# Osnove virtualnih okruženja Vježba 3

Slika na kojoj se prikazuje snimka zaslona, Multimedijski softver, grafički softver, softver videoigre

Opis je automatski generiran

* Implementirano lijepljenje objekata
* Implementirano opadanje objekata
* Implementirana hijerarhija objekata
* Implementiran prikaz hijerarhije

## Implementacija lijepljenja / dopadanja objekata

### Kod: PlayerColision.cs

using System.Collections.Generic;

using System.Collections;

using UnityEngine;

/// <summary>

/// Class that implements actions during a collision.

/// </summary>

public class PlayerCollision : MonoBehaviour

{

    private List<(float, (float, float, float, bool), string, Transform, GameObject)> lastChlist = new List<(float, (float, float, float, bool), string, Transform, GameObject)>();

    [SerializeField] private GameObject \_stickyItems = null;

    private Dictionary<string, float> \_oldMassesOfStickyItems = null;

    /// <summary>

    /// Start is called before the first frame update.

    /// Initialize storage for masses of sticky items.

    /// </summary>

    private void Start()

    {

        \_oldMassesOfStickyItems = new Dictionary<string, float>();

    }

    /// <summary>

    /// On a collision check if the collision object has a tag "Sticky", otherwise ignore it.

    /// If an object has a tag "Sticky" and has a smaller mass than katamari ball, add it to the katamari ball,

    /// store it's mass to the storage and add it to the katamari ball's mass.

    /// If an object has a tag "Sticky" and has a greater mass than katamari ball, get its previous mass and

    /// substract katamari ball's mass by it and remove the object with it's previous mass.

    /// </summary>

    /// <param name="collision">Object that katamari ball has collided with.</param>

    private void OnCollisionEnter(Collision collision)

    {

        // float lastChildMass = null;

        // Rigidbody lastChildRb = null;

        // string lastChildName = null;

        // Transform lastChildParrent = null;

        // GameObject lastChild = null;

        float childMass = collision.gameObject.GetComponent<Rigidbody>().mass;

        Rigidbody childRb = collision.gameObject.GetComponent<Rigidbody>();

        string childName = collision.gameObject.name;

        float katamariMass = GetKatamariMass();

        // consider the collision only if the object is sticky, its mass is smaller than katamariMass and if the sticky object is not already stuck on the ball

        // Consider the collision only if the object is sticky, its mass is smaller than katamariMass and if the sticky object is not already stuck on the ball

        if (collision.gameObject.CompareTag("Sticky") && !\_oldMassesOfStickyItems.ContainsKey(childName))

        {

            Debug.Log($"Object: {childName}, Mass: {childMass}");

            if (katamariMass > childMass)

            {

                // save last child data

                lastChlist.Add((childMass, (childRb.mass, childRb.drag, childRb.angularDrag, childRb.useGravity), childName, collision.transform, collision.gameObject));

                // Add the collided object to the \_oldMassesOfStickyItems

                \_oldMassesOfStickyItems.Add(childName, childMass);

                // Extract the Collider component from the collision object with GetContact(...)

                Collider collidedObjectCollider = collision.GetContact(0).thisCollider;

                // Set the parent of the collided object

                collision.transform.SetParent(collidedObjectCollider.transform);

                // Update the Katamari mass

                UpdateKatamariMass(childMass, "add");

                // after merging the ball with the collided object, it is necessary to destroy the RigidBody component to facilitate their movement as a singular entity

                Destroy(childRb);

            }

            else{

                if (lastChlist.Count > 0)

                {

                    // Revert the mass and re-parent the last child object

                    var lastCh = lastChlist[lastChlist.Count - 1];

                    lastChlist.RemoveAt(lastChlist.Count - 1);

                    UpdateKatamariMass(lastCh.Item1, "substract");

                    lastCh.Item5.transform.SetParent(null);

                    // \_oldMassesOfStickyItems.Remove(lastCh.Item3);

                    StartCoroutine(DetachAndRevert(lastCh));

                }

            }

        }

    }

    private IEnumerator DetachAndRevert((float, (float, float, float, bool), string, Transform, GameObject) lastCh)

    {

        // Short delay to ensure detachment

        UpdateKatamariMass(lastCh.Item1, "subtract");

        lastCh.Item5.transform.SetParent(null);

        // Add a Rigidbody back to the last child object

        Rigidbody rb = lastCh.Item5.AddComponent<Rigidbody>();

        rb.mass = lastCh.Item2.Item1;

        rb.drag = lastCh.Item2.Item2;

        rb.angularDrag = lastCh.Item2.Item3;

        rb.useGravity = lastCh.Item2.Item4;

        yield return new WaitForSeconds(1.5f);

        \_oldMassesOfStickyItems.Remove(lastCh.Item3);

    }

    /// <summary>

    /// A helper function to get katamari ball's current mass.

    /// </summary>

    /// <returns>Mass of the katamari ball</returns>

    private float GetKatamariMass()

    {

        return gameObject.GetComponent<Rigidbody>().mass;

    }

    /// <summary>

    /// A helper function to change katamari ball's mass.

    /// </summary>

    /// <param name="mass">Mass to be added to katamari ball.</param>

    /// <param name="operation">Determines the type of operation that will be performed (substraction or addition)</param>

    private void UpdateKatamariMass(float mass, string operation)

    {

        if (operation == "add")

            gameObject.GetComponent<Rigidbody>().mass += mass;

        else if (operation == "substract")

            gameObject.GetComponent<Rigidbody>().mass -= mass;

        else

            Debug.Log("Error while doing operation.");

    }

}

* Koristi se Lista za pohranu svojstava svih objekata spojenih na igraća, te se pomoću nje miču objekti prilikom sudara težom stvari.
* U listi su sva svojstva tog predmeta koja se potom njemu ponovo pridodaju te je moguće opet pokupiti predmet nakon opadanja.
* Parent od objekta se postavlja na transform od colidera te mu predmet s kojim je sudaren postaje roditelj i time se dobije hijerarhija.
* Implementirana je pomoćna funkcija za micanje objekta u kojoj je delay od 1.5s kako se objekt koji se odvoji ne bi opet odmah spojio. (private IEnumerator DetachAndRevert)

## Prikaz mase i hijerarhija na ekranu

### Kod: PlayerInformation.cs

using UnityEngine;

using UnityEngine.UI;

namespace Assets.Scripts

{

    /// <summary>

    /// Class for viewing information about hierarchy and katamari mass

    /// </summary>

    public class PlayerInformation : MonoBehaviour

    {

        [SerializeField] private Text \_katamariMass = null;

        [SerializeField] private Text \_hierarchy = null;

        private Rigidbody \_katamari = null;

        private Renderer [] \_renderers = null;

        /// <summary>

        /// Start is called before the first frame update.

        /// Initialize Rigidbody component.

        /// </summary>

        private void Start()

        {

            \_katamari = GetComponent<Rigidbody>();

        }

        /// <summary>

        /// Update is called once per frame.

        /// Get rendered objects and update the view of hierarchy and katamari ball mass

        /// </summary>

        private void Update()

        {

            \_katamariMass.text = $"Masa: {\_katamari.mass}";

            string hir = hierarchyText(\_katamari.transform, 0);

            \_hierarchy.text = $"Kamari: \n {hir}";

        }

        private string hierarchyText(Transform current, int depth){

            string ret="";

            // Debug.Log($"djeca : {current.GetComponentsInChildren<Transform>()}");

            if(current.childCount>0){

                for(int x=0;x<current.childCount;x++){

                    Transform child=current.GetChild(x).transform;

                    Debug.Log($":transform\_ch: {child}");

                    for (int i=0;i<depth;i++){

                        ret=ret+"--";

                    }

                    ret=ret + child.name + "\n";

                    if(child.childCount>0){

                        ret=ret + hierarchyText(child, depth+1);

                    }

                }

                Debug.Log($":: {ret}");

                return ret;

            }

            else{

                return ret;

            }

        }

    }

}

* Implementirana je Rekurzivna funkcija (private string hierarchyText) koja prima transform objekta i vraća imena djece tog objekta te dodaje -- za svaku dodatnu dubinu kao prefiks, za svako dijete provjerava ima li djecu te ako ima povećava dubinu i ponavlja to.
* Slika na kojoj se prikazuje tekst, snimka zaslona, Multimedijski softver, softver

  Opis je automatski generiran
* Tu je prikaz ta dva Text objekta kojima skripta mijenja tekst.
* Masa se samo dobije od samog objekta od Rigidbody komponente.

## Dodatne slike:

Slika na kojoj se prikazuje snimka zaslona, crtić

Opis je automatski generiran

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

Slika na kojoj se prikazuje snimka zaslona, crtić, Animacija, igra za PC

Opis je automatski generiran

Slika na kojoj se prikazuje snimka zaslona, softver videoigre, igra za PC, Animacija

Opis je automatski generiran

Opadanje stvari