true;

 this.\_CloseM = (GameObject) null;

 }

 private void InputMagnets()

 {

 if (Input.GetMouseButtonDown(0))

 {

 if ((Object) this.\_potentialTarget != (Object) this.\_CloseM && (Object) this.\_CloseM == (Object) null)

 {

 this.\_CloseM = this.\_potentialTarget;

 this.SoundMagnet.gameObject.SetActive(true);

 this.SoundMagnet.position = this.\_CloseM.transform.position;

 }

 else

 {

 this.\_CloseM = (GameObject) null;

 this.SoundMagnet.gameObject.SetActive(false);

 }

 }

 if (Input.GetMouseButtonUp(0))

 {

 if (!((Object) this.\_CloseM != (Object) null))

 ;

 this.\_CloseM = (GameObject) null;

 this.SoundMagnet.gameObject.SetActive(false);

 }

 if (Input.GetMouseButtonDown(1))

 {

 if ((Object) this.\_potentialTarget != (Object) this.\_CloseM && (Object) this.\_CloseM == (Object) null)

 {

 this.\_CloseM = this.\_potentialTarget;

 this.SoundOrbit.gameObject.SetActive(true);

 this.SoundOrbit.position = this.\_CloseM.transform.position;

 }

 else

 {

 this.\_CloseM = (GameObject) null;

 this.SoundOrbit.gameObject.SetActive(false);

 }

 }

 if (Input.GetMouseButtonUp(1))

 {

 this.\_CloseM = (GameObject) null;

 this.SoundOrbit.gameObject.SetActive(false);

 this.DetachOrbit();

 }

 if (Input.GetMouseButton(0) && (Object) this.\_CloseM != (Object) null)

 {

 this.\_cDelay = 0.0f;

 Vector3 normalized = (this.\_CloseM.transform.position - this.transform.position).normalized;

 Vector3 zero = Vector3.zero;

 float num = 0.0f;

 if ((double) normalized.y <= 0.0)

 {

 if ((double) this.rBody.velocity.y > 0.0)

 num = (this.\_CloseM.transform.position - this.transform.position).y \* -this.coefRecup;

 }

 else if ((double) this.rBody.velocity.y < 0.0)

 num = (this.\_CloseM.transform.position - this.transform.position).y \* this.coefRecup;

 this.rBody.AddForce((normalized \* this.\_attraction + num \* Vector3.up) \* Time.deltaTime, ForceMode.Acceleration);

 }

 if (!Input.GetMouseButton(1) || !((Object) this.\_CloseM != (Object) null))

 return;

 this.rBody.velocity = Vector3.Lerp(this.rBody.velocity, (this.\_CloseM.transform.position - this.transform.position).normalized \* this.rBody.velocity.magnitude, 0.05f).normalized \* this.rBody.velocity.magnitude;

 }

 private void CheckCloseMagnets()

 {

 float magnetRange = this.\_MagnetRange;

 float num = 5f;

 GameObject gameObject = (GameObject) null;

 foreach (GameObject magnet in MagnetManager.Magnets)

 {

 if ((double) Vector3.Distance(this.transform.position, magnet.transform.position) <= (double) this.\_MagnetRange)

 {

 Vector3 viewportPoint = this.\_camera.WorldToViewportPoint(magnet.transform.position);

 viewportPoint.y = (float) (0.5 + ((double) viewportPoint.y - 0.5) / (double) this.ScreenRatio);

 if ((double) viewportPoint.z > 0.0 && (double) Vector2.Distance(new Vector2(0.5f, 0.5f), new Vector2(viewportPoint.x, viewportPoint.y)) < (double) this.viseurSize / (double) this.ScreenRatio \* (double) Screen.width / 1000.0 \* 0.5 / (double) Screen.width && (double) Vector2.Distance(new Vector2(0.5f, 0.5f), new Vector2(viewportPoint.x, viewportPoint.y)) < (double) num)

 {

 num = Vector2.Distance(new Vector2(0.5f, 0.5f), new Vector2(viewportPoint.x, viewportPoint.y));

 gameObject = magnet;

 }

 }

 }

 if ((Object) this.\_potentialTarget != (Object) gameObject)

 {

 if (!((Object) this.\_potentialTarget != (Object) null))

 ;

 this.\_potentialTarget = gameObject;

 }

 if ((Object) gameObject != (Object) null)

 ;

 }

 public void BubbleIn()

 {

 this.rBody.drag = 2f;

 this.WaterBlue.enabled = true;

 this.\_camera.GetComponent<CC\_Wiggle>().enabled = true;

 }

 public void BubbleOut()

 {

 this.rBody.drag = 0.1f;

 this.WaterBlue.enabled = false;

 this.\_camera.GetComponent<CC\_Wiggle>().enabled = false;

 }

 private void InitOrbit()

 {

 this.Axis = Vector3.Cross(this.rBody.velocity, this.transform.position - this.\_CloseM.transform.position);

 this.vitesseIn = this.rBody.velocity.magnitude;

 this.\_center.transform.position = this.\_CloseM.transform.position;

 this.\_center.LookAt(this.\_CloseM.transform.position + this.Axis);

 this.\_oPoint.position = this.transform.position;

 this.\_oPoint.LookAt(this.\_oPoint.position + this.rBody.velocity);

 this.\_oPoint.parent = this.\_center;

 this.rBody.useGravity = false;

 this.rBody.velocity = Vector3.zero;

 }

 private void Orbit()

 {

 this.\_center.transform.RotateAround(this.\_center.transform.position, this.Axis, (float) -(180.0 \* (double) this.vitesseIn / (double) Vector3.Distance(this.\_center.position, this.\_oPoint.position) / 3.14159274101257) \* Time.deltaTime);

 this.transform.position = this.\_oPoint.position;

 }

 private void OnCollisionStay(Collision other)

 {

 this.scoreReduc += Time.deltaTime \* 2f;

 this.\_timeNoHit = 0.0f;

 if ((double) this.scoreReduc < 1.0)

 return;

 --this.scoreReduc;

 SCORE.multipli -= 0.1f;

 }

 private void OnCollisionEnter(Collision other)

 {

 this.\_timeNoHit = 0.0f;

 SCORE.AnneauCombo = 0;

 SCORE.PrismCombo = 0;

 }

}